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

MATEUS PAIVA FOGAÇA

# A New Quadratic Formulation for Incremental Timing-Driven Placement

Dissertation presented in partial fulfillment
of the requirements for the degree of
Master of Microeletronics

Advisor: Prof. Dr. Ricardo Reis

Porto Alegre
November 2016

## CIP — CATALOGING-IN-PUBLICATION

Fogaça, Mateus Paiva

A New Quadratic Formulation for Incremental Timing-Driven Placement / Mateus Paiva Fogaça. – Porto Alegre: PGMICRO da UFRGS, 2016.

88 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2016. Advisor: Ricardo Reis.

1. Timing optimization. 2. Placement. 3. Physical Design. 4. Electronic Design Automation. 5. Microelectronics. I. Reis, Ricardo. II. Título.

*“Caia sete vezes; levante-se oito.”*

— PROVÉRBIO JAPONÊS

## AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer a minha família cujo apoio permitiu chegar até aqui. Agradeço em especial ao meu pai, Odilon Fogaça, minha mãe Dóris Fogaça e minha avó Arlete Fogaça pela confiança depositada em mim.

Também gostaria de agradecer ao meu orientador Ricardo Reis por me aceitar como seu orientando e aos colegas Jucemar Monteiro e Guilherme Flach, por compartilharem sua experiência comigo. Agradeço aos meus ex-orientadores e agora amigos Cristina Meinhardt e Paulo Butzen pelo apoio que me deram para chegar até aqui.

Por fim, gostaria de deixar o agradecimento aos amigos Pierre Calado, Pablo Minuto, Pedro O. Ribeiro, Robson Canez e Silas Amaral. Todos vocês também me motivam a seguir em frente. Muito Obrigado!

## ABSTRACT

The interconnection delay is a dominant factor for achieving timing closure in nanoCMOS circuits. During physical synthesis, placement aims to spread cells in the available area while optimizing an objective function w.r.t. the design constraints. Therefore, it is a key step to determine the total wirelength and hence to achieve timing closure. Incremental placement techniques aim to improve the quality of a given solution. Two quadratic approaches for incremental timing driven placement to mitigate late violations through path smoothing and net load balancing are proposed in this work. Unlike previous works, the proposed formulations include a delay model into the quadratic function. Quadratic placement is applied incrementally through an operation called *neutralization* which helps to keep the qualities of the initial placement solution. In both techniques, the quadratic wirelength is pondered by cell's drive strengths and pin criticalities. The final results outperform the state-of-art by 9.4% and 7.6% on average for WNS and TNS, respectively.

**Keywords:** Timing optimization. Placement. Physical Design. Electronic Design Automation. Microelectronics.

**Uma nova formulação quadrática para posicionamento incremental guiado à tempos de propagação**

## RESUMO

O tempo de propagação dos sinais nas interconexões é um fator dominante para atingir a frequência de operação desejada em circuitos nanoCMOS. Durante a síntese física, o posicionamento visa espalhar as células na área disponível enquanto otimiza uma função custo obedecendo aos requisitos do projeto. Portanto, o posicionamento é uma etapa chave na determinação do comprimento total dos fios e, consequentemente, na obtenção da frequência de operação desejada. Técnicas de posicionamento incremental visam melhorar a qualidade de uma dada solução. Neste trabalho, são propostas duas abordagens para o posicionamento incremental guiado à tempos de propagação através de suavização de caminhos e balanceamento de redes. Ao contrário dos trabalhos existentes na literatura, a formulação proposta inclui um modelo de atraso na função quadrática. Além disso, o posicionamento quadrático é aplicado incrementalmente através de uma operação, chamada de *neutralização*, que ajuda a manter as qualidades da solução inicial. Em ambas as técnicas, o comprimento quadrático de fios é ponderado pelo *drive strength* das células e a criticalidade dos pinos. Os resultados obtidos superam o estado-da-arte em média 9,4% e 7,6% com relação ao WNS e TNS, respectivamente.

**Palavras-chave:** Otimização de *Timing*. Posicionamento. Projeto Físico. Automação do Projeto Eletrônico. Microeletrônica.

# LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| ABU | Average Bin Utilization |
| ASIC | Application-Specific Integrated Circuits |
| AWE | Asymptotic Waveform Evaluation |
| B2B | Bound-to-Bound |
| BFS | Breadth-First Search |
| CAD | Computer-Aided Design |
| CMOS | Complementary Metal Oxide Semiconductor |
| EDA | Electronic Design Automation |
| e.g. | for example |
| eTNS | Early Late Total Negative Slack |
| HDL | Hardware Description Language |
| HPWL | Half-Perimeter Wirelength |
| ICCAD | International Conference on Computer-Aided Design |
| i.e. | in other words |
| IO | Input/Output |
| ITOP | Integrating Timing Optimization within Placement |
| LCB | Local clock buffer |
| lTNS | Late Total Negative Slack |
| LUT | Look-Up Table |
| lWNS | Late Worst Negative Slack |
| MOR | Model Order Reduction |
| PADe | Posicionador Analítico-Detalhado |
| RC | Resistance-Capacitance |
| RTL | Register Transfer Level |

| | |
|---|---|
| STA | Static timing analysis |
| StWL | Steiner Tree Wirelength |
| TDP | Timing-Driven Placement |
| TNS | Total Negative Slack |
| VLSI | Very Large Scale Integration |
| w.r.t. | with respect to |
| WNS | Worst Negative Slack |

# LIST OF SYMBOLS

| | |
|---|---|
| $\Phi$ | Quadratic objective value |
| $x$ | Position on x-axis |
| $A$ | Hessian matrix |
| $\vec{b}$ | Fixed value vector |
| $C$ | Capacitance (fF) |
| $R$ | Resistance (k$\Omega$) |
| $\alpha$ | Tunning Value |
| $V_{th}$ | Threshold Voltage (V) |
| $d$ | Delay (ps) |
| $T$ | Clock period (ps) |
| $t$ | Time (ps) |
| $g$ | Weight for an edge |
| $\phi$ | Percentage of congestion regions to be considered during ABU estimation |

**LIST OF FIGURES**

# LIST OF TABLES

# CONTENTS

# 1 INTRODUCTION

The feature sizes of integrated circuits have experienced an impressive and steadily reduction since the first chips were released. However, from a speed point of view, scaling does not benefit interconnections as it benefits transistors. As metal dimensions shrink, the wire resistance increases due to reduced cross section and their capacitance also increases due to reduced spacing and decreased thickness (ROSSUM, 2009). In newer technology nodes, like 130nm and below, interconnections overcame gates internal timing propagation and became the dominant factor in the total circuit delay (GARGINI, 2000).

Since placement is the main step defining the interconnection length, it is now a central step to achieve timing closure. The classical placement objective – wirelength minimization – continues to be the core goal of placement as optimizing the total wirelength helps to achieve other objectives as routability and timing (YANG; CHOI; SARRAFZADEH, 2002). However, timing optimization cannot be ignored during placement and simply aiming timing indirectly via wirelength minimization is not sufficient. Therefore timing optimization techniques need to be incorporated within placement, creating the algorithms categorized as timing-driven placement. Timing-driven techniques may be divided into Global and Incremental ones (ALPERT; MEHTA; SAPATNEKAR, 2008).

Global techniques perform the placement of the entire circuit, not necessarily respecting previous solutions. They usually apply net-weighting techniques (KONG, 2002; TSAY; KOEHL, 1991; BURSTEIN; YOUSSEF, 1985) to prioritize nets with high violations or assign a max wirelength/delay for them. These techniques can deal with a lot of violations at the same time, keeping a global view of the problem. However, while these nets are optimized, other violations may show up and, thereby, new constraints need to be created or the existing weights updated. This behavior may lead to oscillations and convergence issues (VISWANATHAN et al., 2010).

Incremental techniques aim to improve an existing solution by moving just a reduced number of cells, commonly through path-based techniques (VISWANATHAN et al., 2010). The key idea is to reduce the critical paths lengths by smoothing it through applying local search heuristics (BOCK et al., 2015; FLACH et al., 2016) or linear programming (LUO; NEWMARK; PAN, 2006). One drawback of these techniques is the loss of the global view of the problem since linear programming runtime may be prohibitive for a large number of cells.

Algorithms may also combine placement with other techniques such as buffering (HANI; SHAIKH-HUSIN, 2008), discrete gate sizing and V$_{th}$ selection (FLACH et al., 2014) to further optimize the solution (GARGINI, 2000). Large nets may be divided into several smaller nets by hierarchical buffer insertion, which may reduce both the wire delay and the signal degradation. Gate sizing and V$_{th}$ selection may speed up signals to reduce setup violations and slow down the signals presenting hold violations. Timing-driven placement may reduce the load of critical nets, minimizing the need for these techniques and mitigating the increase in power consumption.

This work addresses quadratic placement techniques which are commonly applied to solve the wirelength-driven placement problem (VISWANATHAN; CHU, 2005; LIN et al., 2013; KLEINHANS et al., 1991). In these techniques, circuit netlist is modeled as a graph $G = (V, E)$, where the cells, macros and primary inputs and outputs nodes are the set of vertices V, and the nets are decomposed into a set of binary edges $E$. Placement is thus solved as a spring system, where nets represent attraction forces between circuit nodes and additional spreading forces are added to the system to spread cells in the circuit area. This same approach can be applied to timing-driven placement as seen in (MONTEIRO et al., 2015; VISWANATHAN et al., 2010). However, quadratic and other force-directed works usually try to optimize timing by assigning higher weights to nets proportionally to their timing violation, which roughly addresses timing.

The analytical techniques in literature deal with incremental timing driven placement with two methodologies:

1. Only cells, nets and/or paths that are being optimized are considered movable, while the rest of the circuit is considered fixed (LUO; NEWMARK; PAN, 2006; BOCK et al., 2015; FLACH et al., 2016);
2. All nodes can be moved, however, the mathematical formulation that generates the initial solution is known and thus additional weights/forces are added to the system to optimize timing (SRINIVASAN; CHAUDHARY; KUH, 1991; RIESS; ETTELT, 1995).

In this work, an incremental quadratic approach is proposed. It relies on an operation called *neutralization* – Given the current solution and the circuit netlist, the classic spring system is constructed. The neutralization finds for each cell, an additional force such that the equilibrium point in the spring system is the current position.

This approach allows improving a given placement incrementally using the quadratic formulation.

## 1.1 The design flow of integrated circuits

Most designs of Application-Specific Integrated Circuits (ASICs) are made relying on standard-cell libraries. In this methodology, cells are previously characterized and tested, providing more reliability and smaller time-to-market to the design (REIS et al., 2000). Although the complete flow may have different features depending on the development team, a general flow is depicted in Figure 1.1.

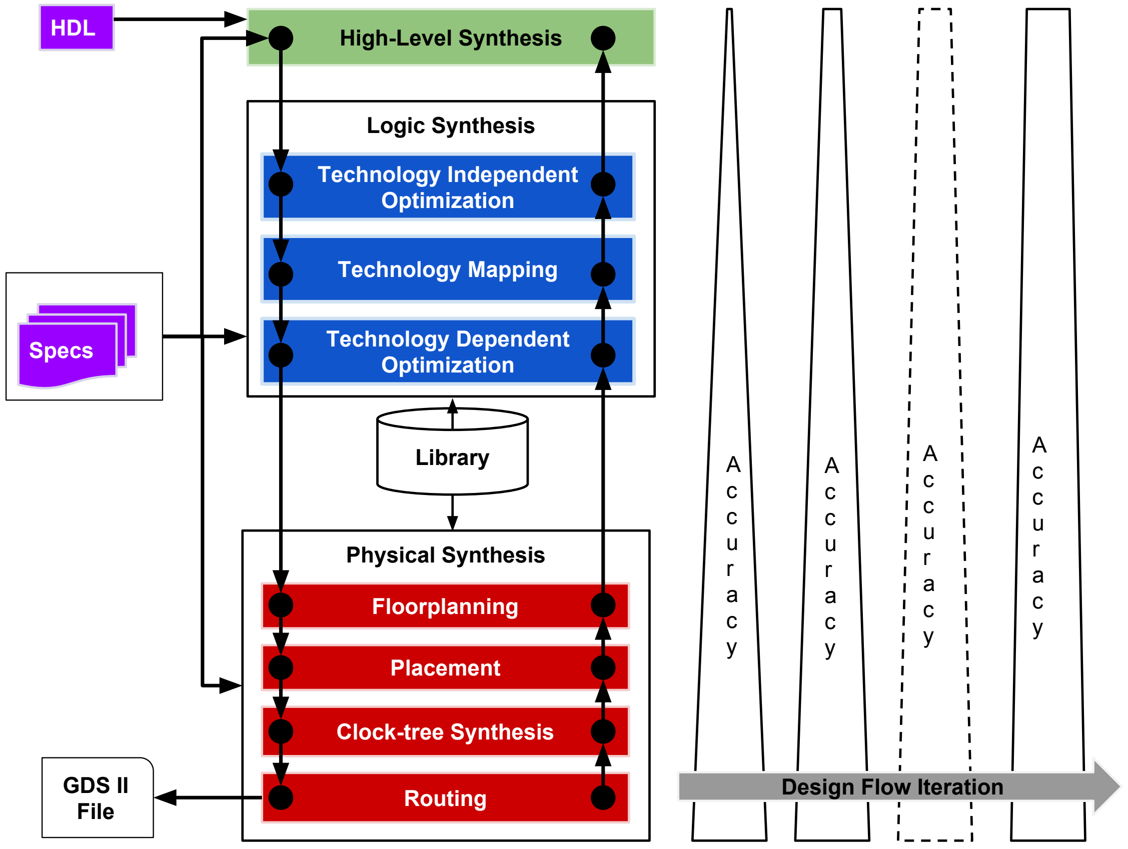
An initial modeling of the system is made using a hardware description language (HDL) which allows different level of abstraction. During high-level synthesis, the system description may be translated several times, starting from the algorithmic level to Register Transfer Level (RTL) and Boolean equations (WANG; CHANG; CHENG, 2009).

The next stage is the logic synthesis, composed of three steps: Technology independent optimization, technology mapping and technology dependent optimization. The former aims to minimize the circuit's logic applying Boolean and arithmetic optimization techniques. In the technology mapping, logic functions are mapped to standard-cells available in the library, e.g. NANDs, NORs and AOIs. The last step, called technology dependent optimization, optimizes the circuit regarding the electrical information of the standard-cells, like timing, area, $V_{th}$ and power (WESTE; ESHRAGHIAN, 1985).

In the physical synthesis, the logic representation is transformed into a geometric representation, where the shapes represent the masks of materials that will be sent the foundry for manufacturing. In the floorplanning step, it is identified the modules that should be placed together and their respective placement, the IO interface and aspect ratio of the chip w.r.t. the available area (WESTE; ESHRAGHIAN, 1985). After the floorplanning, placement properly defines the exact location where each cell must be placed in the die trying to optimize the total wirelength, timing, area and power (HENTSCHKE, 2007). Once the cells locations are defined, the clock tree synthesis performs the interconnection of the clock sources with sequential elements trying to minimize the relative gap of time that the clock signal takes to reach each one (FLACH, 2010). In the last step, routing defines the exact route each signal will travel between the driver pin and its sinks (JOHANN, 2001; REIMANN, 2013).

As the flow is being performed, the designers get a more precise idea of the final solution. If any step fails to achieve a feasible solution, it is possible to go up in the flow and reexecute the earlier steps using additional information obtained from later steps. After the physical synthesis is successfully concluded, the circuit is ready to be manufactured. However, the full set of design constraints is hardly respected in a single iteration of the design flow. In fact, the flow needs to be restarted several times to achieve the design closure (FLACH, 2015).

Figure 1.1: The design flow of digital integrated circuits.



Source: Adapted from (FLACH, 2015).

## 1.2 Contributions

The contributions of this work may be summarized in the following items:

- A review about the state-of-art of quadratic formulations applied to timing-driven placement;
- Two approaches to integrate timing optimization into an incremental quadratic placement modeling – One for path smoothing and other for load balancing;
- A flow integrating the quadratic formulation with the local search algorithms previously proposed in (FLACH et al., 2016);

The dissertation is organized as follows: Chapter 2 presents the theoretical foundation; Some previous works are presented in Chapter 3; Chapter 4 explores the adopted methodology to apply quadratic placement incrementally; The quadratic formulations for incremental placement are presented in Section 4.2 while the proposed placement flow is presented in Section 4.3; Chapter 5 discusses the experimental results and in Chapter 6 are given the final remarks.

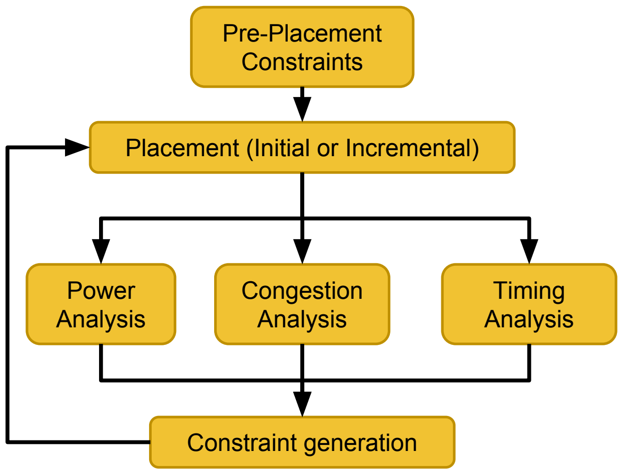
# 2 THEORETICAL FOUNDATION

In this chapter, some concepts are presented. Initially, an overview about placement algorithms is provided in Section 2.1 and the mathematical formulation of quadratic placement tools is further discussed in Section 2.2. The rest of the chapter explains how the timing analysis was performed in this work.

## 2.1 Placement

Placement is an open challenge widely explored in industry and academia for more than 50 years (ALPERT et al., 2012). Its solution has direct influence in the final routing quality and, therefore, in the circuit performance. In the early stages of the design, placement optimizes other metrics indirectly, by optimizing the total wirelength w.r.t the density constraints. As it became possible to get a fair estimation of the final routing, a more precise modeling is needed to cope with timing constraint and help on achieving design closure. Placement may be also executed several times, dealing with different constraints, as illustrated in Figure 2.1. An initial solution is produced by the first iteration and then passes trough several kinds of analysis. New constraints are generated and added to the system which tries to further optimize the solution incrementally. Each placement iteration, initial or incremental, is traditionally composed of three steps: Global Placement, legalization and detailed placement, as discussed in this section. The three steps are depicted in Figure 2.2. This work proposes a global formulation for quadratic placement.

Figure 2.1: Placement Flow



Source: Inspired by (ALPERT et al., 2012)

### 2.1.1 Global Placement

Global placement (Figure 2.2a) aims to spread cells through the chip area while minimizing an objective function, like wirelength, w.r.t. some constraints like area, congestion or timing. This step has been the target of a wide number of research in the literature. The techniques for global placement may be divided into three categories (WANG; CHANG; CHENG, 2009) – Partitioning, meta-heuristics and analytical algorithms. Partitioning techniques solve the problem by applying graph partitioning techniques in the netlist and subdividing the chip area recursively until the search space becomes small enough to apply local search techniques. Some academic tools like Capo (ROY et al., 2005) and FengShui (AGNIHOTRI; ONO; MADDEN, 2005) are examples of partitioning-based tools. Meta-heuristics for placement are commonly based on Simulated Annealing (HADDOCK; MITTENTHAL, 1992), and may also be applied on the flat netlist, like in TimberWolf (SECHEN; SANGIOVANNI-VINCENTELLI, 1986) or applied in the latter stages of partitioning tools, like in Dragon (WANG; YANG; SARRAFZADEH, 2000).

State-of-art algorithms usually model placement as an analytic function which they solve by applying mathematical techniques. Analytical tools may be subdivided into quadratic and non-quadratic ones according to their objective function. The quadratic ones can be efficiently solved by minimizing a system of linear equations (SPINDLER; JOHANNES, 2007) while non-linear methods may adopt more complex objective functions at the cost of runtime. FastPlace (VISWANATHAN; CHU, 2005), SimPL (KIM; LEE; MARKOV, 2013) and POLAR (LIN et al., 2013) are examples of quadratic placement tools while Kraftwerk (SPINDLER; JOHANNES, 2007), ePlace (LU et al., 2016) and NTUPlace (CHEN et al., 2007) are examples of non-quadratic tools.

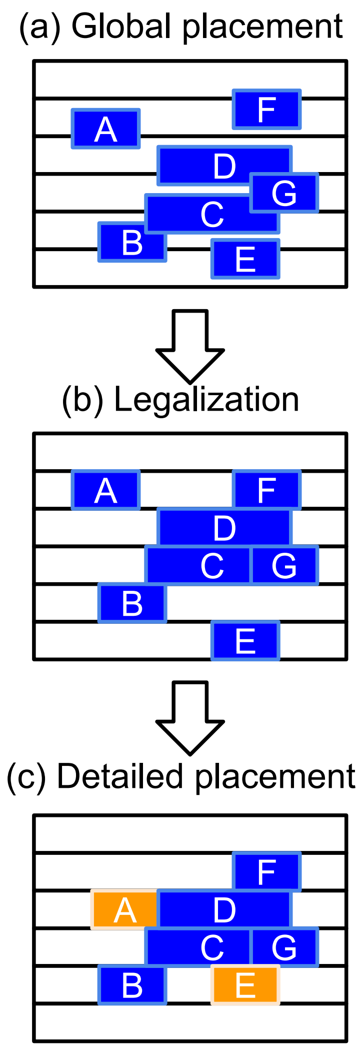
### 2.1.2 Legalization

Due to the high complexity of modern netlists, global tools may not be aware of overlap between cells or alignment to sites in rows, producing infeasible solutions. Legalization (Figure 2.2b) algorithms like Tetris (HILL, 2002), Abacus (SPINDLER; SCHLICHTMANN; JOHANNES, 2008) and Jezz (PUGET et al., 2015) fix these violations by moving the cells to legal locations with minimum impact to the global solution.

### 2.1.3 Detailed Placement

Many cells may be moved during legalization, impacting negatively on the quality of the solution. To cope with this issue, detailed placement performs local refinement to smooth the impact of the legalization and also to further optimize the solution. Unlike the global placement, in this case, the legality constraints are observed. In the example of Figure 2.2c, only the cells $A$ and $E$ are moved. Branch-and-Bound (BREUER, 1977; CALDWELL; KAHNG; MARKOV, 1999), Domino (DOLL; JOHANNES; ANTREICH, 1994), FastDP (PAN; VISWANATHAN; CHU, 2005) and BraveDP (POPOVYCH et al., 2014) are examples of algorithms for detailed placement.

Figure 2.2: Traditional placement steps.



Source: The author.

## 2.2 Quadratic placement

The quadratic placement formulation for general applications traces back to Hall (HALL, 1970) and it was successfully applied to circuit design by GORDIAN (KLEINHANS et al., 1991). Years later, FastPlace authors (VISWANATHAN; CHU, 2005) adapted the formulation to support VLSI circuits up to 10x faster than other academic tools, making it widely applied to global placement in academy and industry, until nowadays (LIN et al., 2013).

Quadratic placement can be physically interpreted as a spring system where the cells, macros and IO pads are dimensionless dots and the nets are attraction forces among them. However, this formulation only supports node-to-node connections. Therefore, nets with more than 2 pins have to be decomposed into node-to-node connections. The system can be solved independently for each axis, which is traditionally done.

The general objective function is shown in Equation 2.1 using matrix notation.

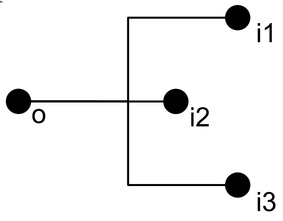
$$\Phi(x) = \frac{1}{2}\vec{x}^{\,T} A\vec{x} + \vec{b_x}^{\,T}\vec{x} + constant \tag{2.1}$$

where $\vec{x}$ represents the current cells positions for $x$ axis, $A$ is the Hessian matrix from the objective function and $\vec{b_x}$ represents the fixed forces for axis $\vec{x}$.

### 2.2.1 Wirelength models

The quadratic function only supports 2-pin connections. Since the circuit netlist is a hyper-graph and nets may connect more than 2 pins, these hyper-edges must be adapted to fit the quadratic formulation. The main approaches in the literature are discussed bellow using the 4-pin connection illustrated in Figure 2.3.

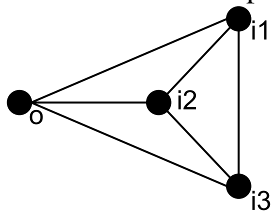
Figure 2.3: A multi-pin connection between pins $o$, $i1$, $i2$ and $i3$.



Source: The author.

One simple approach is to decompose the hyper-edge into $n(n-1)/2$ edges, where $n$ is the number of pins in the net. This approach is traditionally called Clique model, or complete graph and is shown in Figure 2.4. Each new edge is commonly weighted by a factor $1/(n-1)$ in order to balance the forces in the system.

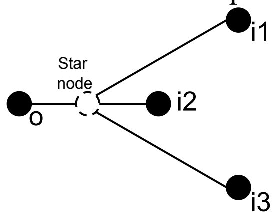
Figure 2.4: Clique model for the net presented in Fig. 2.3.



Source: The author.

Alternatively, Figure 2.5 presents the star model, which creates a virtual node, called Star. The Star is connected to the other nodes of the net by binary edges whose weight factor commonly is $n/(n-1)$.

Figure 2.5: Star model for the net presented in Fig. 2.3.



Source: The author.

The Clique model is efficient for small nets because does not create additional variables to the system while the Star model has more null elements in the Hessian matrix, which helps the solver convergence. Viswanathan et. al. (VISWANATHAN; CHU, 2005) proved analytically that using the presented edge weighting approaches, both models converge to the same result and, therefore, may be combined to achieve a good trade-off between additional variables and the number of non-zero elements in the matrix. In their empirical experiments, they concluded that using the Clique model for 2 and 3-pin nets and star model for 4-pin nets and beyond the system solver runtime was 5 times faster than using the clique model alone.

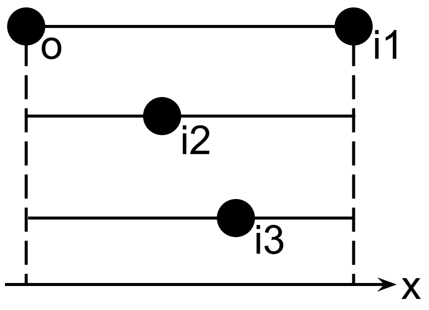
Finally, a model called Bound2Bound (B2B) or Bounding-Box Net Model was proposed by Spindler et. al. (SPINDLER; SCHLICHTMANN; JOHANNES, 2008). In this model, the pins are ordered according to their current position in a given axis. An edge between the boundary pins is created and 2 edges between each inner pin and the boundary pins by a factor:

$$w = \frac{1}{(n-1)|x_i - x_j|} \tag{2.2}$$

where $n$ is the number of pins of the net and $x_i$ and $x_j$ are the positions of the nodes being connected.

Figure 2.6 presents a possible example, considering the net from Figure 2.3. This model has two key advantages: The first is that the number of total edges is smaller than in the Clique model without the addition of new variables to the system. The second is that this model addresses the half perimeter of the nets more precisely than the former models. The major drawbacks, however, is that a connection matrix is necessary for each axis individually. Furthermore, these matrices need to be rebuilt after a certain number of iterations since the boundary pins may change while solving the linear system.

Figure 2.6: B2B model for the net presented in Fig. 2.3.

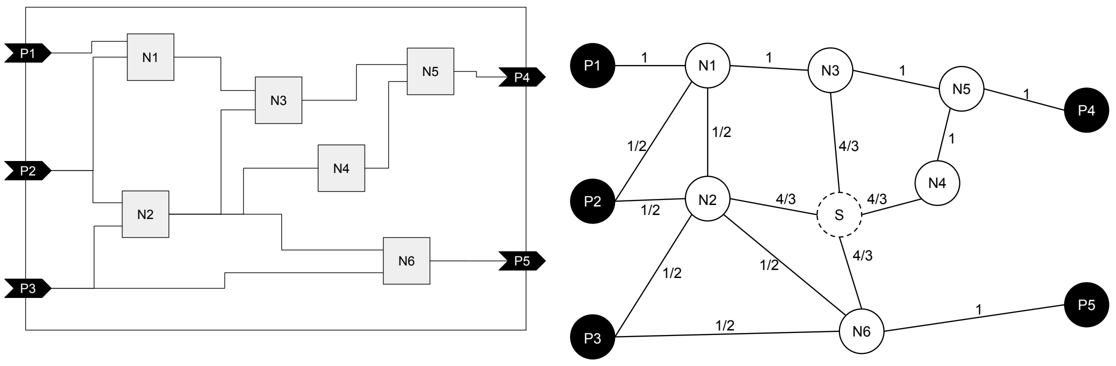


Source: The author.

### 2.2.2 Building and solving the quadratic formulation

Consider the circuit presented in Figure 2.7(a), composed of 6 cells, 5 ports and 9 nets. The first step to apply the quadratic formulation is to generate the graph representation of the netlist, identifying and decomposing the multi-pin nets. Figure 2.7(b) depicts a possible graph originated from the circuit using the hybrid net model.

Figure 2.7: The graph representation of a netlist using hybrid net model. The 4-pin net connecting $N2$, $N3$, $N4$ and $N6$ is replaced by a star node, called $S$.



(a) Original Circuit (b) Graph representation

Source: The author.

Once the graph is known, one may build the Hessian matrix, which represents the interconnections between nodes and its weight. The value for a given element $A_{ij} | i \neq j$ is the **negative** weight of the edge connecting nodes $i$ and $j$, the position is set as 0 if there is no edge connecting the nodes. Note that only connections between movable nodes are represented in the Hessian matrix. For diagonal positions, the value is set as the **positive** value of the summation of all edge weights of the node, including connections with fixed nodes. The Hessian matrix for the example circuit is presented in Equation 2.3. In the proposed example, movable nodes are cells and fixed nodes are I/O pins, however, in real VLSI circuits, some cells and IP blocks may have been fixed during the floorplaning.

$$A = \begin{array}{c} \\ N1 \\ N2 \\ N3 \\ N4 \\ N5 \\ N6 \\ S \end{array} \begin{array}{c} \begin{array}{ccccccc} N1 & N2 & N3 & N4 & N5 & N6 & S \end{array} \\ \begin{pmatrix} 3 & -\frac{1}{2} & -1 & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & \frac{10}{3} & 0 & 0 & 0 & -\frac{1}{2} & -\frac{4}{3} \\ -1 & 0 & \frac{10}{3} & 0 & -1 & 0 & -\frac{4}{3} \\ 0 & 0 & 0 & \frac{7}{3} & -1 & 0 & -\frac{4}{3} \\ 0 & 0 & -1 & -1 & 3 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & 0 & 0 & \frac{10}{3} & -\frac{4}{3} \\ 0 & -\frac{4}{3} & -\frac{4}{3} & -\frac{4}{3} & 0 & -\frac{4}{3} & \frac{16}{3} \end{pmatrix} \end{array} \quad (2.3)$$

Vector $\vec{b_x}$ represents the connection with fixed nodes. As it takes into account the coordinates of fixed points, the vector must be build for each axis independently. Considering the given example presented previously, let the $x$ coordinates for the I/O pins to be $x_{p1} = x_{p2} = x_{p3} = 2$ and $x_{p4} = x_{p5} = 10$. Each position of vector is the summation of the weights of edges connecting the node to the fixed point multiplied by its position.

The vector $\vec{b_x}$ for the given example is shown in Equation 2.4.

$$\vec{b} = \begin{array}{c} N1 \\ N2 \\ N3 \\ N4 \\ N5 \\ N6 \\ S \end{array} \begin{pmatrix} 3 \\ 2 \\ 0 \\ 0 \\ 10 \\ 10 \\ 0 \end{pmatrix} \tag{2.4}$$

The derivative $\nabla\Phi(x) = A\vec{x} - \vec{b_x} = 0$ is solved to find the optimal positions for the cells (ALPERT et al., 1997). Solving the example, it is found the following values:

$$\vec{x} = \begin{array}{c} N1 \\ N2 \\ N3 \\ N4 \\ N5 \\ N6 \\ S \end{array} \begin{pmatrix} 3.44 \\ 4.09 \\ 5.29 \\ 6.08 \\ 7.12 \\ 5.73 \\ 5.30 \end{pmatrix} \tag{2.5}$$

## 2.3 Static timing analysis

Timing is traditionally estimated through static timing analysis (STA). The delay models used in STA depend on the information available and accuracy/runtime trade-off required at each stage or optimization step of the design flow. During early stages, simpler interconnection models are preferred like the linear delay and Elmore delay (ELMORE, 1948), as many changes are being performed and the exact routing is not defined yet. On the other hand, more accurate models like Asymptotic Waveform Evaluation (AWE) (PILLAGE; ROHRER, 1990) and Model Order Reduction (MOR) are preferred during late stages of the design flow.

### 2.3.1 Routing estimation

Routing wirelength can be estimated using several strategies as the bounding box (CALDWELL et al., 1998), driver-to-sink, minimum spanning tree (ZHENG; LIM; IYENGAR, 1996) and minimum Steiner tree (CHU, 2004). With the exception of the bounding box, the other strategies also provide an estimate of the routing topology being the Steiner tree the closer one related to the final routing. Regarding only the wirelength, the bounding box matches exactly the wirelength of the minimum Steiner tree for nets with 2 and 3 pins, which are the vast majority of nets in a design.

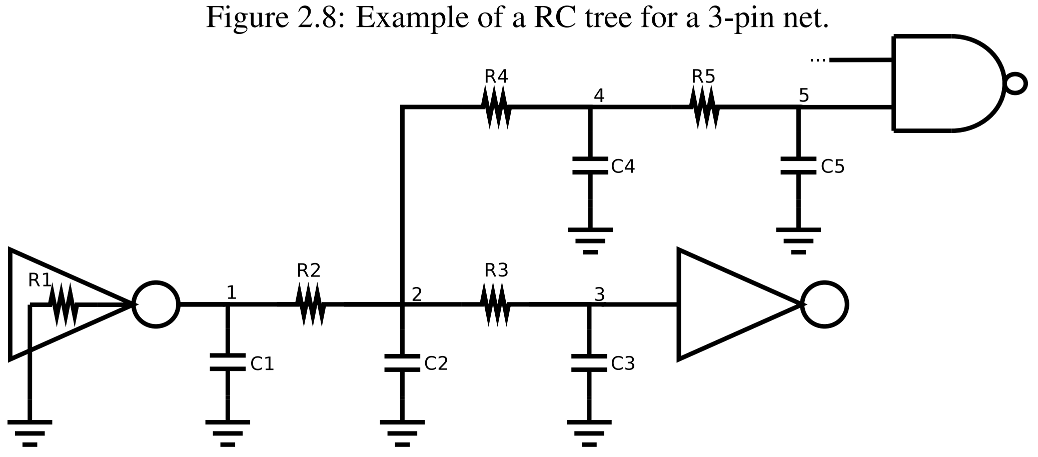
### 2.3.2 Elmore delay model

Elmore delay (ELMORE, 1948) presents a good trade-off between accuracy and performance when compared to linear delay (fast) and AWE (precise) respectively. To compute the Elmore delay of a net the wire is divided into segments and each segment is modeled as a RC circuit, composing a RC tree. The value for the delay in each segment is calculated propagating the delay from the driver to the current segment until it reaches a sink. The general equation for a segment $n$ is given by Equation 2.6, Figure 2.8 presents an example of a RC tree and Equations 2.7-2.11 while the delay calculation for each wire segment.

$$d_i = d_{up} + R_i \times C_{down} \tag{2.6}$$

where $d_i$ is the delay in the node $i$; $d_{up}$ is the upstream delay; $R_i$ is the wire resistance and $C_{down}$ is the downstream capacitance.

Figure 2.8: Example of a RC tree for a 3-pin net.



Source: The author.

$$d_1 = R_1 \times (C_1 + C_2 + C_3 + C_4 + C_5) \tag{2.7}$$

$$d_2 = d_1 + R_2 \times (C_2 + C_3 + C_4 + C_5) \tag{2.8}$$

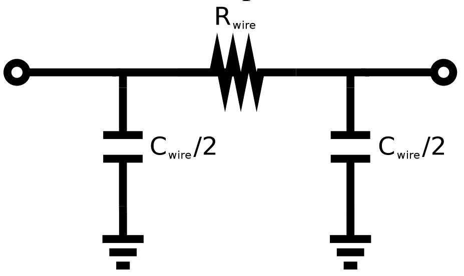
$$d_3 = d_2 + R_3 \times (C_3) \tag{2.9}$$

$$d_4 = d_2 + R_4 \times (C_4 + C5) \tag{2.10}$$

$$d_5 = d_4 + R_5 \times (C5) \tag{2.11}$$

Elmore may be applied applied to a simplified $\Pi$ network with a parasitic $R_{wire}$ and $C_{wire}$, as shown in Figure 2.9 which delay is given by Equation 2.12.

Figure 2.9: $\Pi$ model representation for a net.



Source: The author.

$$d_{wire} = \frac{R_{wire} \times C_{wire}}{2} + R_{wire} \times C_{load} \tag{2.12}$$

The first term is divided by 2 since only the second capacitor, $C_{wire}/2$, "sees" the resistance $R_{wire}$. In the second term, $C_{load}$ is the capacitance being loaded. The factors $C_{wire}$ and $R_{wire}$ are calculated as follows:

$$R_{wire} = R_{unit} \times L \tag{2.13}$$

$$C_{wire} = C_{unit} \times L \tag{2.14}$$

where $R_{unit}$ and $C_{unit}$ are technology parameters and $L$ is the estimated wirelength.

### 2.3.3 Gate propagation delay and slew

The modern standard cell libraries provide Look-Up Tables (LUTs). The values for cell delay and slew may be represented as a function of the input slew and output capacitance, for instance. These values usually are obtained from experimental measures or from simulations using complex models (BHASKER; CHADHA, 2009). An example of a set of $3X4$ delay tables for an inverter are shown in Figure 2.10 using Liberty format (OPENSOURCE. . . , 2016).

Figure 2.10: Example of a LUT for an inverter using Liberty format.

```
cell ("INV_X1") {
  pin ("o") {
    direction : output ;
    capacitance : 0.0 ;
    max_capacitance : 12.80 ;
    min_capacitance : 0.00 ;
    timing() {
      cell_fall ("delay_outputslew_template_3X4") {
       index_1 ("0.00,1.00,2.00") ; /* Input transition */
       index_2 ("5.00,30.00,50.00,80.00") ; /* Output capacitance */
       values (\
       "14.064, 21.864, 27.204",\
       "20.316, 28.116, 34.32",\
       "26.556, 34.356, 40.596",\
       "39.06, 46.86, 53.1",\
       );
      }
      fall_transition ("delay_outputslew_template_3X4") {
       index_1 ("0.00,1.00,2.00") ; /* Input transition */
       index_2 ("5.00,30.00,50.00,80.00") ; /* Output capacitance */
       values (\
       "15, 16.464, 19.656",\
       "22.5, 23.04, 25.38",\
       "30, 30.108, 31.692",\
       "45, 45, 45.384, 48.12",\
       );
      }
      cell_rise ("delay_outputslew_template_3X4") {
       index_1 ("0.00,1.00,2.00") ; /* Input transition */
       index_2 ("5.00,30.00,50.00,80.00") ; /* Output capacitance */
       values (\
       "14.064, 21.864, 27.204",\
       "20.316, 28.116, 34.32",\
       "26.556, 34.356, 40.596",\
       "39.06, 46.86, 53.1, 62.46",\
       );
      }
      rise_transition ("delay_outputslew_template_3X4") {
       index_1 ("0.00,1.00,2.00") ; /* Input transition */
       index_2 ("5.00,30.00,50.00,80.00") ; /* Output capacitance */
       values (\
       "15, 16.464, 19.656",\
       "22.5, 23.04, 25.38",\
       "30, 30.108, 31.692",\
       "45, 45, 45.384",\
       );
      }
      timing_sense : negative_unate ;
      related_pin  : "a" ;
    }
```

Source: The author.

Interpolation is commonly adopted to obtain intermediate values from the table, using the following formulation (BHASKER; CHADHA, 2009):

$$\begin{aligned} d_{00} = x_{20} \times y_{20} \times d_{11} + x_{20} \times y_{01} \times d_{12} + \\ x_{01} \times y_{20} \times d_{21} + x_{01} \times y_{01} \times d_{22} \end{aligned} \tag{2.15}$$

$$x_{01} = \frac{x_0 - x_1}{x_2 - x_1} \tag{2.16}$$

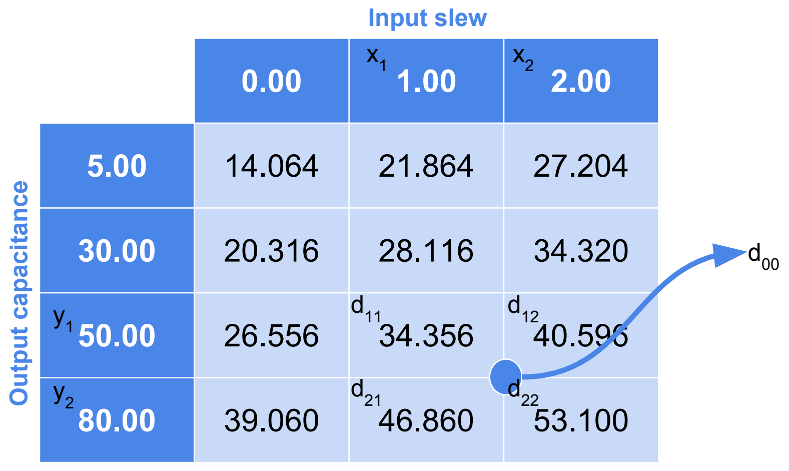
$$x_{01} = \frac{x_2 - x_0}{x_2 - x_1} \tag{2.17}$$

$$y_{01} = \frac{y_0 - y_1}{y_2 - y_1} \tag{2.18}$$

$$y_{20} = \frac{y_2 - y_0}{y_2 - y_1} \tag{2.19}$$

where $d_{00}$ is the desired value of delay for a given value of input slew $x_0$ and of an output capacitance $y_0$. The values $x_1$ and $x_2$ are the closest left and right neighbor in the table regarding $x_0$. The same stands for the $y$s. Finally, $d_{ij}$ is an entry in the table for a pair $(x_i, y_j)$. An example of interpolation is depicted in Figure 2.11.

Figure 2.11: Assume an inverter from Figure 2.10 with an input slew of 1.6ps, driving a capacitance of 75fF. The values $x_1$, $x_2$, $y_1$, $y_2$, $d_{11}$, $d_{12}$, $d_{21}$ are $d_{22}$ for the calculation of cell falling transition are highlighted below.



| | 0.00 | $x_1$ 1.00 | $x_2$ 2.00 |
|---|---|---|---|
| 5.00 | 14.064 | 21.864 | 27.204 |
| 30.00 | 20.316 | 28.116 | 34.320 |
| $y_1$ 50.00 | 26.556 | $d_{11}$ 34.356 | $d_{12}$ 40.596 |
| $y_2$ 80.00 | 39.060 | $d_{21}$ 46.860 | $d_{22}$ 53.100 |

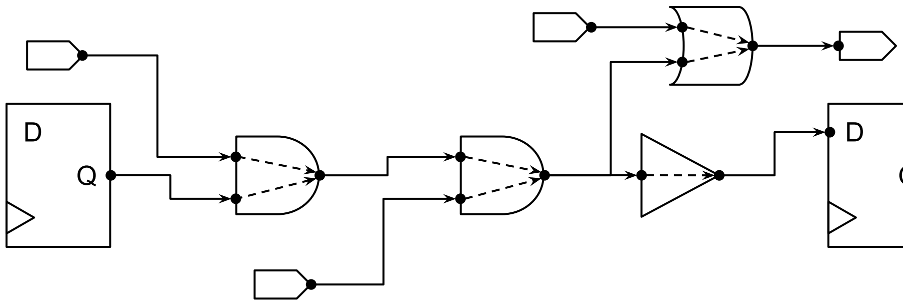
Input slew
Output capacitance
$d_{00}$

Source: The author.

### 2.3.4 Path delay

Once is determined how to estimate the delay for nets and cells, one can calculate the total propagation delay for a given signal in the circuit by traversing the timing graph of the netlist. Timing graph is a directed graph where nodes represent the circuit pins and the edges represent a path connecting these pins. The edges are commonly called timing arcs. An example of a timing graph is depicted in Figure 2.12. The start point may be a primary input of the chip while the endpoints may be a primary output of the chip or an input pin from a sequential element.

Figure 2.12: An example of a timing graph.



Source: The author.

STA may be performed in two modes: early and late. In early mode, it is considered the best case scenario, i.e. minimum propagation times, while late mode considers the worst case, i.e. maximum propagation times.

The arrival time ($t_a$) for an input pin ($p_i$) is given by:

$$t_{a-late}(p_i) = d(wire_{net}) + \max_{p \in net} [t_a(p)] \tag{2.20}$$

$$t_{a-early}(p_i) = d(wire_{net}) + \min_{p \in net} [t_a(p)] \tag{2.21}$$

where $d(wire)$ is the delay in the net driving the pin. The arrival time for an output pin is given by:

$$t_{a-late}(p_o) = d(cell) + \max_{p_i \,\in\, cell} [t_a(p_i)] \tag{2.22}$$

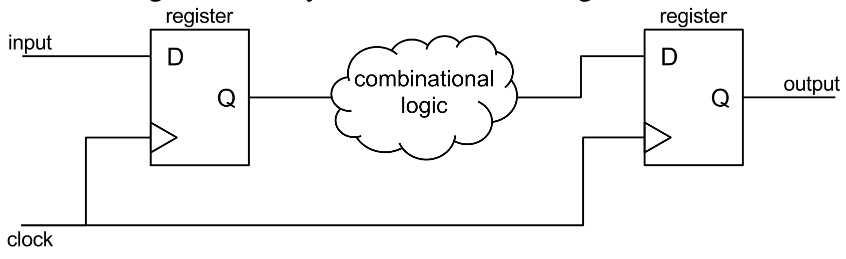
$$t_{a-early}(p_o) = d(cell) + \min_{p_i \,\in\, cell} [t_a(p_i)] \tag{2.23}$$

where $t_a$ is the arrival time for a pin and $d(cell)$ is the cell internal propagation delay. The LUTs specifies times for both rising and falling transitions w.r.t each input pin. Therefore, it is to store both values for each pin.

### 2.3.5 Timing violations

The large majority of the modern circuits is synchronous, which means they are subject to one or more clock domains. Figure 2.13 shows the general organization of these circuits, divided into sequential and combinational logic. The synchronous elements, mostly composed by registers are responsible for storing data for one or more clock periods while the combinational elements perform operations on them.

Figure 2.13: Synchronous circuit organization.



Source: The author.

In order to be stored correctly, the data must remain stable within a certain time range, determined by Equations 2.24 and 2.25.

$$T + t_{skew} - t_{Ck \,\rightarrow\, Q} - t_{comb} \;\geq\; t_{setup} \tag{2.24}$$

where $T$ is the clock period, $t_{Ck \,\rightarrow\, Q}$ is the register internal delay; $t_{comb}$ is the delay of the worst path in the combinational logic; $t_{setup}$ is the amount of time the data should be stable before a clock edge arrives at the register and $t_{skew}$ is the difference of time the clock signal arrives in the initial and the final registers.

$$t_{Ck \,\rightarrow\, D} + t_{comb} \;\geq\; t_{hold} + t_{skew} \tag{2.25}$$

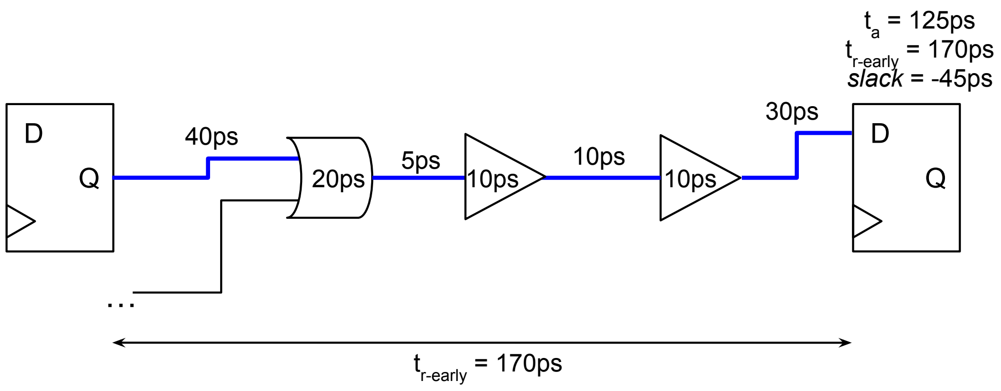
where $t_{Ck \rightarrow D}$ is the delay to the register begin storing the data after a clock edge; $t_{hold}$ is the time a data must remain stable after a clock edge and $t_{skew}$ is the difference of time the clock signal arrives in the initial and the final registers.

Knowing the target operation makes possible to calculate the minimum (early) and maximum (late) required arrival time that a signal must take to assure a correct storage. The difference between the required arrival time ($t_r$) and the actual arrival time ($t_a$) is called slack. If slack assumes a negative value for a pin, there is at least one path violating the timing constraints passing through the pin. The calculation of early and late slack are shown in Equations 2.26-2.27 and examples of path delay calculations are shown in Figures 2.14 and 2.15.

$$slack_{early} = t_a - t_{r-early} \tag{2.26}$$

$$slack_{late} = t_{r-late} - t_a \tag{2.27}$$

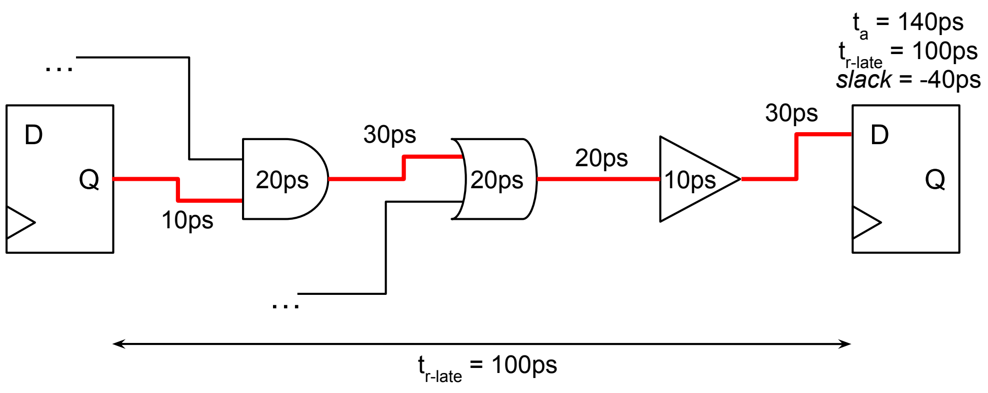
Figure 2.14: Example of a path with early violation.



Source: The author.

Henceforth, two important terms, called Worst Negative Slack (WNS) and Total Negative Slack (TNS), are employed referring to the most negative slack and the summation of all negative slacks at the endpoints of the circuit, respectively.

Figure 2.15: Example of a path with late violation.



Source: The author.

# 3 RELATED WORKS

Timing-driven placement algorithms have been addressed by several works in the literature. This chapter starts providing a short survey on timing optimization though quadratic formulations, which are the main focus of this work, from section 3.1 to 3.5. In section 3.6, a set of local search algorithms proposed by Bock et. al. is discussed due to their influence in our previous work presented in (FLACH et al., 2016). The former proposes several timing-driven single-cell movements which are discussed in Section 3.7.

## 3.1 RITUAL

A two-phase timing-driven placement algorithm is presented in RITUAL (SRINIVASAN; CHAUDHARY; KUH, 1991). The first phase is a continuous optimization of the weighted quadratic wirelength through quadratic placement and the latter is a discrete optimization using recursive partitioning and weighted assignment. In both phases, the weight is obtained by modeling timing constraints in a Lagrangian Relaxation formulation. The delays are estimated through Bakoglu (BAKOGLU, 1990) timing model and star net topology.

## 3.2 SPEED

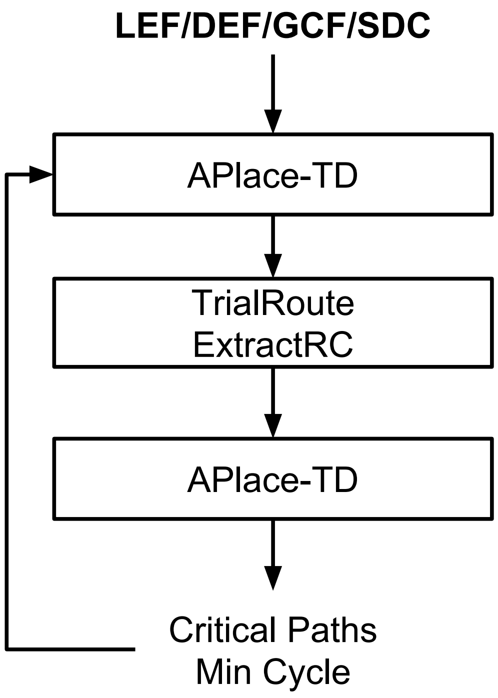
SPEED (RIESS; ETTELT, 1995) proposes a net weighting approach applied to quadratic placement. Net delays are estimated using the Elmore delay model and a star net topology. Initially, all weights of the circuit are set as the number of pins on the net. The weights are updated dynamically. As mentioned before, when a critical net is optimized, other may become critical. To avoid this kind of oscillation, the algorithm keeps a track of net criticality in the last two iterations. Once a net is evaluated as critical, the weight is increased. If the net stays non-critical for two iterations its weight is reduced by half.

## 3.3 APlace

In Kanhg et. al. (KAHNG; WANG, 2004), an extension of the wirelength-driven analytical tool APlace is proposed to support both mixed-size placement and timing-driven optimization. APlace performs global placement using a log-sum-exp method (NAYLOR; DONELLY; SHA, 2001) to address the linear wirelength optimization and a bell-shaped function to generate repulsion forces between cells in order to reduce congestion. The work copes with mixed-size circuits adapting the bell-shaped function to work with IP-blocks, which are much larger than cells.

The timing-driven methodology is shown in Figure 3.1. The output of the global placement stage is routed using the industrial tool TrialRoute and a RC extraction is performed on the final solution. Another tool, called Pearl, performs the STA. The information about obtained critical paths is incorporated by the global algorithm as net weights. Global placement is executed and these steps are repeated until the target value of improvement is achieved.

Figure 3.1: APlace timing-driven flow.

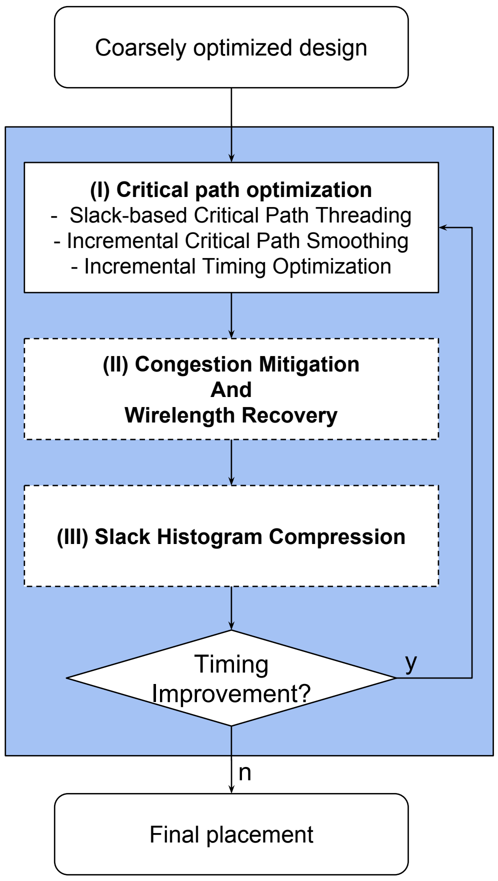


Source: Adapted from (KAHNG; WANG, 2004)

## 3.4 ITOP

ITOP (VISWANATHAN et al., 2010) is an incremental timing driven placement algorithm whose objective is to optimize timing w.r.t initial solution in terms of total wirelength and roteability. Figure 3.2 presents the proposed flow, which is divided into three major stages: (I) Critical path optimization, (II) Congestion mitigation and wirelength recovery and (III) slack histogram compression.

Figure 3.2: ITOP


Coarsely optimized design
(I) Critical path optimization
- Slack-based Critical Path Threading
- Incremental Critical Path Smoothing
- Incremental Timing Optimization
(II) Congestion Mitigation And Wirelength Recovery
(III) Slack Histogram Compression
Timing Improvement?
y
n
Final placement

Source: Adapted from (VISWANATHAN et al., 2010).

Stage I is subdivided into three steps. The first one is responsible for identifying critical paths and to create virtual 2-pin connections between elements of the path aiming to give higher priority to these paths in the global algorithm. In the second step, local movements, whose goal is to reduce the total path length, are applied in critical cells. An industrial static timer re-evaluates the solution. The last stage applies buffering and sizing techniques to optimized remaining critical paths.

Several locations of the chip may become congested due to the cell movements on stage I. To mitigate the generated congestion, non-critical cells in critical regions are identified and moved to the closest non-critical region. This strategy may degrade the solution regarding total wirelength. The authors deal with this issue applying a wirelength-driven detailed placement algorithm in the degraded solution.

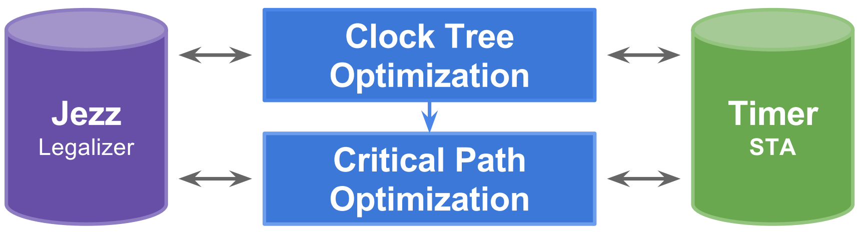
Continuous movements on top critical paths may degrade timing of non-critical paths. A slack histogram compression techniques are hence applied in Stage III for a set of non-critical paths. This stage covers a large set of paths which also helps the flow convergence.

## 3.5 PADe

The quadratic formulations discussed so far consider all cells to be movable and thus may have a huge impact on the initial solution. During incremental flows may be desired to keep the properties of the initial solution. Monteiro et. al. (MONTEIRO et al., 2015) adapted the quadratic formulation to move only a reduced group of cells.

PADe starts by traversing the netlist and setting all registers as fixed elements. In the following steps it is identified the most critical path in critical endpoints and it is set its cells and topological neighbors as movable elements. The remaining cells are set as fixed. Therefore, the quadratic formulation aims to find the optimal position only for cells in the critical paths and its neighbors. In (FLACH et al., 2015), PADe is combined with an algorithm that performs local movements in the registers targeting useful clock skew, as shown in Figure 3.3, a legalization algorithm and a static timer.

Figure 3.3: PADe flow and infrastucture.



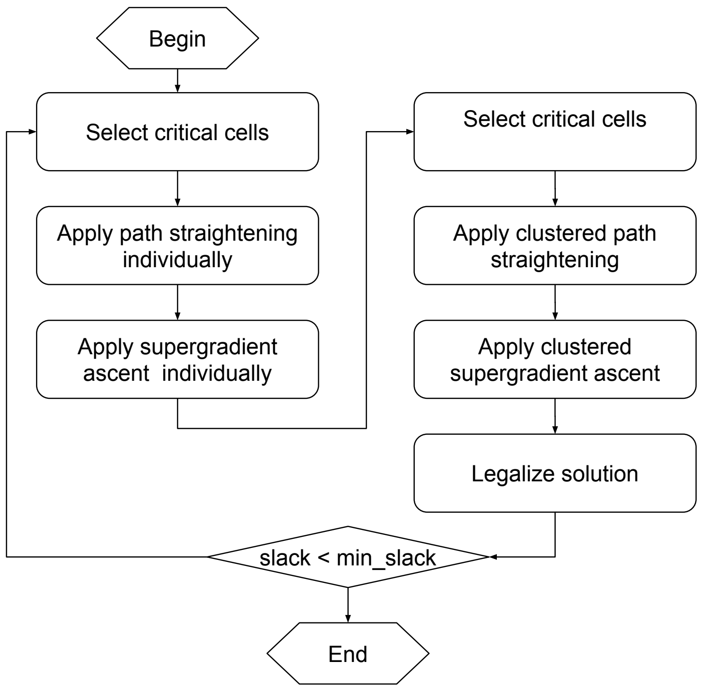
Source: (FLACH et al., 2015)

## 3.6 Bock et. al. local-search algorithms

A set of local search algorithms is proposed in (BOCK et al., 2015). The first technique focus on critical path straightening. For a given cell $c$, it is identified the most critical upstream pin and the most critical downstream pin. The cell is placed in the middle position between these two pins and timing is analyzed. Local movements are performed moving the cell towards each pin. The position of minimum slack is always kept until no further optimization is achieved. Another technique, called supergradient, assumes the slack is a concave function $f(x)$, where $x$ is the current position of the cell. The value of the slack gradient is computed shifting cells towards both positions horizontally and vertically. This procedure is repeated until the cell reaches the optimum value position.

The authors also demonstrate empirically that single cell movements may lead the algorithm to a local minima. A methodology for cell clustered based on slack and netlist topology is hence proposed. Figure 3.4 shows the flow integrating the techniques. Both path straightening and supergradient ascent are applied in single cells and clusters. Finally, the authors highlight that the presented methodology is timing-model independent and may be integrated with different STA tools.

Figure 3.4: (BOCK et al., 2015) flow.

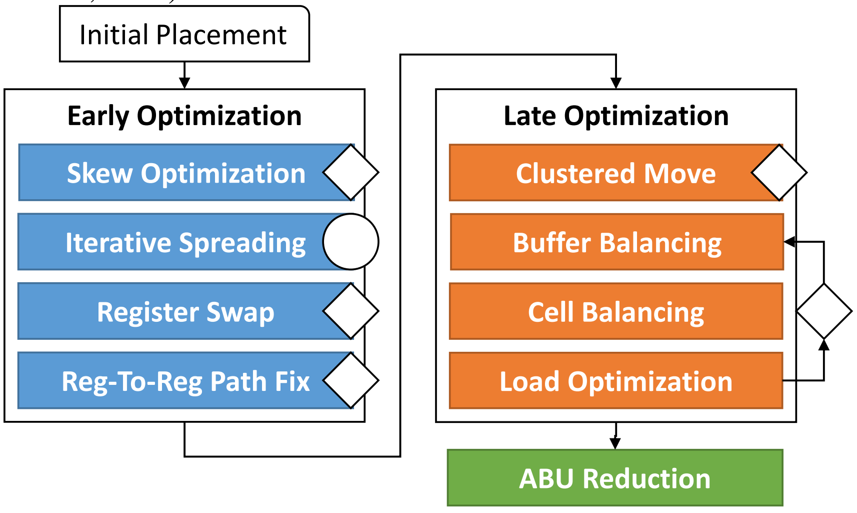

Begin
Select critical cells
Apply path straightening individually
Apply supergradient ascent individually
Select critical cells
Apply clustered path straightening
Apply clustered supergradient ascent
Legalize solution
slack < min_slack
End

Source: The author.

## 3.7 UFRGS analytical single-cell movement techniques

In a previous work (FLACH et al., 2016) a flow of techniques addressing both early and late timing violations was proposed. To mitigate early violations useful clock skew, iterative cell spreading, register swaps and register-to-register path fixing techniques are presented. Late violations are addressed through clustered movement (based on (BOCK et al., 2015)) and single cell movements aiming to reduce the load capacitance in the critical nets and balance load based on the cells drive strength. The techniques were integrated into the flow presented on Fig. 3.5 which presents the best known results for ICCAD 2015 CAD contest (KIM et al., 2015) on incremental timing-driven placement. The diamond shape indicates that the steps run until the quality of the result is not improved while the circle shape indicates that the quality of the result can degrade a certain number of times before exiting. The best solution found is restored (FLACH et al., 2016).

Figure 3.5: The incremental timing driven placement flow proposed by Flach et. al. (FLACH et al., 2016).



Source: (FLACH et al., 2016).

**Clustered Movement**

Bock et. al. (BOCK et al., 2015) demonstrated that single-cell movements may lead the algorithms to a local minima. To cope with this issue, a clustered movement approach is proposed. The clustering operation, based on Breadth-First Search (BFS), is described in Algorithm 3.1. From a given cell, $g$, every neighbor within a topological distance are traversed (lines 7-14). If the neighbor is critical, it is added to the list of cells, which represents the cluster (lines 9-13). The operation is repeated until no further critical neighbors are found (lines 4-15).

**Algorithm 3.1:** Cell clustering

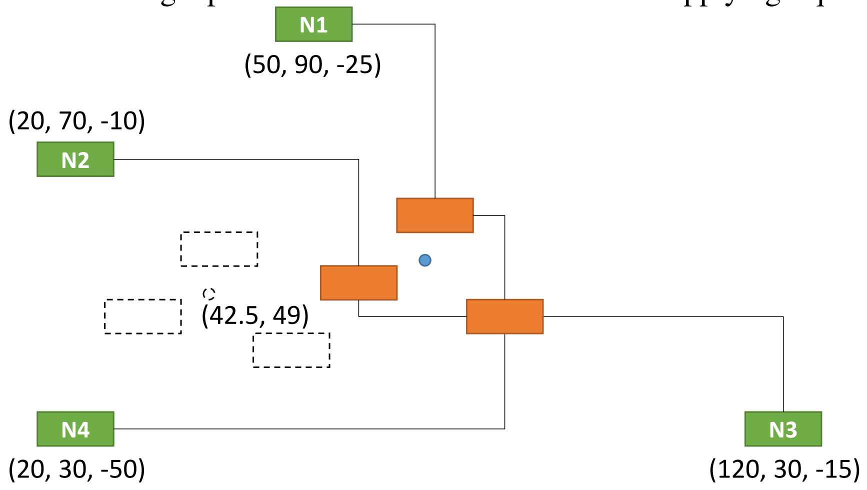
```
    Data: Initial cell $g$
    Result: Cluster $c$
 1  begin
 2      c $\leftarrow \emptyset$
 3      neighbors $\leftarrow g$
 4      while neighbors $\neq \emptyset$ do
 5          currCell $\leftarrow$ neighbors.top()
 6          neighbors.pop()
 7          if topological_distance(currCell, g) $< \gamma$ then
 8              cluster $\leftarrow$ cluster $+$ currCell
 9              for each neighbors $v \in$ currCell.neighbors do
10                  if criticality($v$) > 0 then
11                      neighbors $\leftarrow$ neighbors $+ v$
12                  end
13              end
14          end
15      end
16      return $c$
17  end
```

The next step is to identify the critical neighbors of the cluster and calculate their center of mass, pondered by the slacks of the pins connected to the cluster (Equation 3.1). Finally, every cell in the cluster is shifted towards the target position, as shown in Equation 3.2. Figure 3.6 depicts a cluster of three cells in orange and the neighbor critical cells in green. The target position is calculated as (42.5, 49), therefore, cells are shifted towards the dashed positions.

$$target\_pos = \frac{\sum_{i=0}^{P} pos(P_i) \times slack(P_i)}{\sum_{i=0}^{P} slack P_i}. \qquad (3.1)$$

$$new\_pos(cell) = pos(cell) - target\_pos - centerOfMass(grupo) \qquad (3.2)$$

Figure 3.6: The target position for a cluster. For each neighbor node N a tuple (x, y, slack) is specified and the target position for the cluster is calculated applying Equation 3.1.



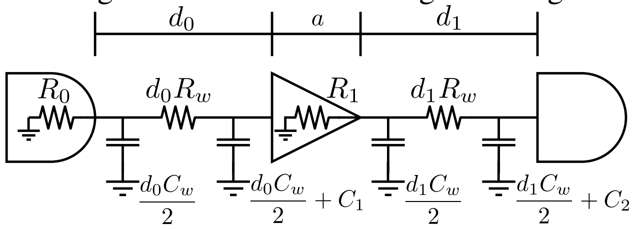
Source: (FLACH et al., 2016).

## Buffer Balancing

Buffer insertion is a common practice during physical design which decomposes large wires into smaller ones to avoid large parasitic capacitance and resistance, creating large buffer chains. However, buffer insertion may not be aware of different drive strengths and hence may restrict the optimization domain. The analytical formulation proposed in this section determines the position for a given buffer where the delay is minimum. The cells drive strength is taken into account, as well as, RC tree interconnection modelling and Elmore delay (ELMORE, 1948). Two simplifications are made: Since only one buffer is moved at a time, it is considered that its driver and sink will remain fixed. Moreover, there is only one driver and sink for each buffer. This idea is applied iteratively to handle buffer chains. Experimental data showed that only a few iterations are needed to achieve convergence.

An example is shown in Figure 3.7 and its delay ($D$) is described using Elmore delay in Equation 3.3

Figure 3.7: Buffer balancing modeling.



Source: (FLACH, 2015)

$$\begin{aligned} D = R_0\left(C_1 + d_0C_w\right) + d_0R_w\left(C1 + \frac{d_0C_w}{2}\right) + p_0 \\ + R_1\left(C_2 + d_1C_w\right) + d_1R_w\left(C2 + \frac{d_1C_w}{2}\right) + p_1 \end{aligned} \tag{3.3}$$

where $R_0$ represents the resistance of the cell driving the buffer; $C_1$ is the buffer input capacitance; $d_0$ is the interconnection length between the driver and the buffer; $R_w$ is the wire resistance per distance unit; $C_w$ is the wire capacitance per distance unit; $R_1$ is the resistance of the buffer; $d_1$ is the interconnection length from the buffer to the sink; $C_2$ is the sink load capacitance of the sink, while $p_0$ and $p_1$ are the internal delays from the driver and buffer, respectively.

The total distance between the buffer driver and the sink is given by, $d = d_0+a+d_1$ where $a$ is the distance between input and output pins of the buffer. By setting $\frac{\partial D}{\partial d_0} = 0$ one may find the minimum delay position for the buffer. The optimal distance between the driver and the buffer is shown in Equation 3.4.

$$d_0 = \frac{C_w\left(R_1 - R_0\right) + R_w\left[C_2 - C_1 + C_w\left(d - a\right)\right]}{2C_wR_w} \tag{3.4}$$

The new position for the buffer is calculated as follows.

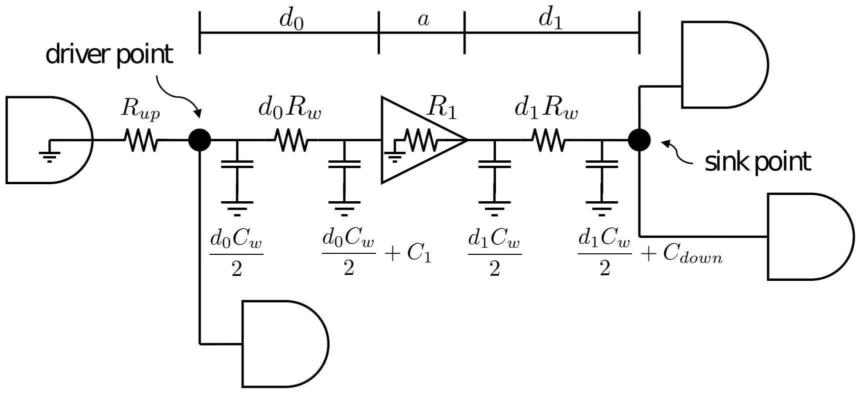
$$P_b = P_d + \frac{d_0}{d} \times \left(P_s - P_d\right) \tag{3.5}$$

where $P_b$ is the new position for the buffer, $P_d$ is the driver position and $P_s$ is the position of the sink.

**Cell ballancing**

The buffer balancing technique was extend do handle non-buffer cells and multi-pin nets. An optimal cell position for every timing arc is computed and then combined using a weighted arithmetic mean where, the weights are based on the negative slack and number of critical paths passing through it. The calculation is also simplified by restricting the cell movement to the closest Steiner point driving and sinking the cell, henceforth called driver point and sink point. The modelling is shown in Figure 3.8.

Figure 3.8: Cell balancing modeling.



Source: (FLACH, 2015)

In the following equations, consider upstream resistance ($R_{up}$) as the summation of every resistance from the driver cell up to the driver point, including the driver resistance; The upstream delay ($D_{up}$) as the propagation delay from the driver cell up to the driver point and the downstream capacitance $C_{down}$ as the summation of all capacitances from the sink point to all sinks. The total delay from the driver point up to the sink point may be represented as follows.

$$D = D_0 + D_1 \tag{3.6}$$

where

$$D_0 = D_{up} + R_{up}\left(C_1 + C_w d_0\right) + d_0 R_w\left(C_1 + \frac{d_0C_w}{2}\right) + p_0 \tag{3.7}$$

is the delay from the driver to the input of the cell and

$$D_1 = R_1 \left[C_{down} + d_1 C_w\right] + d_1 R_w \left[C_{down} + \frac{d_1 C_w}{2}\right] + p_1 \tag{3.8}$$

is the delay from the cell up to the sink point, where $C_1$ is the cell input capacitance; $d_0$ is the connection length between driver point and cell; $d_1$ is the connection length from the cell to the sink point; $R_1$ is the gate resistance and $p_0$ and $p_1$ are the drive and cell internal delay, respectively.

The Equation 3.4 may be rewritten to fit the new formulation, as follows.

$$d_0 = \frac{w_1 C_w R_1 - w_0 R_w C_1 + w_1 R_w \left[C_w(d-a) + C_{down})\right]}{R_w C_w (w_0 + w_1)} - \frac{w_0 R_{up} C_w}{R_w C_w (w_0 + w_1)} \tag{3.9}$$

where $w_0$ and $w_1$ are the driver cell and sink cell critical relevance, respectively.

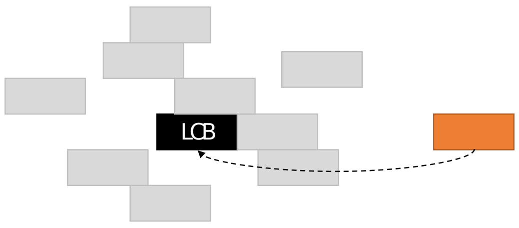
### Load Optimization

Noncritical sinks may be easily addressed to mitigate timing violations in nets with fan-out bigger than one by moving them closer to their driver. The main idea behind this strategy is to reduce the total interconnection length and hence the wire load capacitance and the resistance. Furthermore, by moving these sinks towards the driver reduce the cumulative parasitics in the downstream nodes of the RC tree.

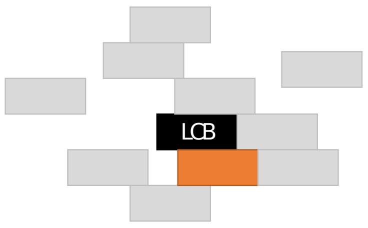
### Skew Optimization

Early violations may be mitigated decreasing the clock latency on the endpoint register. The simplest way to achieve that is moving the register towards the clock source, as depicted in Figure 3.9.

Figure 3.9: Clock skew optimization by moving registers closer to LCBs.



(a) Initial



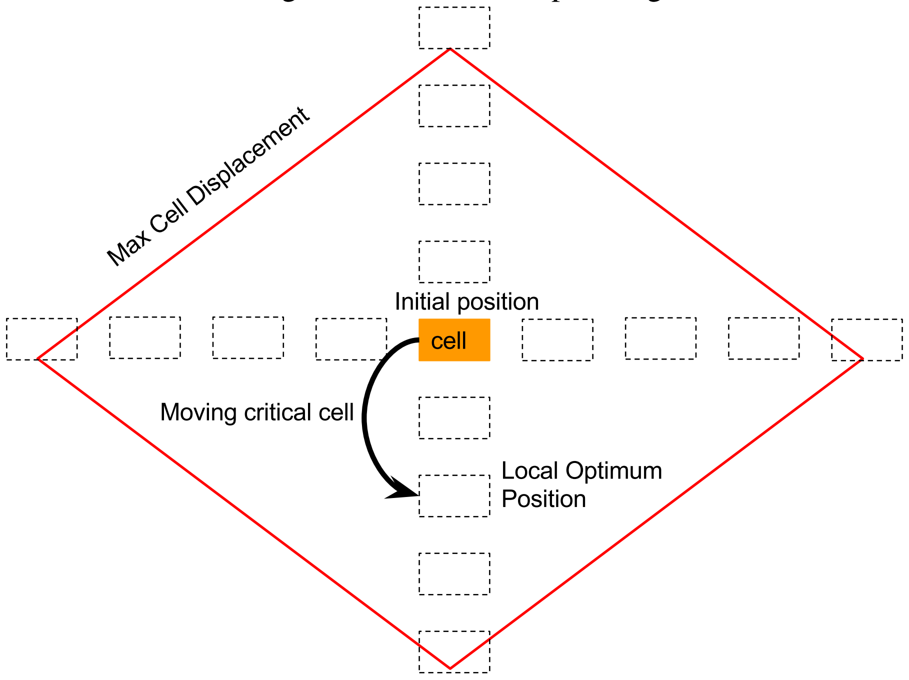
(b) Optimized

Source: (FLACH, 2015)

### Iterative Spreading

This technique tentatively shifts a given cell with early violation up, down, left and right – The best one is kept. A maximum displacement for the method is specified by the user. In the beginning, cells are shifted only 10% of the total displacement value. If there is no improvement, the search area is increased. Figure 3.10 depicts an example of the process, where the diamond shape represents the maximum displacement for the orange cell whose optimum position is found in the south.

Figure 3.10: Iterative Spreading



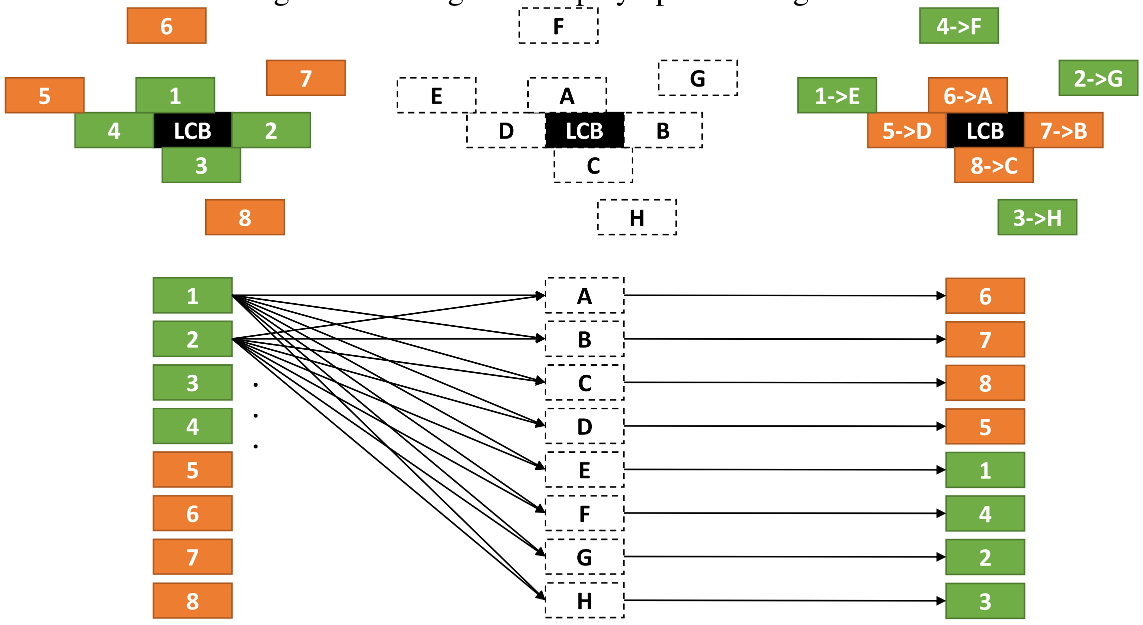
Source: (FLACH, 2015).

## Register Swap

By swapping the position of two or more register, the method tries to avoid the side effect of clock latency present in clock skew optimization. Assuming that all the registers have the same size and $V_{th}$, while swapping two registers being drived by a same clock source, the clock tree will not change and hence, the latency on each endpoint may be seen as a constant.

The register swap was modeled as an assignment problem which may optimally solved in polynomial time using the Hungarian algorithm (KUHN, 1955). Again, the idea is to mitigate in early paths is to reduce the clock latency at the endpoint registers. Figure 3.11 illustrated an example, where a LCB drivers 8 registers. In this case each slot is referred by a letter and it is desired to minimize the latency on the orange registers. Therefore, the algorithm assign them positions closer to the LCB.

Figure 3.11: Register swap by optimal assignment.



Source: (FLACH, 2015).

**Register-to-Register Path Fix**

In paths connecting directly two registers, the timing violation may be fixed by just increasing the interconnection parasitic between them. The zero-slack delay between two registers is given by:

$$d_{path}^{early} = l_o^{late} + t_{hold} - l_i^{early} \tag{3.10}$$

where $l_i^{early}$ and $l_o^{late}$ are the early and late clock latency at the clock pin of start and end point registers, respectively, $d_{path}^{early}$ is the early delay among the registers and $t_{hold}$ is the hold time of the end point register.

When there is no combinational logic between the registers, the path delay is composed only by the input register propagation delay and the wire parasitic delay. The way to fix early violations in these paths regarding placement is to move the registers apart. Assuming cell delay as its drive strength and wire delay as Elmore, Equation 3.10 may be rewritten as shown in Equation 3.11.

$$d_{path}^{early} = R_i \left(xC_w + C_o\right) + xR_w \left(\frac{xC_w}{2} + C_o\right) \tag{3.11}$$

where $x$ is the optimum wirelength for the connection; $R_i$ is the start point register driver resistance and $C_o$ is the end point register input capacitance. Assuming that there is no latency change while moving the registers and the hold timing is also a constant, Equation 3.11 may be solved as shown below:

$$x = \frac{\sqrt{2C_wR_wd_{path}^{early} + {C_o}^2{R_w}^2 + {C_w}^2{R_i}^2} - C_oR_w - C_wR_i}{C_wR_w} \tag{3.12}$$

Once $x$ is computed, the input register is shifted away from the end point register following the straight line formed by the two registers.

**Average Bin Utilization (ABU) reduction**

A key challenge during timing-driven placement is to avoid congestion during optimization. This technique ranks noncritical cells according to their positive slack inside regions with a high density of cells. These regions are identified using the metric ABU commonly adopted by placement tool (for more information about ABU, please refer to Chapter 5). The cells are then moved to the nearest non-congested bin. After every move, the Steiner trees are rebuilt and timing updated to assure no timing violation was created. The movements are performed until all bins achieve a given target density.

# 4 INCREMENTAL QUADRATIC PLACEMENT TECHNIQUES FOR PATH SMOOTHING

For a better understanding, this chapter is divided into 3 sections. The former presents a survey on how quadratic placement may be applied incrementally by the addition of anchors to the linear system, in an operation called neutralization. Once the system is neutralized, additional forces are added between the pins of critical paths. The weight of these forces are modeled to address Elmore delay and drive strength, and hence to optimize the delays of these paths. This methodology is further explained in section 4.2. The latter shows the flow integrating the proposed techniques with our previous work (FLACH et al., 2016), composed of single-cell movements and previously discussed in chapter 3.

## 4.1 Incremental Quadratic Placement

A good global placement solution carries on several important properties as reduced wirelength, overlap, congestion and timing. An incremental placement takes such a solution and tries to further improve it while keeping the global properties of the initial solution.

In order to make use of an initial solution in a quadratic placement formulation, constant forces or anchors may be added to the linear system to hold movable elements in their current position (HU; MAREK-SADOWSKA, 2005). We call this procedure neutralization as the linear system is adjusted in such a way that its solution provides the current element positions and all forces acting on the system cancel each other. After the neutralization, the linear system is perturbed again so that a new solution is generated.

In this work the linear system is neutralized using anchors as they can be viewed as a generalization of constant forces and are typically more stable (HU; MAREK-SADOWSKA, 2005). Assume that the linear system presented in Equation (4.1) describes a quadratic placement formulation. No assumption on net modeling (e.g. clique, star) is made.

$$A\vec{x} = \vec{b} \tag{4.1}$$

where $A$ is the Hessian matrix; $\vec{x}$ represents the cell's current position and $\vec{b}$ represents the connections with fixed nodes.

A movable element is connected to a fixed element whenever the diagonal value respective to that element is greater than the sum of the absolute values of the off-diagonal elements. Therefore, to add an anchor to every cell, one needs to add a positive value to the diagonal of the Hessian matrix as shown in Equation (4.2) where $w_i > 0$ is the weight associated to the $i_{th}$ anchor. Multiple fixed points are automatically merged into one. So preexistent movable-to-fixed connections are seamlessly handled.

$$\dot{A} = A + diag(w_1, w_2, \cdots, w_n) \tag{4.2}$$

To finally neutralize the system, the anchor positions need to be defined. This is accomplished by setting the right-hand-side vector as in Equation (4.3) where $\vec{\mathbf{x_0}}$ is the current element positions.

$$\dot{\vec{b}} = \dot{A}\vec{x_0} \tag{4.3}$$

The neutralized system is shown in Equation (4.4). By construction, the system solution is $\vec{\mathbf{x}} = \vec{\mathbf{x_0}}$.

$$\dot{A}\vec{x} = \dot{\vec{b}} \tag{4.4}$$

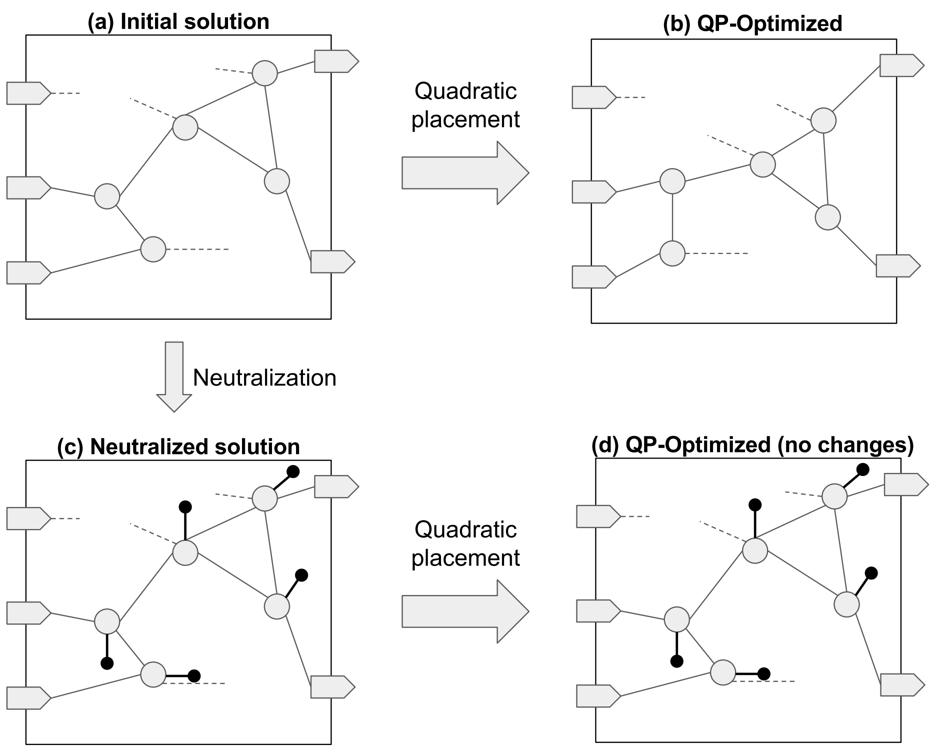
When the neutralization is done by the addition of anchors, their position $\dot{\mathbf{x}}_\mathbf{i}$, can be retrieved by applying the Equation (4.5).

$$\dot{x}_i = \frac{\dot{b}_i - b_i}{w_i} \tag{4.5}$$

Note that to neutralize the system using constant forces instead of anchors, simply set $w_i = 0$.

An example for a small circuit is depicted in Figure 4.1. The initial placement for the given circuit is presented in Figure 4.1(a). When the quadratic placement formulation is applied without neutralization, all the cells are moved to the position which the quadratic wirelength is minimum, with no respect to the initial solution (Figure 4.1(b)). In Figure 4.1(c), the neutralization process adds a new force connecting each cell to an anchor such that the quadratic forces are nullified. Therefore, if the quadratic placement is applied in the neutralized solution, the cells do not move, keeping the initial solution (Figure 4.1(d)).

Figure 4.1: A comparison between applying quadratic placement before and after the initial solution is neutralized.



Source: The author.

## 4.2 Net weighting techniques for timing-driven quadratic placement

Equation 4.6 is a second-degree function whose quadratic term is the length of interconnection, $L_{ij}$, multiplied by the resistance and capacitance per unit length of interconnections, $R_w$ and $C_w$, respectively. If we assume that each 2-pin connection between cells in quadratic placement is a wire and its weight is $g_{ij} = R_w C_w$, then minimize the quadratic placement function (Eq. 4.7) also should minimize the wire delay. Since $R_w C_w$ is a constant multiplying every term in the circuit, removing them from the formulation do not change the results.

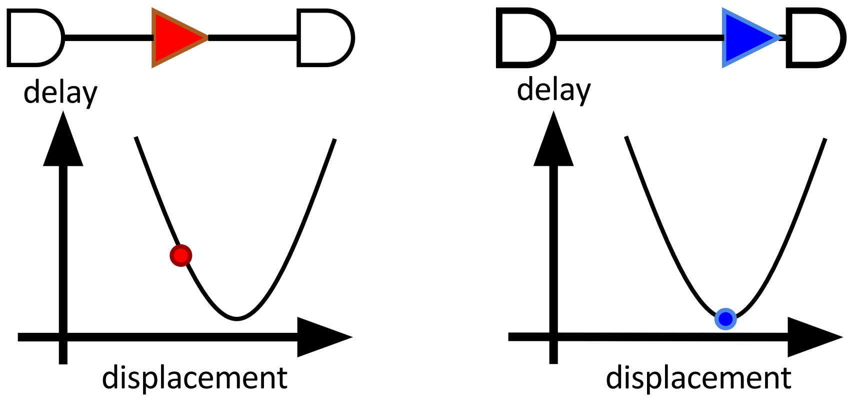
$$D_w(i,j) = R_w C_w (L_{ij})^2 + C_l R_w (L_{ij}) \tag{4.6}$$

$$W(\vec{x},\vec{y}) = \sum_{i}^{N}\sum_{j}^{N} g_{ij}(L_{ij})^2 \tag{4.7}$$

where $W(\vec{x},\vec{y})$ is the objective function; $g_{ij}$ is the weight for a interconnection and $L_{ij}$ is the interconnection length.

However, minimizing the wires alone does not produce good results regarding the delay. The local optimum solution for a buffer may not be the center position between its driver and sink cells, as shown in Fig. 4.2, unless the buffer has the same drive strength as its driver.

Figure 4.2: The optimum local position for a buffer may not be the center position between its driver and sink cells. The position is correlated to driver and buffer strength.



Source: Adapted from (FLACH et al., 2016)

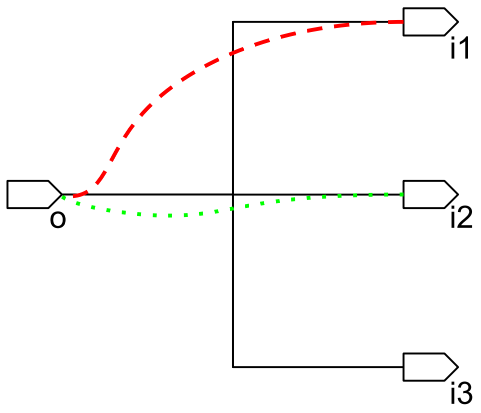
### 4.2.1 Driver-sink additional forces

This net weighting approach aims to perform critical path smoothing. The first step is to traverse all nets in the circuit. For each net it is verified if the driver is critical, i.e. there is at least one critical path passing through it. Consider the example illustrated in Fig 4.3 – There is a net with a critical driver $o$ and 3 sinks; 2 of them, $i1$ and $i2$ are also critical. To give priority for both paths passing through this net, two extra edges are added to the system, one between $o$ and $i1$ (dashed red) and another between $o$ and $i2$ (dotted green). The weight of those connections is also enhanced by a tunning factor $\alpha$, defined empirically, and the criticality of the sink. In this work, the criticality is a way to estimate the importance of a pin or a cell w.r.t the circuit WNS. The criticality of a pin is the negative slack of the pin divided by the worst negative slack found in the circuit while the criticality of a cell is equal to the maximum criticality in the pins that belong to the cell. Therefore, criticality is a real value in the range $[0, 1]$. The final weight of the new edges are:

$$g_{ij} = \alpha R_{drive}(1 + critically(j)) \tag{4.8}$$

where $R_{drive}$ is the driver resistance and $criticality(j)$ is the sink criticality.

Figure 4.3: Driver-sink additional forces for timing critical edges.



Source: The author.

This approach is depicted in Figure 4.4, using the same circuit from the example presented in Section 4.1. The initial placement (Figure 4.4a) is neutralized by the addition of connections with anchors (Figure 4.4b). In Figure 4.4b, a critical path connecting 3 cells and 2 IO pins is drawn in red. Extra edges connecting all the pairs driver-sink of the critical path are added. When performing the quadratic placement, the extra edges approximate the cells from the critical path while the anchors prevent the topological neighbor cells from moving away from the their initial position.

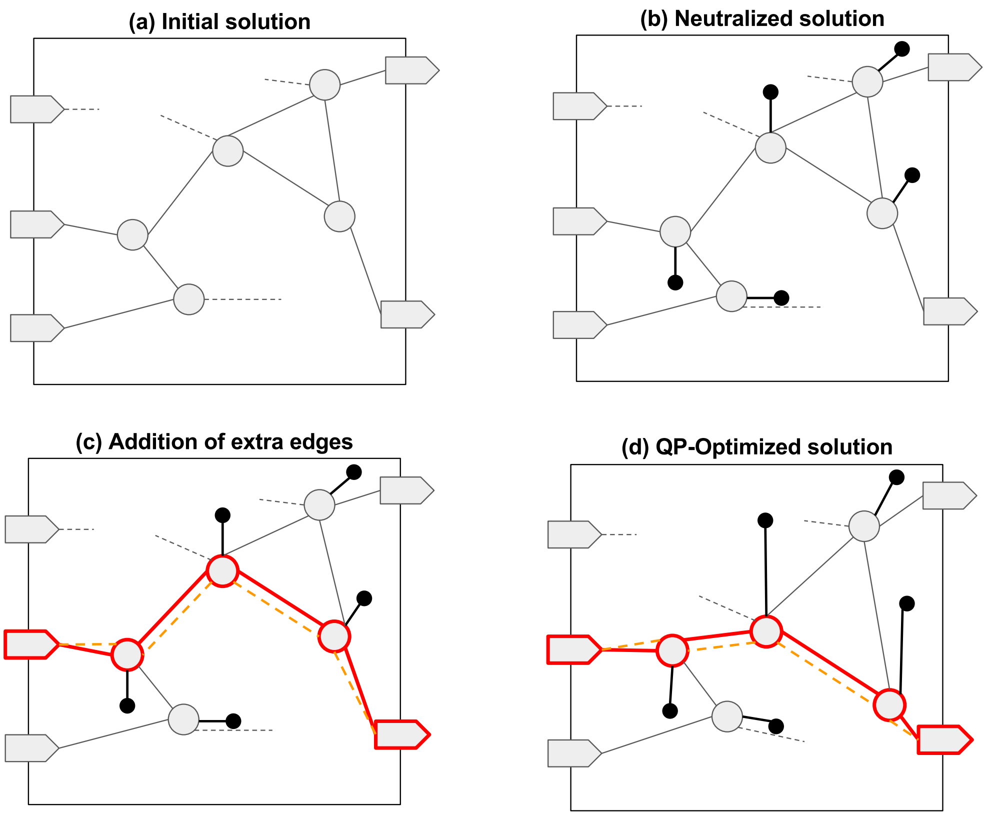
### 4.2.2 Drive strength aware clique net model

One may mitigate timing violations by properly balancing the load in the nets composing critical paths. To address load balancing, critical nets were decomposed in cliques as shown in Fig. 4.5. The weight of each edge is given by:

$$g_{ij} = \frac{\alpha R_{drive}(1 + criticality(i))}{(k-1)} \tag{4.9}$$

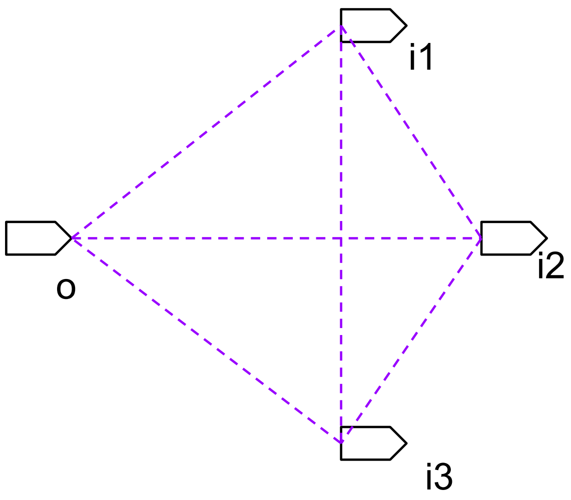
where $criticality(i)$ is the driver criticality and $k$ is the number of pins of the net.

Figure 4.4: An example of critical path (bold red) smoothing by the use of driver-sink additional forces (dashed orange).



Source: The author.

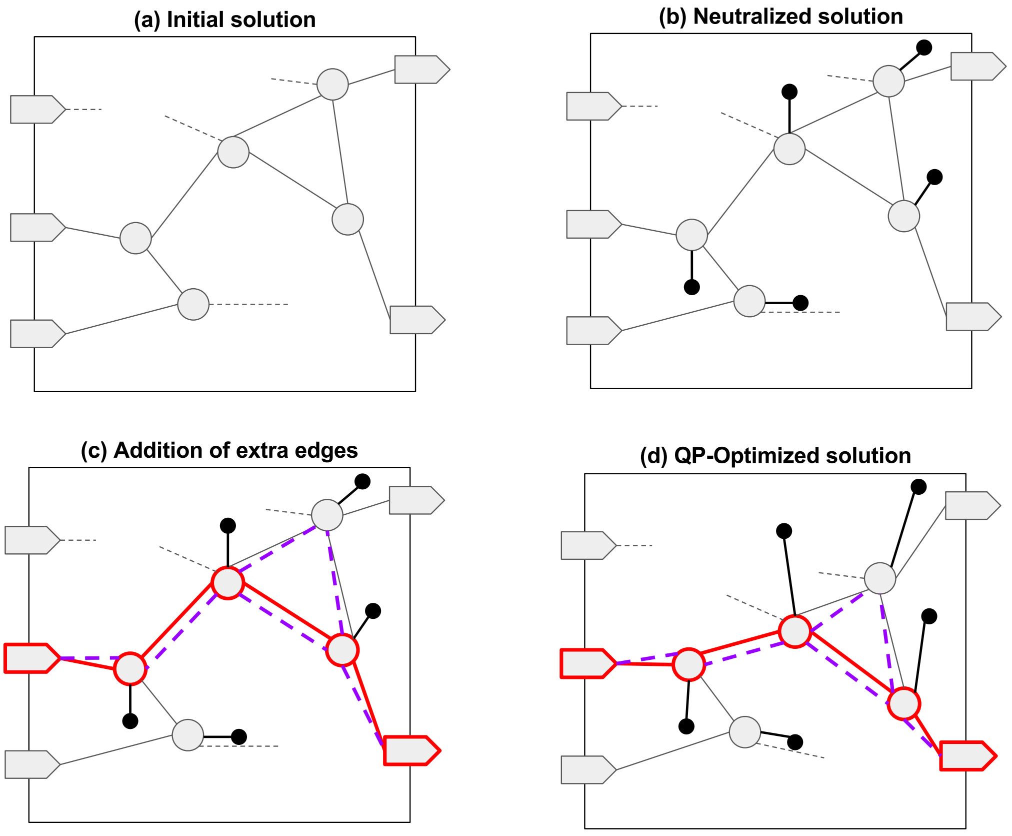
Figure 4.5: Modeling load optimization of critical nets into clique net model.



Source: The author.

An example of load balancing using drive strength aware clique is depiced in Figure 4.6. The initial solution from Figure 4.6(a) is neutralized by the addition of anchors in Figure 4.6(b). The critical path (bold red) is identified in Figure 4.6(c). Extra edges, represented in dashed purple, are created between all elements of nets. Figure 4.6(d) presents the final solution after applying the quadratic placement. The load balancing approach does align the critical path as well as the previous formulation, depicted in Figure 4.4, however, the load in critical nets is optmized due to the reduction in its wirelength.

Figure 4.6: An example of the load balancing forces (dashed purple) by the use of driver strength aware clique model.



Source: From author.

### 4.3 Proposed Incremental Timing-driven placement Flow

The proposed analytical formulations were integrated into the flow presented in Fig. 4.7. It relies on three major stages: (I) Top critical paths smoothing, (II) iterative net weighting and (III) the baseline flow. The baseline flow is a flow composed by local search techniques previously proposed by us (FLACH et al., 2016) and presented in Chapter 3. It is important to notice that the implementation of the baseline methods made in this work highly correlates with the original proposal in (FLACH et al., 2016) with results varying less than 1% on average.

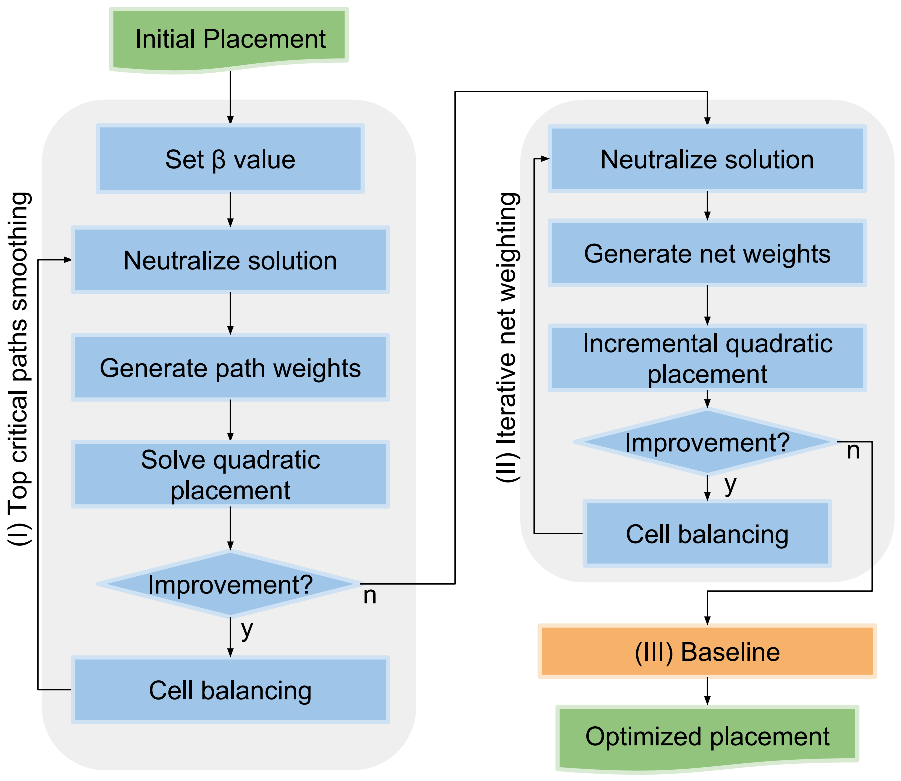
Top critical path smoothing consists of iterating through the circuit netlist finding the nets whose driver has a criticality higher than a threshold $\beta$. In these nets, the edges between the driver cell and their respective criticals sinks are strengthened following the methodology driver-sink presented in Section 4.2.1. In the experiments performed, the value of $\beta$ was set automatically for each benchmark as the criticality of the third most critical endpoint. This threshold estimation methodology proved to be efficient for tested benchmarks. The path smoothing formulation is repeated until the solution stops improving.

Iterative net weighting improves the current solution quality by assigning higher weights to the cliques proportionally to their driver criticality following the technique proposed in Section 4.2.2. At this stage, the goal is to reduce the load on the critical nets and to further align the critical paths. As pointed on other timing-driven analytical works, the net weighting technique may present oscillations before converging to the final solution (VISWANATHAN et al., 2010).

Every time a new solution of the quadratic placement is accepted in stages (I) and (II), it is applied one iteration of the cell balancing technique from our previous work. Experiments showed that combining this technique helped to improve the gains from the quadratic placement. The loop verifications are done before cell balancing execution to assure that the gains come from quadratic placement, avoiding local minima – This conjecture was also confirmed empirically.

The last stage is the baseline flow, composed by our 9 local search algorithms. For more details, please refer to Section 3.7.

Figure 4.7: Proposed timing-driven placement flow.



Source: The author.

# 5 EXPERIMENTAL SETUP

This chapter presents the validation of the proposed flow. The implementation was made using C++11 language and the tests performed on an Intel Core i7-4790K CPU @ 4.00GHz $\times$ 8 CPU with 32GB running Ubuntu 14.04 LTS (64-bit). The experiments were performed using the infrastructure of 2015 ICCAD CAD Contest in Incremental Timing-Driven Placement (KIM et al., 2015) which is presented in Section 5.1.

Section 5.2 presents the results based on the infrastructure metrics and the following sections present experiments for further understanding of the algorithm behavior. Section 5.3 presents an analysis of the slack histogram in different placement solutions while Section 5.4 presents the individual gains for each technique applied in the flow. Finally, Section 5.5 shows an experiment evaluating the path smoothing.

## 5.1 ICCAD 2015 CAD Contest in Incremental Timing-Driven Placement infrastructure

Although several works addressed timing-driven placement, it was difficult to measure the real contribution of each one since they were validated by different metrics and executed in different circuits. The ICCAD 2015 CAD Contest in Incremental Timing-Driven Placement proposed a full framework to motivate the research in this field and standard metrics do make easier to compare the results. It provides eight mixed-size benchmark circuits, ranging from 700k to 2M components and derived from industrial Application-Specific Integrated Circuit (ASIC) designs (KIM; HU, 2016). More information about the benchmarks is given in Table 5.1.

Table 5.1: ICCAD 2015 CAD contest benchmarks.

| Circuit | Number of nodes | | | Max disp. ($\mu$m) | | Target utilization | Target clock period (ns) |
|---|---|---|---|---|---|---|---|
| | Movable | Registers | Fixed | Short | Long | | |
| superblue1 | 1,209,716 | 144,266 | 56,898 | 40 | 500 | 0.8 | 9 |
| superblue3 | 1,213,253 | 167,923 | 58,970 | 40 | 400 | 0.87 | 10 |
| superblue4 | 795,645 | 176,895 | 45,289 | 50 | 500 | 0.9 | 6 |
| superblue5 | 1,086,888 | 114,103 | 76,676 | 30 | 400 | 0.85 | 9 |
| superblue7 | 1,931,639 | 270,219 | 72,256 | 50 | 400 | 0.9 | 5.5 |
| superblue10 | 1,876,103 | 241,267 | 101,837 | 20 | 400 | 0.87 | 10 |
| superblue16 | 981,559 | 142,543 | 4,868 | 20 | 400 | 0.85 | 5.5 |
| superblue18 | 768,068 | 103,544 | 27,099 | 30 | 400 | 0.85 | 7 |

Source: The author.

An initial placement optimized for wirelength and routability for each circuit is provided. One should minimize the timing violations w.r.t. the initial solution. The metric to evaluate these changes in the quality of the solution is based on early and late slack improvement and congestion. The timing evaluation is presented by Equation 5.1.

3

$$\begin{aligned} Timing_{Improv} = 10 \times \frac{lTNS_{final}}{lTNS_{initial}} + 2 \times \frac{eTNS_{final}}{eTNS_{initial}} + \\ 5 \times \frac{lWNS_{final}}{lWNS_{initial}} + \frac{eWNS_{final}}{eWNS_{initial}} \end{aligned} \tag{5.1}$$

where lWNS and lTNS are the worst and total negative slack for late mode respectively, while eWNS and eTNS are the worst and total negative slack for early mode respectively.

The congestion penalty is given by the Average Bin Utilization (ABU) (KIM et al., 2012), calculated as shown in Eq. 5.2.

$$Congestion_{penalty} = \frac{\sum^{\phi}(K_{\phi} \times U_{\phi})}{\sum^{\phi} K_{\phi}} \tag{5.2}$$

where $U$ is the summation of the cells area in a given region divided by the total available area in the same region; $U_{\phi}$ is the average value of U for the $\phi\%$ most congested regions; $\phi$ is in the set of values $\{2\%, 5\%, 10\%, 20\%\}$ and $K_2 = 10$, $K_5 = 4$, $K_{10} = 2$ and $K_{20} = 1$. The quality of the solution is computed as shown in Equation 5.3.

$$QS = max(Timing_{Improv} \times (1 - Congestion_{penalty}), 0) \tag{5.3}$$

A soft constraints is implied by the QS metric: Maintain the same congestion routing as well as pin density of the initial solution. However, some hard constraints are also suggested. The first one refers to the legality of the solution – The optimization techniques must assure the cells are aligned to sites in rows and do not have overlaps among them, i. e., it is a feasible solution. To restrict the impact on initial placement, two max displacement constraints cells are given for each benchmark, called *short* and *long* displacement. The short displacement usually has a value of tens of micra restricting techniques to perform only local movements while long displacement allows cells to move hundreds of micra giving more room for global optimization.

Finally, the infrastructure allows the algorithms to assign the registers to different clock buffers other than the initial ones, under three rules:

- Every register must be assigned to one source of clock signals;
- A register cannot be driven by more than one clock source;
- For each circuit, a maximum fanout for the local clock buffers (LCBs) is determined.

To provide an implementation-free comparison between different tools, an evaluation script is provided. It reports timing, wirelength and hard-constraints violation of a given placement solution. Note that a hard-constraint violation does not prevent a solution to be evaluated and hence may be ignored, at the cost of a fair comparison with other works.

## 5.2 Quality of solution evaluation

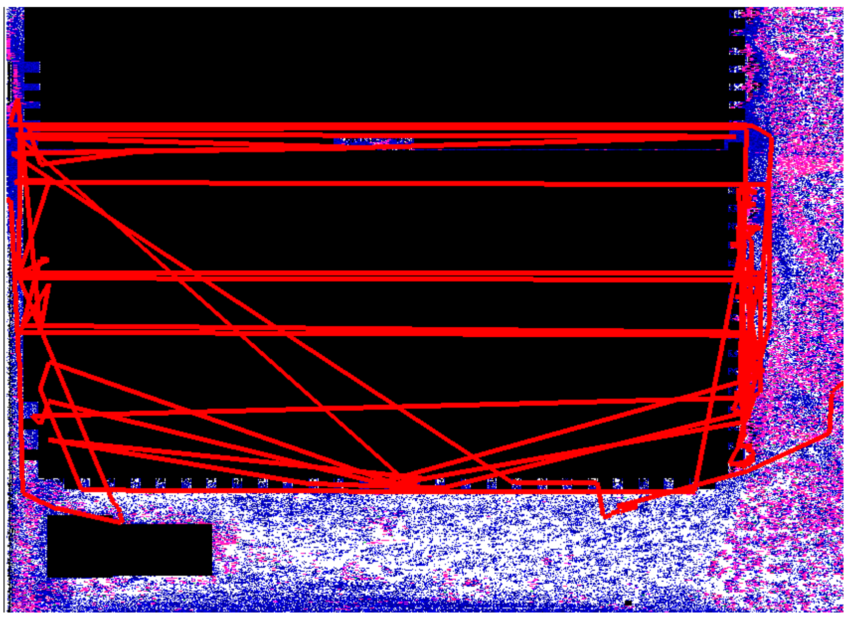
A comparison between the initial solution, 2015 ICCAD CAD Contest winner (KIM et al., 2015), the baseline flow and the proposed flow is presented in Table 5.2 regarding wirelength (StWL), congestion (ABU), timing (early and late), runtime and quality of solution. Figures 5.2 and 5.3. The proposed flow produces the best results, on average, regarding the late timing violations. For late WNS (lWNS), it achieves a reduction from 5.8% to 32.3% and for late TNS (lTNS) a reduction from 15.54% to 36.29% w.r.t. the initial solution. Note that none of the compared techniques presented gains for lWNS in superblue7. The reason is that the most critical paths in the circuit are composed just by primary inputs, primary outputs and fixed nodes and thereby this benchmark will be ignored in lWNS statistics.

The proposed flow also outperforms the contest winner in all benchmarks, achieving 21.1% smaller lWNS in superblue4 and 35.2% smaller lTNS in superblue16. The average improvement w.r.t. contest winner solutions were 12.7% for lWNS and 17.5% for lTNS. Note that the baseline flow (FLACH et al., 2016) was not able to outperform the contest winner in timing metrics for superblue5. However, when integrated with the proposed analytical techniques the resulting flow outperformed the contest winner in 13.9% and 8% for lWNS and lTNS, respectively.

When compared to the solutions of the baseline alone, the proposed flow improves the lWNS on average by 8.2% and lTNS by 7.6%. However, it was not able to outperform lTNS in superblue10 – Further investigation pointed that the gains on path smoothing for this benchmark were lost in legalization due to the high congestion around the macro blocks, as shown in Fig. 5.1, But the loss in lTNS is just 0.12%. The smaller gains in superblue18 regarding the TNS were also investigated. It was noticed that the quadratic placement moved groups of cell to suboptimal positions. The single-cell movements from the baseline flow are greedy, and therefore, they failed to further optimize the delay of the cells in these groups.

Although the proposed flow moves a great number of cells, the Steiner wirelength (StWL) is only 0.2% higher compared to the baseline flow and 2.4% higher than contest winner. The ABU metric was on average 2.8% higher than the baseline, but due to the ABU reduction algorithm, it still is on average 73% smaller than the initial solution and 75% smaller than contest winner. Runtime was increased on average 56.2% w.r.t the baseline, but its absolute value is not prohibitive – The worst case do not exceeds 10 minutes. The proposed flow also presented a slight increase in early total negative slack (eTNS). However, this kind of violations will not be further discussed, since they may be properly fixed (SRINIVASAN; CHAUDHARY; KUH, 1991) by several techniques, e.g. gate sizing and buffering.

Figure 5.1: The critical path of the circuit superblue10 painted in red, surrounding a macro block in black and standard-cells in blue/purple.



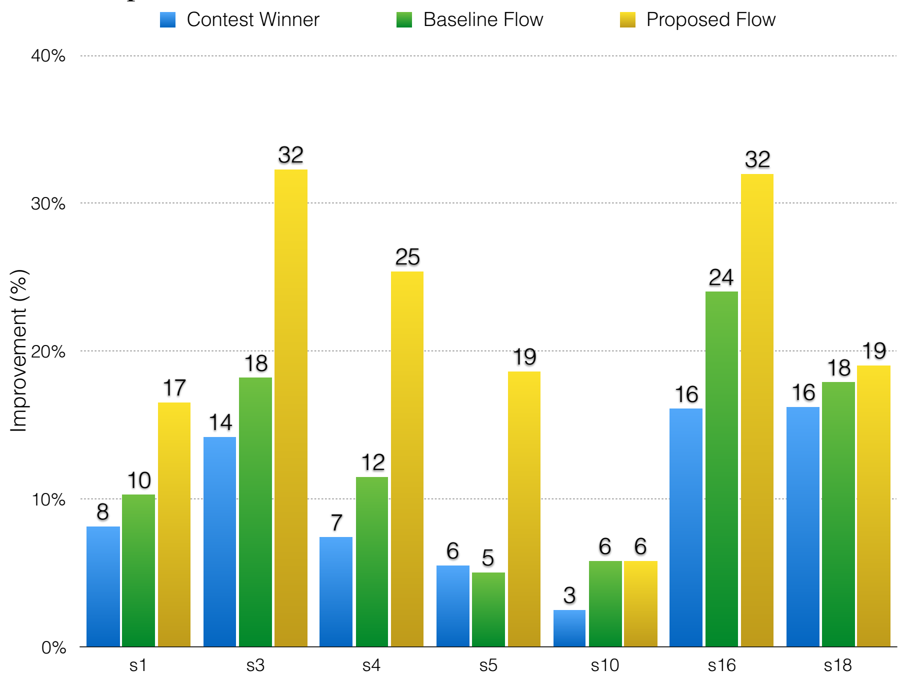
Source: The author.

Table 5.2: Experimental results of our incremental timing-driven placement flow on ICCAD 2015 contest benchmarks.

| Benchmark | Solution | ABU | StWL ($\mu$m) $\times 10^7$ | Early (ps) WNS $\times 10^0$ | Early (ps) TNS $\times 10^0$ | Late (ps) WNS $\times 10^3$ | Late (ps) TNS $\times 10^5$ | Run-time (min) | Quality of solution |
|---|---|---|---|---|---|---|---|---|---|
| superblue1 | Initial | 0.05 | 9.59 | -9.34 | -317.44 | -4.98 | -4.60 | - | - |
| | Contest winner | 0.06 | 9.61 | -16.65 | -80.89 | -4.57 | -3.51 | 3.20 | 346.64 |
| | Baseline | 0.01 | 9.88 | -9.25 | -36.70 | -4.46 | -3.40 | 2.33 | 512.57 |
| | Proposed flow | 0.01 | 9.91 | -9.25 | -36.70 | -4.21 | -3.26 | 5.89 | 568.64 |
| superblue3 | Initial | 0.03 | 11.43 | -78.36 | -1458.78 | -10.15 | -15.03 | - | - |
| | Contest winner | 0.03 | 11.46 | -13.13 | -214.03 | -8.71 | -11.60 | 2.70 | 551.74 |
| | Baseline | 0.01 | 11.59 | -10.72 | -91.01 | -8.30 | -9.68 | 2.66 | 735.72 |
| | Proposed flow | 0.01 | 11.61 | -10.72 | -91.01 | -6.87 | -8.10 | 9.78 | 915.24 |
| superblue4 | Initial | 0.04 | 7.15 | -12.55 | -519.39 | -6.22 | -34.77 | - | - |
| | Contest winner | 0.05 | 7.16 | -12.28 | -53.84 | -5.76 | -24.65 | 1.86 | 507.31 |
| | Baseline | 0.04 | 7.53 | 0.00 | 0.00 | -5.51 | -23.61 | 2.96 | 680.91 |
| | Proposed flow | 0.04 | 7.55 | 0.00 | 0.00 | -4.64 | -22.93 | 5.25 | 770.34 |
| superblue5 | Initial | 0.02 | 10.75 | -36.77 | -591.42 | -25.70 | -69.65 | - | - |
| | Contest winner | 0.02 | 10.78 | -36.77 | -618.27 | -24.29 | -58.42 | 2.53 | 179.53 |
| | Baseline | 0.00 | 10.92 | 0.00 | 0.00 | -24.42 | -59.75 | 1.97 | 476.82 |
| | Proposed flow | 0.00 | 11.02 | 0.00 | 0.00 | -20.92 | -53.71 | 6.65 | 634.89 |
| superblue7 | Initial | 0.03 | 14.01 | -7.65 | -1985.85 | -15.22 | -18.57 | - | - |
| | Contest winner | 0.03 | 14.03 | -6.75 | -1958.34 | -15.22 | -15.11 | 5.31 | 200.72 |
| | Baseline | 0.01 | 14.22 | -7.53 | -1976.58 | -15.22 | -13.61 | 3.13 | 275.64 |
| | Proposed flow | 0.01 | 14.24 | -7.53 | -1978.27 | -15.22 | -11.55 | 8.63 | 389.19 |
| superblue10 | Initial | 0.04 | 20.53 | -8.62 | -620.95 | -16.49 | -331.53 | - | - |
| | Contest winner | 0.04 | 20.55 | -5.15 | -373.75 | -16.08 | -315.18 | 3.74 | 181.33 |
| | Baseline | 0.01 | 21.08 | 0.00 | 0.00 | -15.54 | -279.71 | 4.96 | 499.38 |
| | Proposed flow | 0.01 | 21.08 | 0.00 | 0.00 | -15.53 | -280.03 | 7.35 | 498.80 |
| superblue16 | Initial | 0.03 | 9.33 | -10.65 | -113.75 | -4.58 | -7.76 | - | - |
| | Contest winner | 0.04 | 9.37 | -7.55 | -37.64 | -3.85 | -2.66 | 2.24 | 894.76 |
| | Baseline | 0.00 | 9.50 | 0.00 | 0.00 | -3.48 | -2.03 | 1.78 | 1196.97 |
| | Proposed flow | 0.00 | 9.54 | 0.00 | 0.00 | -3.11 | -1.72 | 5.15 | 1279.93 |
| superblue18 | Initial | 0.04 | 5.77 | -19.01 | -283.00 | -4.55 | -10.35 | - | - |
| | Contest winner | 0.05 | 5.78 | -1.95 | -6.86 | -3.82 | -7.76 | 1.59 | 613.07 |
| | Baseline | 0.01 | 5.90 | 0.00 | 0.00 | -3.74 | -6.19 | 1.50 | 815.37 |
| | Proposed flow | 0.01 | 5.90 | 0.00 | 0.00 | -3.69 | -6.38 | 2.82 | 801.98 |
| Avg Change (%) | Initial | 73.08 | -2.68 | 73.60 | 85.32 | 21.39 | 37.72 | - | - |
| | Contest winner | 75.05 | -2.47 | 68.92 | 76.39 | 12.78 | 17.53 | -133.34 | 97.56 |
| | Baseline | -2.85 | -0.25 | 0.00 | -0.03 | 9.47 | 7.60 | -156.26 | 16.18 |

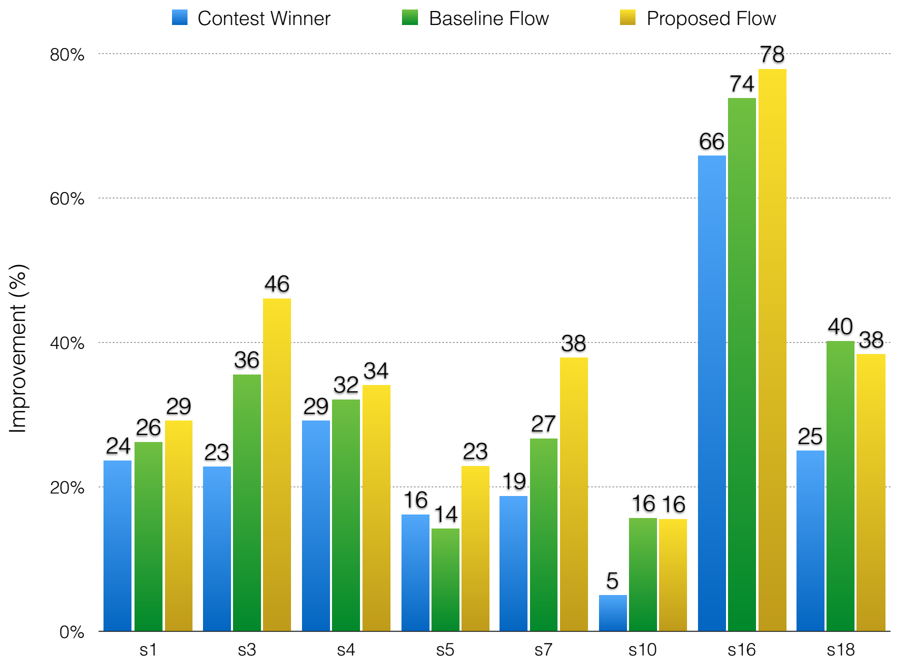
Source: The author.

Figure 5.2: A comparison between the contest winner, baseline and proposed flow in terms of lWNS improvement w.r.t. initial solution.



Source: The author.

Figure 5.3: A comparison between the contest winner, baseline and proposed flow in terms of lTNS improvement w.r.t. initial solution.



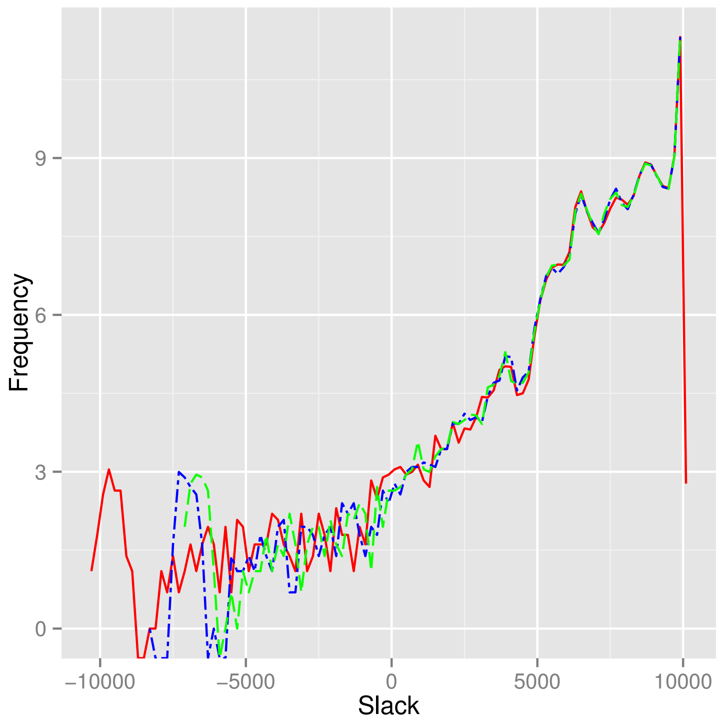
Source: The author.

## 5.3 Slack Histogram Analysis

One way to validate the timing improvement is through the analysis of a slack histogram. The slack histogram is built regarding the slack in all endpoints of the circuit, including the non-critical ones. In the experiments, the histograms were successfully compressed towards positive values by the baseline and further compressed by the proposed flow for all benchmarks. Figures 5.4-5.7 present the slack histogram for the initial solution in red, baseline in blue and the proposed flow in green.

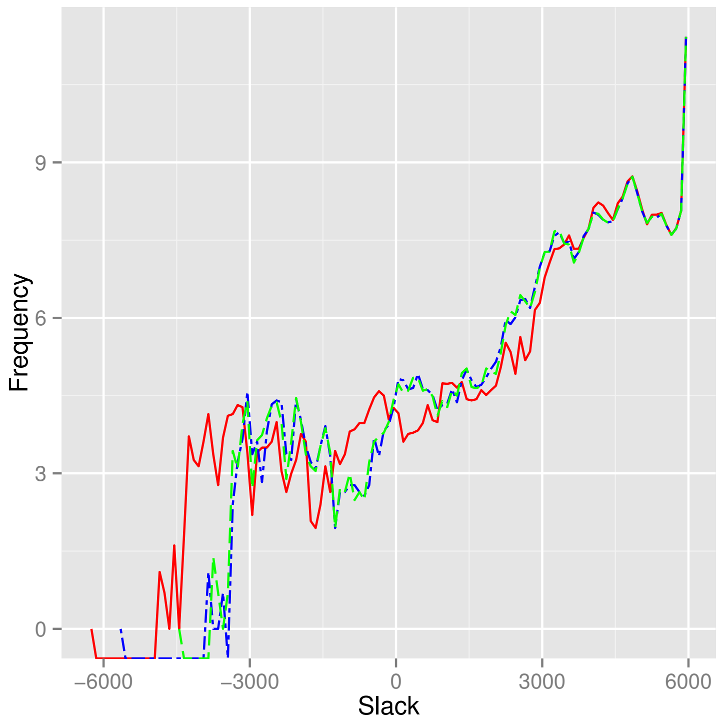
The slack on the worst endpoint for some circuits, such as superblues 4 and 5, is notably higher than the others. In these cases, the proposed flow successfully reduced the WNS by 25.4% and 18.6%, overcoming the gains of techniques in the baseline by 15.7% and 14.3%. One may notice that the slack histogram produced by the proposed flow for superblue18 presents more peaks, which may explain the increase in lTNS. Finally, superblue16 which has a well distributed slack histogram is clearly compressed by baseline alone, and further optimized by the proposed flow. The later one decreases the number of peaks in the histogram.

Figure 5.4: The slack histogram comparison for Superblue3. The red curve stands for the initial solution, the blue to the baseline flow and the green to the proposed flow.



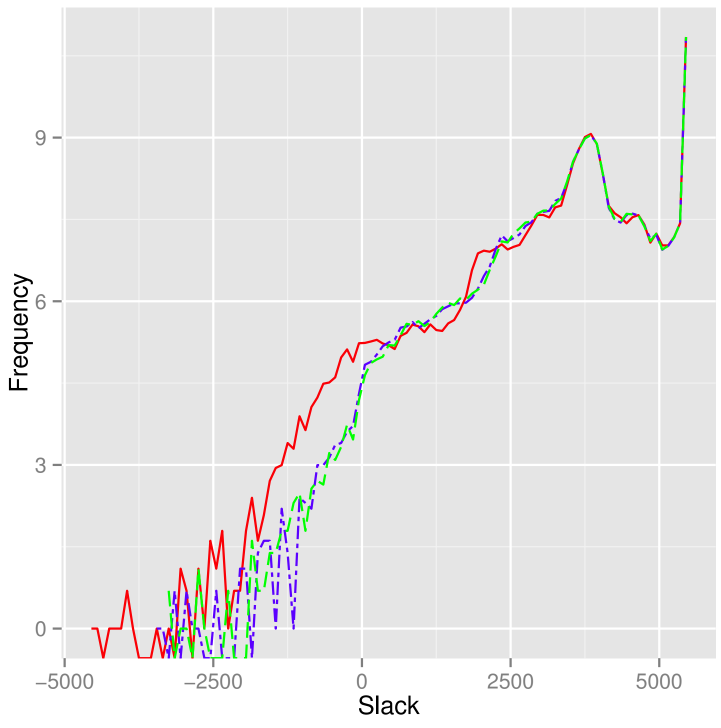
Source: The author.

Figure 5.5: The slack histogram comparison for Superblue4. The red curve stands for the initial solution, the blue to the baseline flow and the green to the proposed flow.



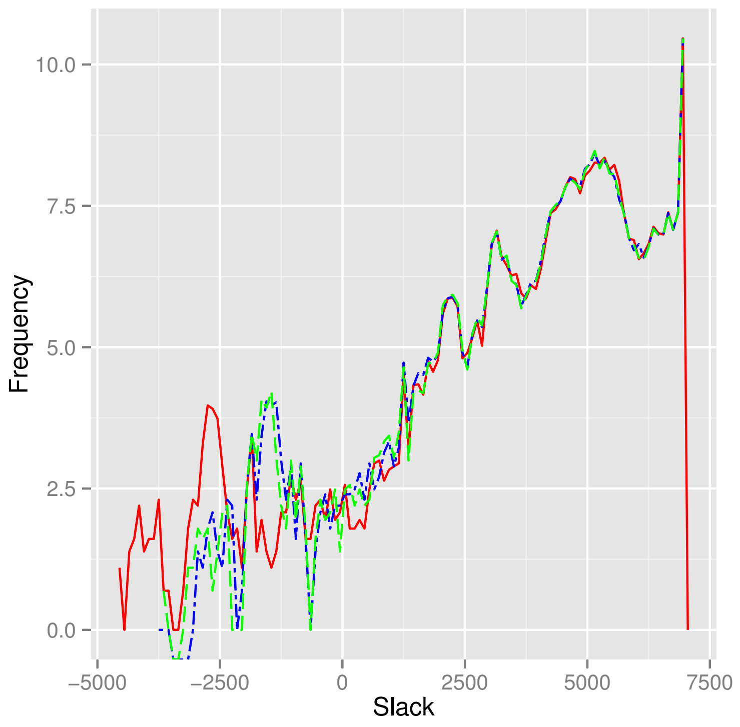
Source: The author.

Figure 5.6: The slack histogram comparison for Superblue16. The red curve stands for the initial solution, the blue to the baseline flow and the green to the proposed flow.



Source: The author.

Figure 5.7: The slack histogram comparison for Superblue18.



Source: The author.

## 5.4 Individual gains analysis

To measure the efficiency of the proposed quadratic formulations and compare with the baseline flow, the gains in the solution were summed and stored for each benchmark after applying the techniques individually. Table 5.3 summarizes the average share in the total gain for each technique.

The proposed path smoothing approach presents the major gains for lWNS and lTNS while the net weighting technique presents more modest but still relevant gains. Tests showed that, for some benchmarks, the net weighting technique was determining to achieve the quality of results. The presented gains obtained with these techniques came at the cost of a loss of quality in ABU and StWL.

Although early optimization is not the focus of this work, the gains of the techniques that mitigate early violations were presented, since they may affect other metrics. In these experiments, *Register Swap* and *Register-to-Register* did not present significant changes in any metric. Skew Optimization and Iterative Spreading presented slightly improvements in ABU and degraded StWL. Skew Optimization also slightly degraded lWNS.

Table 5.3: Average improvement obtained by each step after the flow execution for ABU, StWL, lWNS and lTNS Positive values mean improvement.

| Step | Average Improvement (%) | | | |
|---|---|---|---|---|
| | ABU | StWL | lWNS | lTNS |
| QP - Path Smoothing | -8.56 | -6.48 | 52.92 | 38.81 |
| QP - Net Weighting | -5.84 | -6.90 | 10.50 | 10.85 |
| Skew Optimization | 0.09 | -0.46 | N/A | N/A |
| Iterative Spreading | 0.05 | -0.28 | N/A | N/A |
| Register Swap | 0.00 | 0.00 | N/A | N/A |
| Reg-to-Reg Path Fix | 0.00 | 0.00 | N/A | N/A |
| Clustered Move | -0.24 | -0.12 | 7.24 | 0.57 |
| Buffer Balancing | 0.03 | -0.03 | 10.23 | 7.08 |
| Cell Balancing | 0.47 | -0.22 | 10.81 | 8.95 |
| Load Optmization | -25.49 | -57.87 | 8.35 | 21.27 |
| ABU Reduction | 139.47 | -27.64 | -0.03 | -0.01 |

Source: The author.

The late optimization techniques presented similar gains for lWNS. The *Load Optimization* was crucial for TNS, while *Clustered movement* did not show significative gains. The techniques from the baseline flow did not present significant changes in ABU and StWL with the one exception of the *Load Optimization* which significantly degraded both metrics. Finally, *ABU Reduction* successfully reduced congestion in the benchmarks, but at the cost of wirelength while the impact in lWNS and lTNS was negligible.

## 5.5 Critical path smoothing

Since the premise of this work is to optimize timing by means of critical path smoothing, an experiment measuring the paths length before and after the execution of the proposed flow was made. As discussed in Section 5.1, the contest infrastructure presents a heterogeneous benchmark suite, with circuits varying from 700k up to 2M movable elements. Furthermore, in Section 5.3 it was shown that the slack histogram of these circuits presents different characteristics from each other. To obtain a fair comparison between them, the following methodology was applied – For each circuit, it was identified the most critical path of the 100 most critical endpoints on the initial solution. The next step was to apply the proposed flow and finally verify the changes in these paths length. In the experiments, the paths length were calculated using the Manhattan distance since it correlates more with the Steiner three wirelength used by the static timer.

The results are summarized in Table 5.4. The first two columns show the summation of all the paths length before and after execution of the flow, respectively; Column "Total" presents the relative improvement regarding the total length while columns "Average", "Max" and "Min" highlight the average, maximum and minimum value of improvement in the paths length individually. When comparing timing and length results for the circuit superblue5 one may conclude that is possible to improve timing while worsening the total paths length. This conjecture makes sense, because while optimizing the most critical path, other critical or near-critical paths related to the former may degrade.

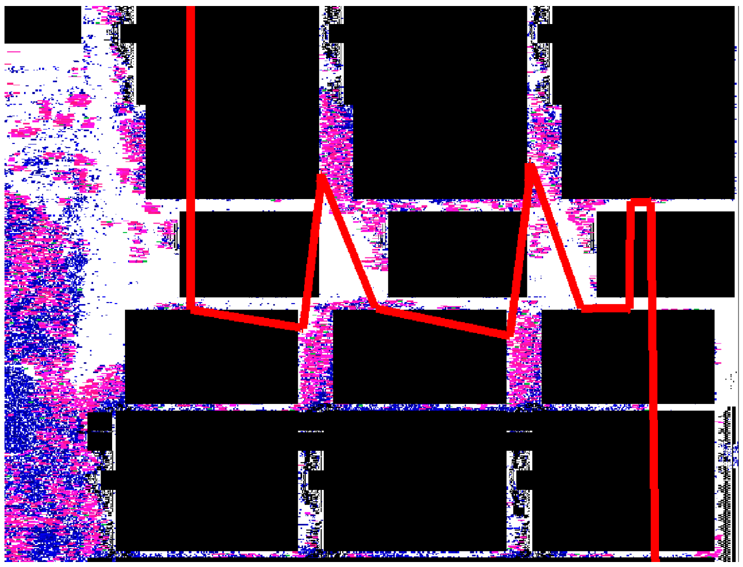
However, the values regarding the maximum individual improvement highly correlated with the timing results. The proposed flow failed to show significative improvement in both timing and length for superblue10 and superblue18 while produced good results for superblue5, superblue7 and superblue16. Figure 5.8 depicts the most critical path of superblue5 being smoothed.

Table 5.4: Evaluation of critical path smoothing by the proposed flow.

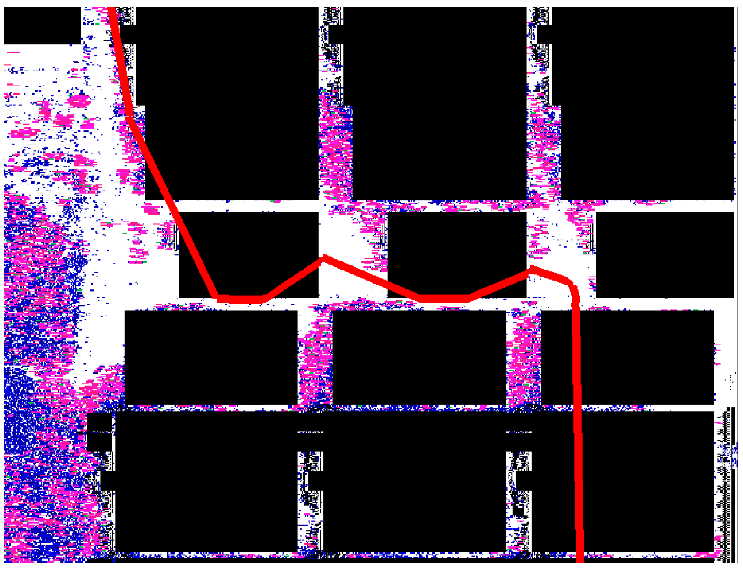
| Benchmark | Wirelength (m) | | Improvement (%) | | | |
|---|---|---|---|---|---|---|
| | Initial | Final | Total | Average | Max | Min |
| superblue1 | 1.24 | 1.19 | 3.49 | 3.42 | 11.41 | -0.3 |
| superblue3 | 1.59 | 1.59 | 0.38 | 0.12 | 10.41 | -1.13 |
| superblue4 | 1.21 | 1.19 | 2.29 | 1.86 | 11.71 | -0.39 |
| superblue5 | 0.7 | 0.73 | -4.29 | -21.71 | 15.14 | -42.04 |
| superblue7 | 1.31 | 1.25 | 4.75 | 4.48 | 15.2 | -0.35 |
| superblue10 | 4.6 | 4.57 | 0.59 | 0.59 | 1.29 | 0.09 |
| superblue16 | 1.31 | 1.25 | 4.75 | 4.48 | 15.2 | -0.35 |
| superblue18 | 1.21 | 1.21 | -0.57 | -0.39 | 1.49 | -2.1 |

Source: The author.

Figure 5.8: The critical path of circuit Superblue5, drawn in red, before (a) and after (b) the proposed flow. Standard cells are drawn in blue/purple and macro-blocks in black.



(a) Initial



(b) Optimized

Source: The author.

# 6 CONCLUSIONS

Achieving timing closure for modern nanoCMOS circuits demands huge effort from designers and the use of high quality EDA tools. During the late stages of the design, key steps like placement and routing should not only produce feasible solutions, but also be able to optimize performance with no harm to other aspects like area and power. This work focused on incremental timing-driven placement algorithms.

Quadratic techniques have been broadly applied for placement by academic (VISWANATHAN; CHU, 2005; KIM; LEE; MARKOV, 2013) and commercial tools (VISWANATHAN et al., 2010). However, most of these approaches concerning performance present a huge impact on previous solutions, which is not desirable in late stages stages of the design. To keep information from former solutions, an operation called neutralization was proposed and applied to quadratic placement. It modifies the quadratic system so that the zero-energy state is the current placement. Therefore, the cells only if new forces are added to the system.

Two strategies to optimize timing are proposed with different objectives. The first aims to reduce wires smoothing critical paths, and therefore mitigate violations, while the later focus on balancing the load of critical nets. Both techniques are implemented by the addition of extra edges in the quadratic formulation. They differ in the process that generates the forces. While path smoothing adds new forces between elements composing critical paths, the net weighting creates forces binding all nodes of critical nets.

The quadratic formulation was integrated into the previous proposed single-cell movement techniques (FLACH et al., 2016) to provide a full placement flow and it was evaluated using the infrastructure of ICCAD 2015 CAD Contest on incremental timing-driven placement. Results show that the use of the quadratic formulation with the global view of the problem produce good results alone, outperforming both the ICCAD winner and the previous flow (FLACH et al., 2016) by more than 10% in WNS. The flow integrating the quadratic techniques with the single-cell movements (FLACH et al., 2016) outperforms the known state-of-art by up to 17.3% in WNS and up to 16.3% in TNS for superblue3. Considering the ICCAD 2015 metric for measuring quality of result, this work outperforms, on average, 16.1% the known state-of-art.

Three experiments were performed to investigate the proposed flow behavior. The first one consisted in comparing the slack histogram of the initial solution, after the execution of the baseline flow (FLACH et al., 2016) and after the execution of the proposed flow. The latter successfully compressed the slack histogram toward positive values, which is the desired behavior of a timing-driven technique.

In the second experiment, it was computed how much each step of the proposed flow contributed to the final results. The proposed path smoothing technique was responsible for 52.9% of the gains regarding WNS and 38.8% of the gains regarding TNS while the net weighting technique contributed to 10% of the gains regarding both WNS and TNS.

Finally, the length of the most critical paths was estimated before and after executing the proposed flow. It was observed that it is possible to optimize timing even when the average path length was increased. On the other hand, the benchmarks with good results regarding timing also presented at least one critical path whose length was reduced by 10% or more.

## 6.1 Future works

Other techniques, like buffering, gate sizing and $V_{th}$ selection, can be easily integrated within the proposed flow to further optimize the solutions to address problems that placement cannot solve alone, like cells driving large fan-outs. Besides, integrating these techniques may help us to measure the real space for optimization existing in the benchmarks.

Incremental flows from industrial tools also perform placement combined with netlist restructuring. The implementation of these techniques will allow a fair comparison with these tools. It is also intended to expand the infrastructure to support more realistic features, e.g., hierarchical netlist and multiple clock domains.

# REFERENCES

AGNIHOTRI, A. R.; ONO, S.; MADDEN, P. H. Recursive bisection placement: Feng shui 5.0 implementation details. In: **Proceedings of the 2005 International Symposium on Physical Design**. New York, NY, USA: ACM, 2005. (ISPD '05), p. 230–232. ISBN 1-59593-021-3. Disponível em: <http://doi.acm.org/10.1145/1055137.1055186>.

ALPERT, C. et al. Placement: Hot or not? In: **Proceedings of the International Conference on Computer-Aided Design**. New York, NY, USA: ACM, 2012. (ICCAD '12), p. 283–290. ISBN 978-1-4503-1573-9. Disponível em: <http://doi.acm.org/10.1145/2429384.2429442>.

ALPERT, C. J. et al. Quadratic placement revisited. In: **Proceedings of the 34th Annual Design Automation Conference**. New York, NY, USA: ACM, 1997. (DAC '97), p. 752–757. ISBN 0-89791-920-3. Disponível em: <http://doi.acm.org/10.1145/266021.266362>.

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

BAKOGLU, H. B. **Circuits, interconnections, and packaging for VLSI**. [S.l.]: Reading, Mass: Addison-Wesley Pub. Co., 1990.

BHASKER, J.; CHADHA, R. **Static Timing Analysis for Nanometer Designs: A Practical Approach**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 0387938192, 9780387938196.

BOCK, A. et al. Local search algorithms for timing-driven placement under arbitrary delay models. In: **Proceedings of the 52Nd Annual Design Automation Conference**. New York, NY, USA: ACM, 2015. (DAC '15), p. 29:1–29:6. ISBN 978-1-4503-3520-1. Disponível em: <http://doi.acm.org/10.1145/2744769.2744867>.

BREUER, M. A. A class of min-cut placement algorithms. In: **Proceedings of the 14th Design Automation Conference**. Piscataway, NJ, USA: IEEE Press, 1977. (DAC '77), p. 284–290. Disponível em: <http://dl.acm.org/citation.cfm?id=800262.809144>.

BURSTEIN, M.; YOUSSEF, M. N. Timing influenced layout design. In: **Proceedings of the 22Nd ACM/IEEE Design Automation Conference**. Piscataway, NJ, USA: IEEE Press, 1985. (DAC '85), p. 124–130. ISBN 0-8186-0635-5. Disponível em: <http://dl.acm.org/citation.cfm?id=317825.317845>.

CALDWELL, A. E. et al. On wirelength estimations for row-based placement. In: **Proceedings of the 1998 International Symposium on Physical Design**. New York, NY, USA: ACM, 1998. (ISPD '98), p. 4–11. ISBN 1-58113-021-X. Disponível em: <http://doi.acm.org/10.1145/274535.274536>.

CALDWELL, A. E.; KAHNG, A. B.; MARKOV, I. L. Optimal partitioners and end-case placers for standard-cell layout. In: **Proceedings of the 1999 International Symposium on Physical Design**. New York, NY, USA: ACM, 1999. (ISPD '99), p. 90–96. ISBN 1-58113-089-9. Disponível em: <http://doi.acm.org/10.1145/299996.300032>.

CHEN, T.-C. et al. Ntuplace3: An analytical placer for large-scale mixed-size designs. In: NAM, G.-J.; CONG, J. (Ed.). **Modern Circuit Placement: Best Practices and Results**. Boston, MA: Springer US, 2007. p. 289–309. ISBN 978-0-387-68739-1. Disponível em: <http://dx.doi.org/10.1007/978-0-387-68739-1_11>.

CHU, C. Flute: fast lookup table based wirelength estimation technique. In: **Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on**. [S.l.: s.n.], 2004. p. 696–701. ISSN 1092-3152.

DOLL, K.; JOHANNES, F. M.; ANTREICH, K. J. Iterative placement improvement by network flow methods. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 13, n. 10, p. 1189–1200, Oct 1994. ISSN 0278-0070.

ELMORE, W. C. The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. **Journal of Applied Physics**, v. 19, n. 1, p. 55–63, 1948.

FLACH, G. **Clock Mesh Optimization**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2010.

FLACH, G. **Discrete Gate Sizing and Timing-Driven Detailed Placement for the Design of Digital Circuits**. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, 2015.

FLACH, G. et al. Drive strength aware cell movement techniques for timing driven placement. In: **Proceedings of the 2016 on International Symposium on Physical Design**. New York, NY, USA: ACM, 2016. (ISPD '16), p. 73–80. ISBN 978-1-4503-4039-7.

FLACH, G. et al. An incremental timing-driven flow using quadratic formulation for detailed placement. In: **2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)**. [S.l.: s.n.], 2015. p. 1–6. ISSN 2324-8432.

FLACH, G. et al. Effective Method for Simultaneous Gate Sizing and V th Assignment Using Lagrangian Relaxation. **TCAD IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 33, n. 4, p. 546–557, April 2014.

GARGINI, P. The international technology roadmap for semiconductors (itrs): "past, present and future". In: **GaAs IC Symposium, 2000. 22nd Annual**. [S.l.: s.n.], 2000. p. 3–5. ISSN 1064-7775.

HADDOCK, J.; MITTENTHAL, J. Simulation optimization using simulated annealing. **Comput. Ind. Eng.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 22, n. 4, p. 387–395, out. 1992. ISSN 0360-8352. Disponível em: <http://dx.doi.org/10.1016/0360-8352(92)90014-B>.

HALL, K. M. An r-Dimensional Quadratic Placement Algorithm. **Management Science**, v. 17, n. 3, p. 219–229, November 1970.

HANI, M. K.; SHAIKH-HUSIN, N. Simultaneous Routing and Buffer Insertion algorithm for interconnect delay optimization in VLSI layout design. In: **2008 International Conference on Microelectronics**. [S.l.: s.n.], 2008. p. 175–178.

HENTSCHKE, R. **Algorithms for Wire Length Improvement of VLSI Circuits With Concern to Critical Paths**. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, 2007.

HILL, D. **Method and system for high speed detailed placement of cells within an integrated circuit design**. 2002. US Patent 6,370,673.

HU, B.; MAREK-SADOWSKA, M. Multilevel fixed-point-addition-based vlsi placement. **IEEE Trans. on CAD of Integrated Circuits and Systems**, v. 24, n. 8, p. 1188–1203, 2005. Disponível em: <http://dblp.uni-trier.de/db/journals/tcad/tcad24.html#HuM05a>.

JOHANN, M. d. O. **Novos algoritmos para roteamento de circuitos VLSI**. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, 2001.

KAHNG, A. B.; WANG, Q. An analytic placer for mixed-size placement and timing-driven placement. In: **Proceedings of the 2004 IEEE/ACM International Conference on Computer-aided Design**. Washington, DC, USA: IEEE Computer Society, 2004. (ICCAD '04), p. 565–572. ISBN 0-7803-8702-3. Disponível em: <http://dx.doi.org/10.1109/ICCAD.2004.1382641>.

KIM, M.-C.; HU, J. **ICCAD-2015 CAD Contest in Incremental Timing-driven Placement and Benchmark Suite**. 2016. Disponível em: <http://cad-contest.el.cycu.edu.tw/problem_c/default.html>.

KIM, M.-C. et al. ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite. In: **Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on**. [S.l.: s.n.], 2015. p. 921–926.

KIM, M.-C.; LEE, D.-J.; MARKOV, I. L. Simpl: An algorithm for placing vlsi circuits. **Commun. ACM**, ACM, New York, NY, USA, v. 56, n. 6, p. 105–113, jun. 2013. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2461256.2461279>.

KIM, M.-C. et al. Maple: Multilevel adaptive placement for mixed-size designs. In: **Proceedings of the 2012 ACM International Symposium on International Symposium on Physical Design**. New York, NY, USA: ACM, 2012. (ISPD '12), p. 193–200. ISBN 978-1-4503-1167-0. Disponível em: <http://doi.acm.org/10.1145/2160916.2160958>.

KLEINHANS, J. M. et al. GORDIAN: VLSI placement by quadratic programming and slicing optimization. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 10, n. 3, p. 356–365, Mar 1991. ISSN 0278-0070.

KONG, T. T. A novel net weighting algorithm for timing-driven placement. In: **Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design**. New York, NY, USA: ACM, 2002. (ICCAD '02), p. 172–176. ISBN 0-7803-7607-2. Disponível em: <http://doi.acm.org/10.1145/774572.774597>.

KUHN, H. W. The hungarian method for the assignment problem. **Naval Research Logistics Quarterly**, Wiley Subscription Services, Inc., A Wiley Company, v. 2, n. 1-2, p. 83–97, 1955. ISSN 1931-9193. Disponível em: <http://dx.doi.org/10.1002/nav.3800020109>.

LIN, T. et al. POLAR: Placement based on novel rough legalization and refinement. In: **2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.: s.n.], 2013. p. 357–362. ISSN 1092-3152.

LU, J. et al. eplace-3d: Electrostatics based placement for 3d-ics. In: **Proceedings of the 2016 on International Symposium on Physical Design**. New York, NY, USA: ACM, 2016. (ISPD '16), p. 11–18. ISBN 978-1-4503-4039-7. Disponível em: <http://doi.acm.org/10.1145/2872334.2872361>.

LUO, T.; NEWMARK, D.; PAN, D. Z. A new LP based incremental timing driven placement for high performance designs. In: **2006 43rd ACM/IEEE Design Automation Conference**. [S.l.: s.n.], 2006. p. 1115–1120. ISSN 0738-100X.

MONTEIRO, J. et al. An analytical timing-driven algorithm for detailed placement. In: **Circuits Systems (LASCAS), 2015 IEEE 6th Latin American Symposium on**. [S.l.: s.n.], 2015. p. 1–4.

NAYLOR, W.; DONELLY, R.; SHA, L. **Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer**. Google Patents, 2001. US Patent 6,301,693. Disponível em: <http://www.google.com/patents/US6301693>.

OPENSOURCE Liberty. 2016. Disponível em: <https://www.opensourceliberty.org/>.

PAN, M.; VISWANATHAN, N.; CHU, C. An efficient and effective detailed placement algorithm. In: **ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.** [S.l.: s.n.], 2005. p. 48–55. ISSN 1092-3152.

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

POPOVYCH, S. et al. Density-aware detailed placement with instant legalization. In: **Proceedings of the 51st Annual Design Automation Conference**. New York, NY, USA: ACM, 2014. (DAC '14), p. 122:1–122:6. ISBN 978-1-4503-2730-5. Disponível em: <http://doi.acm.org/10.1145/2593069.2593120>.

PUGET, J. C. et al. Jezz: An effective legalization algorithm for minimum displacement. In: **Proceedings of the 28th Symposium on Integrated Circuits and Systems Design**. New York, NY, USA: ACM, 2015. (SBCCI '15), p. 19:1–19:5. ISBN 978-1-4503-3763-2. Disponível em: <http://doi.acm.org/10.1145/2800986.2801013>.

REIMANN, T. J. **Roteamento global de circuitos VLSI**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2013.

REIS, R. A. d. L. et al. **Concepção de circuitos integrados**. [S.l.]: Instituto de Informática da UFRGS. Editora Sagra Luzzatto, 2000.

RIESS, B. M.; ETTELT, G. G. SPEED: fast and efficient timing driven placement. In: **Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on**. [S.l.: s.n.], 1995. v. 1, p. 377–380 vol.1.

ROSSUM, M. V. MOS Device and Interconnects Scaling Physics. In: SHACHAM-DIAMAND, Y. et al. (Ed.). **Advanced Nanoscale ULSI Interconnects: Fundamentals and Applications**. Springer New York, 2009, (SpringerLink: Springer e-Books). cap. 2. ISBN 9780387958682. Disponível em: <https://books.google.com.br/books?id=OHpNQTt2-SwC>.

ROY, J. A. et al. Capo: Robust and scalable open-source min-cut floorplacer. In: **Proceedings of the 2005 International Symposium on Physical Design**. New York, NY, USA: ACM, 2005. (ISPD '05), p. 224–226. ISBN 1-59593-021-3. Disponível em: <http://doi.acm.org/10.1145/1055137.1055184>.

SECHEN, C.; SANGIOVANNI-VINCENTELLI, A. Timberwolf3.2: A new standard cell placement and global routing package. In: **Proceedings of the 23rd ACM/IEEE Design Automation Conference**. Piscataway, NJ, USA: IEEE Press, 1986. (DAC '86), p. 432–439. ISBN 0-8186-0702-5. Disponível em: <http://dl.acm.org/citation.cfm?id=318013.318083>.

SPINDLER, P.; JOHANNES, F. M. Kraftwerk: A fast and robust quadratic placer using an exact linear net model. In: NAM, G.-J.; CONG, J. (Ed.). **Modern Circuit Placement: Best Practices and Results**. Boston, MA: Springer US, 2007. p. 59–93. ISBN 978-0-387-68739-1. Disponível em: <http://dx.doi.org/10.1007/978-0-387-68739-1_4>.

SPINDLER, P.; SCHLICHTMANN, U.; JOHANNES, F. M. Abacus: Fast legalization of standard cell circuits with minimal movement. In: **Proceedings of the 2008 International Symposium on Physical Design**. New York, NY, USA: ACM, 2008. (ISPD '08), p. 47–53. ISBN 978-1-60558-048-7. Disponível em: <http://doi.acm.org/10.1145/1353629.1353640>.

SRINIVASAN, A.; CHAUDHARY, K.; KUH, E. S. RITUAL: a performance driven placement algorithm for small cell ICs. In: **Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE International Conference on**. [S.l.: s.n.], 1991. p. 48–51.

TSAY, R.-S.; KOEHL, J. An analytic net weighting approach for performance optimization in circuit placement. In: **Proceedings of the 28th ACM/IEEE Design Automation Conference**. New York, NY, USA: ACM, 1991. (DAC '91), p. 620–625. ISBN 0-89791-395-7. Disponível em: <http://doi.acm.org/10.1145/127601.122882>.

VISWANATHAN, N.; CHU, C. C. N. FastPlace: efficient analytical placement using cell shifting, iterative local refinement,and a hybrid net model. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 24, n. 5, p. 722–733, May 2005. ISSN 0278-0070.

VISWANATHAN, N. et al. ITOP: Integrating Timing Optimization Within Placement. In: **Proceedings of the 19th International Symposium on Physical Design**. [S.l.]: ACM, 2010. (ISPD '10), p. 83–90. ISBN 978-1-60558-920-6.

WANG, L.-T.; CHANG, Y.-W.; CHENG, K.-T. T. (Ed.). **Electronic Design Automation: Synthesis, Verification, and Test**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009. ISBN 9780080922003.

WANG, M.; YANG, X.; SARRAFZADEH, M. Dragon2000: standard-cell placement tool for large industry circuits. In: **Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on**. [S.l.: s.n.], 2000. p. 260–263. ISSN 1092-3152.

WESTE, N. H.; ESHRAGHIAN, K. **Principles of CMOS VLSI design**. [S.l.]: Addison-Wesley New York, 1985. v. 188.

YANG, X.; CHOI, B.-K.; SARRAFZADEH, M. A Standard-Cell Placement Tool for Designs with High Row Utilization. In: **ICCD**. IEEE Computer Society, 2002. p. 45–. ISBN 0-7695-1700-5. Disponível em: <http://dblp.uni-trier.de/db/conf/iccd/iccd2002.html#YangCS02>.

ZHENG, S. Q.; LIM, J. S.; IYENGAR, S. S. Routing using implicit connection graphs [vlsi design]. In: **VLSI Design, 1996. Proceedings., Ninth International Conference on**. [S.l.: s.n.], 1996. p. 49–52. ISSN 1063-9667.