MASTER THESIS

Zero Skew Buffered Clock Tree Synthesis supporting Integrated Clock Gating (ICG) cells

Supervisor:
Christos Sotiriou
chsotiriou@e-ce.uth.gr
Committee:
Georgios Stamoulis
georges@e-ce.uth.gr
Fotios Plessas
fplessas@e-ce.uth.gr

Student:
Aikaterini Tsilingiri
tsaikaterini@uth.gr

A thesis submitted in fulfillment of the requirements for the degree of Master in the

Circuits & Systems Laboratory (CASlab)
Department of Electrical and Computer Engineering

November 2, 2023

Abstract

Clock Tree Synthesis is one of the most important steps in the implementation of digital circuits, especially for high-performance and high-frequency applications. This work focuses on developing a Clock Tree Synthesis Flow that aims to achieve zero-skew and consists of three different algorithms. Moreover, the algorithms take into consideration the existence of Integrated Clock Gating cells, such that minimum delay is achieved for the sequential cells that belong to the fanout of the Integrated Clock Gating cells. The three algorithms that the flow consists of are the algorithm Method of Means and Medians (MMM), the Deferred Merge Embedding (DME) algorithm, and a Buffer Insertion algorithm that uses two different heuristics to decide the position of the buffers. Lastly, the proposed flow was implemented and the produced measurements were compared with an open source tool and an Industrial tool.

Περίληψη

Η Σύνθεση του Δέντρου Ρολογιού είναι ένα απο τα πιο σημαντικά βήματα για την δημιουργία ψηφιακών συστημάτων, κυριώς για κυκλωμάτα υψηλής συχνότητας και απόδοσης. Η παρούσα εργασία εστιάζει στην ανάπτυξη μιας ροής για Σύνθεση του Δέντρου Ρολογιού με στόχο την επίτευξη της ταυτόχρονης πυροδότησης των καταχωρητών και αποτελείται απο τρεις διαφορετικούς αλγορίθμους. Επίσης, οι αλγόριθμοι λαμβάνουν υπόψην τους την ύπαρξη Ολοκληρωμένων Πυλών Απομόνωσης, έτσι ώστε να επιτευχθεί η ελάχιστη καθυστέρηση στα ακολουθιακά στοιχεία που ανήκουν στο υποκύκλωμα της κάθε Ολοκληρωμένης Πύλης Απομόνωσης. Οι τρείς αλγόριθμοι που αποτελούν την ροή είναι ο αλγόριθμος Μέσων και Διαμέσων (MMM), ο αλγόριθμος Deferred-Merge Embedding (DME), και ένας αλγόριθμος Εισαγωγής Μονάδων Καθυστέρησης, ο οποίος χρησιμοποιεί δυο διαφορετικά ευριστικά για να καθορίσει τις τελικές θέσεις των μονάδων καθυστέρησης. Εν κατακλείδι, υλοποιήθηκε η ροή που προτείνεται και τα αποτελέσματα συγκρίθηκαν με Εργαλεία Ανοικτού Κώδικα και με Βιομηχανικά Εργαλεία.

Acknowledgements

I would like to thank my thesis supervisor Prof. Christos Soritiou, who was always willing to help me and for his guidance through this research project. I would also like to thank Prof. Georgios Stamoulis and Prof. Fotios Plessas for the confidence they have shown in my face.

I would also like to thank the CASlab members for their support and the friendly environment they have created, especially Christos Georgakidis and Dimitrios Valiantzas for their help during my Thesis.

Finally, I would like to thank my parents and my friends for their support throughout my years of study at the University of Thessaly.

Aikaterini Tsilingiri
Volos, 2023

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The Declarant

Aikaterini Tsilingiri

20/10/2023

Zero Skew Buffered Clock Tree Synthesis supporting Integrated Clock Gating (ICG) cells

Aikaterini Tsilingiri
tsaikaterini@uth.gr

Σύνθεση Δέντρου Ρολογιού με Υποστήριξη Ολοκληρωμένων Πυλών Απομόνωσης και Μονάδες Καθυστέρησης για Ταυτόχρονη Πυροδότηση Καταχωρητών

Αικατερίνη Τσιλιγγίρη
tsaikaterini@uth.gr

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Clock Tree Synthesis is one of the most important steps in the implementation of digital circuits, especially for high-performance and high-frequency applications. Clock Tree Synthesis algorithms are responsible for evenly distributing the clock signal along the circuit's sequential cells with minimum skew. In case the clock tree is unbalanced, it will lead the clock signal to arrive at different times for some of the sequential elements, rendering the functionality of the circuit unstable. Moreover, timing closure is very important for digital designs. Each circuit must meet some timing targets, with the most important being the setup time and hold time. The procedure of Clock Tree Synthesis has a fundamental role in achieving these requirements as it ensures that the clock signal arrives at the sequential cells at the required time.

Furthermore, the clock tree contributes majorly to the power consumption of the circuit. The high-switching activity which is caused due to the high frequency of the clock signals can easily lead to high amounts of power consumption. An effective way to reduce power consumption while at the same time preserving a minimum skew clock tree is the insertion of Integrated Clock Gating (ICG) cells. These cells are capable of disabling the propagation of the clock signal to some specific areas of the design, reducing thus the dynamic power consumption of the design.

However, inserting the Integrated Clock Gating cells can introduce some concerns during the Clock Tree Synthesis. Along the clock cycles that the ICG cells have prevented the propagation of the signal to their fanouts, the terminal points for the Clock Tree Synthesis are the clock pins of the ICG cells. On the other hand, when the logic of the ICG cells is disabled, indicating that the clock signal reaches all of the sequential cells of the design, the terminal points for the Clock Tree Synthesis consist of all the sequential cells. One of the main concerns and the motivation of this thesis is to achieve zero-skew in both cases, such that the design can meet the defined timing constraints. Moreover, in the case where the ICG cells restrain the clock signal, it is vital to achieve a minimum delay of the gated clock signal from the output of the ICG cells to the input of the sequential cells that belong to their fanout. This type of delay depends highly on the distance between each ICG cell and its fanout.

Therefore, the implementation of a valid and accurate algorithm for the Clock Tree Synthesis is fundamental for the design to operate correctly. This thesis proposes a three-step Clock Tree Synthesis Methodology, that aims to achieve a Zero-Skew Buffered Clock network, in the presence of Integrated Clock Gating Cells. The proposed flow consists of the algorithm of Method of Means and Medians (MMM), the Deferred Merge Embedding (DME) algorithm, and the Buffering Algorithm, which uses two heuristics.

Chapter 2

Theoretical Background

2.1 EDA Background

2.1.1 Introduction to EDA

An integrated circuit (IC) is constituted of a group of electronic components that are placed on a substrate of semiconductor material (usually silicon). Integrated circuits (ICs) have undertaken a substantial and influential role in the development of the contemporary technological landscape. Nowadays, ICs have found extensive applications in many devices that are commonly used in daily operations, like computers, cell phones, and even in means of transport like cars and airplanes.

Electronic Design Automation (EDA) or electronic computer-aided design (ECAD) refers to the software tools responsible for designing electronic systems like Integrated Circuits (ICs). The first kind of automation in the design of electronic systems was introduced in the mid-1970s, as before that, the ICs were designed by hand. The need for EDA tools emerged due to the rapid growth of the electronics industry, a phenomenon further substantiated by Moore's Law in figure 2.1. According to Moore's Law, the number of transistors in an integrated circuit (IC) doubles about every two years. As it becomes apparent over time, the integrated circuits (ICs) become even more complex rendering the development of the EDA tools necessary, to achieve low power consumption and precise timing, in a reasonable timing period.

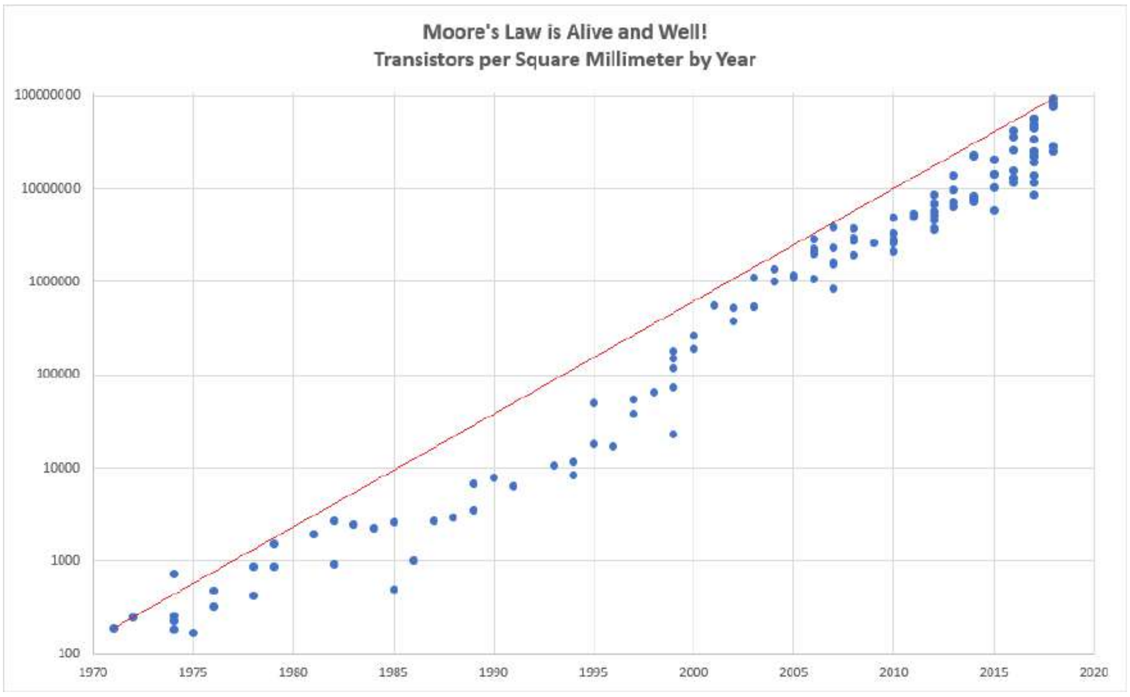


Figure 2.1: Moore's Law from 1970 till 2020

The flow of steps that the Electronic Design Automation (EDA) Tools follow is depicted in figure 2.2. Initially, the specifications of the integrated circuit (IC) are determined. Then, the RTL Synthesis is performed, which maps the RTL to equivalent logic gates using a synthesis tool. The next five steps (Partitioning, Floorplanning, Placement, Clock Tree Synthesis, and Routing) consist of the Physical Design and aim to transform the synthesized netlist into the layout. During the Verification step, signoff checks are performed, like physical verification checks, timing analysis, and logical equivalence checking. Finally, the GDSII stream format (GDSII) is produced, which is a binary file format that represents the layout data in a hierarchical format.

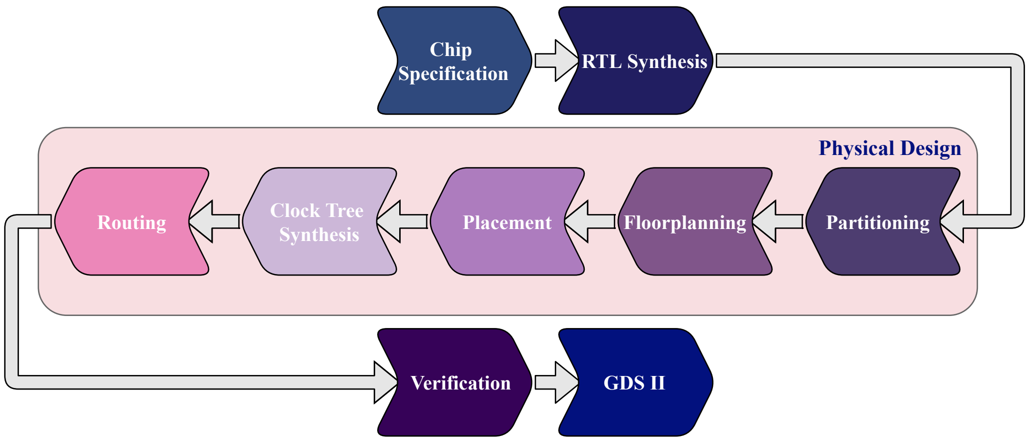


Figure 2.2: EDA Flow

2.1.2 Physical Design

The Physical Design is considered one of the most critical stages in Electronic Design Automation (EDA) flow, as it highly affects the performance, power consumption, and area of the integrated circuit. As shown in figure 2.2, which depicts the EDA Flow, the Physical Design consists of five different steps: Partitioning, Floorplanning, Placement, Clock Tree Synthesis, and Routing. Each of these steps is referred to in the following subsections and depicted in figure 2.3.

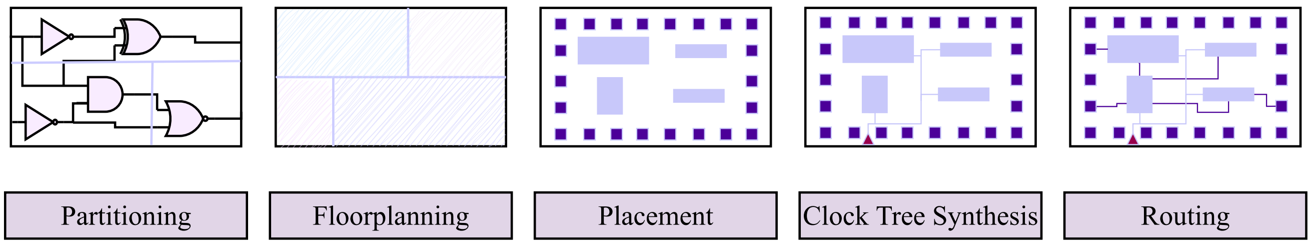


Figure 2.3: Steps of Physical Design

Partitioning

Partitioning has as its main goal to divide the circuit into smaller sub-circuits to achieve better performance, lower power consumption, and improved reliability. The division of the main circuit into smaller sub-circuits is done recursively using the appropriate cost functions that minimize the wirelength between the sub-circuits.

Floorplanning

The step of Floorplanning aims to determine the topology of the layout. Based on the interconnection requirements of the circuit, the algorithms of floorplanning try to find the relative positions of the modules.

Placement

The most important step during the Physical Design is Placement, which contains three subsequent steps, **Global Placement (GP)**, **Legalization (LG)**, and **Detailed Placement (DP)**. The first step, Global Placement, aims to evenly distribute the cells, allowing them to overlap without resolving any constraint violation. The next step, Legalization, emphasizes resolving any violation from the prior step and lastly, Detailed Placement performs precise moves targeting to optimize the circuit metrics, like power and performance. Overall QoR of the design greatly depends on the quality of the circuit placement, rendering this step crucial.

Clock Tree Synthesis

Clock Tree Synthesis is the step that aims to distribute the clock signal throughout an entire chip, while at the same time trying to minimize the power consumption and the skew of the clock network since in most chips, as much as half of the chip's power consumption can be associated with the clock trees. Some vital terms used in Clock Tree Synthesis are presented below, such that the present thesis becomes more understandable.

Insertion delay is the time it takes the clock signal to propagate through the clock tree to the sinks, whereas sinks are defined as the sequential elements of the circuit.

Global Skew is the difference between the slowest and fastest clock sink delays, as it is defined below [1]:

$$\text{Global Skew} = max_{i \in sinks}(d_i) - min_{j \in sinks}(d_j) \tag{2.1}$$

Local Skew is the difference in the delays of a pair of related clock sinks, d_A, and d_B [1]:

$$\text{Local Skew} = d_A - d_B \tag{2.2}$$

Slew is the transition time of the signal, for the output voltage to go from 90% of V_{max} to its 10% value. Slew is vital because it highly impacts the setup time and hold time.

Clock Latency is the total time taken by the clock signal to reach the clock source to the clock pin of a selected sequential cell [2].

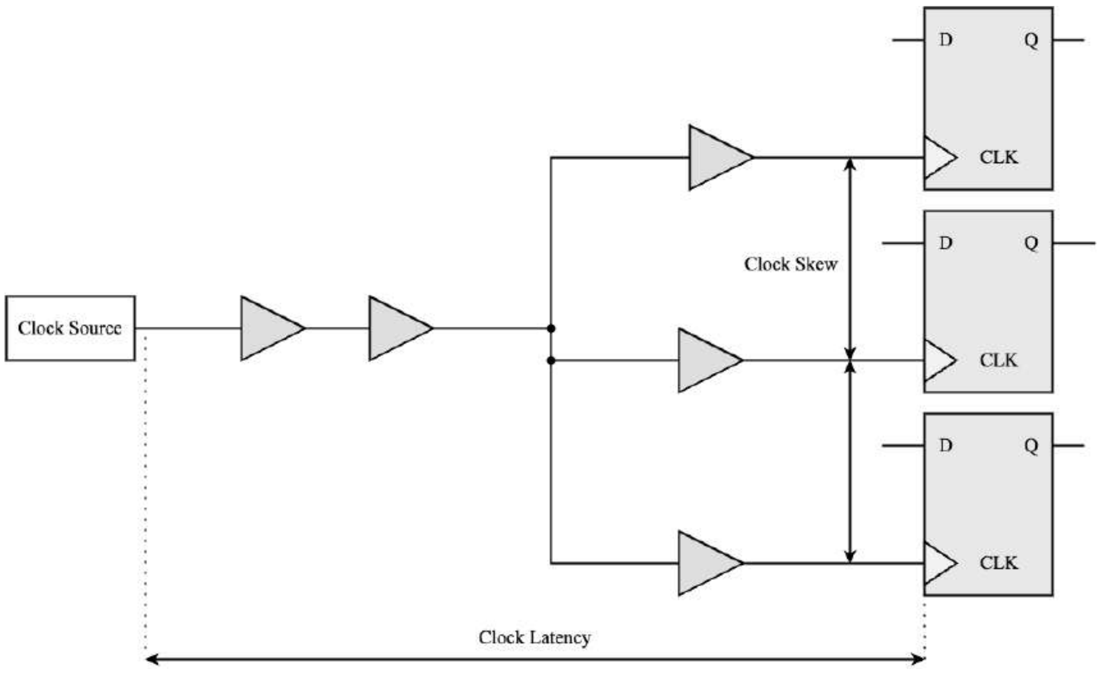


Figure 2.4: Clock Latency and Clock Skew in a Clock Tree

Routing

The final step of the Physical Design is the Routing step. The Router connects the pins of the already placed cells using the available metal layers while at the same time minimizing the length and reducing the congestion of the metals in the floorplan.

2.1.3 Static Timing Analysis

Static Timing Analysis (STA) is a crucial stage within the EDA Flow since it is a method of verification. All possible paths of the circuit (from each startpoint to each endpoint) are inspected in order to check if there are any timing violations under the worst-case conditions. It is important that each circuit must operate at the target clock frequency, without any timing violations.

During the procedure of Static Timing Analysis (STA), the circuit is represented using a timing graph $G = (V, E)$, where the nodes of the graph (V) consist of the inputs and outputs of the logic gates in the circuit. Each node of graph G is associated with the term **slack**, which is defined as the difference between the Required Arrival Time (RAT) and Arrival Time (AT). Slack can be positive, indicating that the design meets the timing constraints and can be further optimized. Negative slack means that the design does not satisfy the defined requirements and setup and hold violations are present. Lastly, zero slack shows that the design is critically working at the desired frequency.

Setup time and **Hold time** constraints are essential aspects of STA because they guarantee the correct synchronization of the data and the clock signals. The **Setup constraints** define the minimum time that the data signal must be stable before the arrival of the active clock signal. Supposing that there are setup violations in a design, this will result in data not being captured properly at the next clock edge. Let a pair of Flip flops, FF_i and FF_j, be connected by a combinational path. The equation (2.3) [3] guarantees that the time it takes to transfer the data from the FF_i to the FF_j, $d(i, j)$, is not too small.

$$T_{q_i} + d(i,j) \leq P - T_{s_j} \implies d(i,j) \leq P - T_{s_j} - T_{q_i} \tag{2.3}$$

where T_{q_i}, denotes the time that the data are available at the launching Flipflop (FF_i), P defines the target clock period, and T_{s_j}, the setup time.

The figure 2.5 depicts the case where setup violation is present in the design. As it is shown, the data travel from the output of FF_i (Q) to the input of FF_j (D) through some combinational logic. This logic is very slow since the new data arrive at D of FF_j late resulting in losing the data.

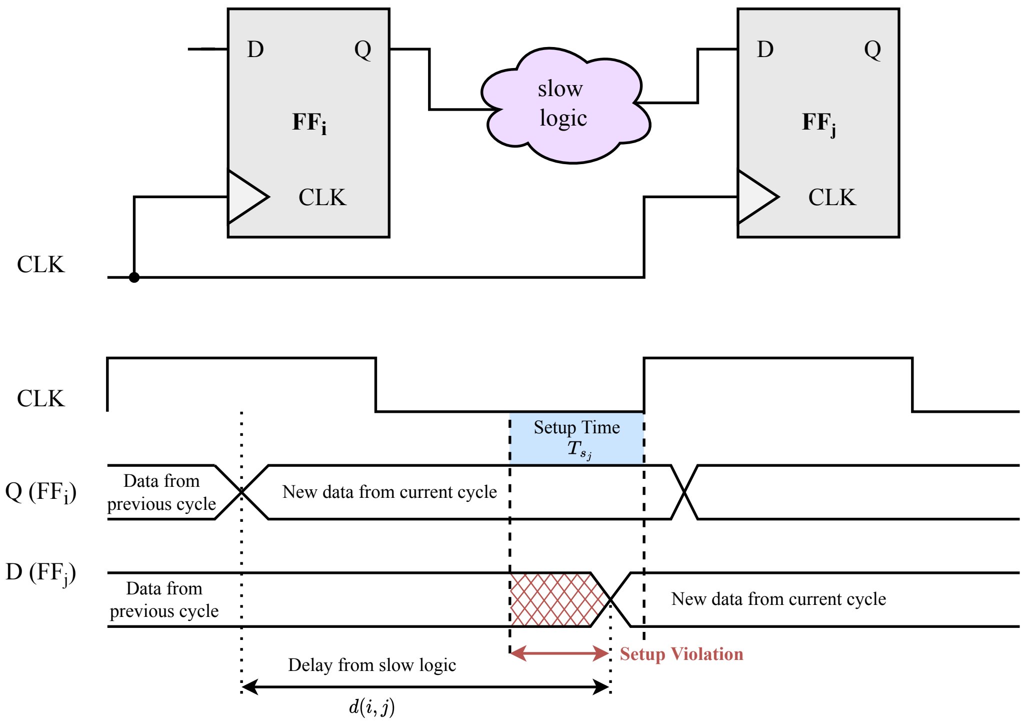


Figure 2.5: Example of Setup Violation between a pair of Flipflops

The **Hold constraints** ensure that the data transfer is not too rapid, since the data must be stable for a time (T_{h_j}). In the case that Hold violations are present in a design, then incorrect data will be captured. The equation (2.4) [3] guarantees that the data are captured correctly by the capturing Flipflop FF_j.

$$T_{q_i} + d(i,j) \geq T_{h_j} \implies d(i,j) \geq T_{h_j} - T_{q_i} \tag{2.4}$$

The figure 2.6 presents an example where the data arrive too fast from the output of FF_i to the input of FF_j leading to a hold violation.

As it is shown, the data travel from the output of FF_i (Q) to the input of FF_j (D) through some combinational logic. This logic is very slow since the new data arrive at D of FF_j late resulting in losing the data.

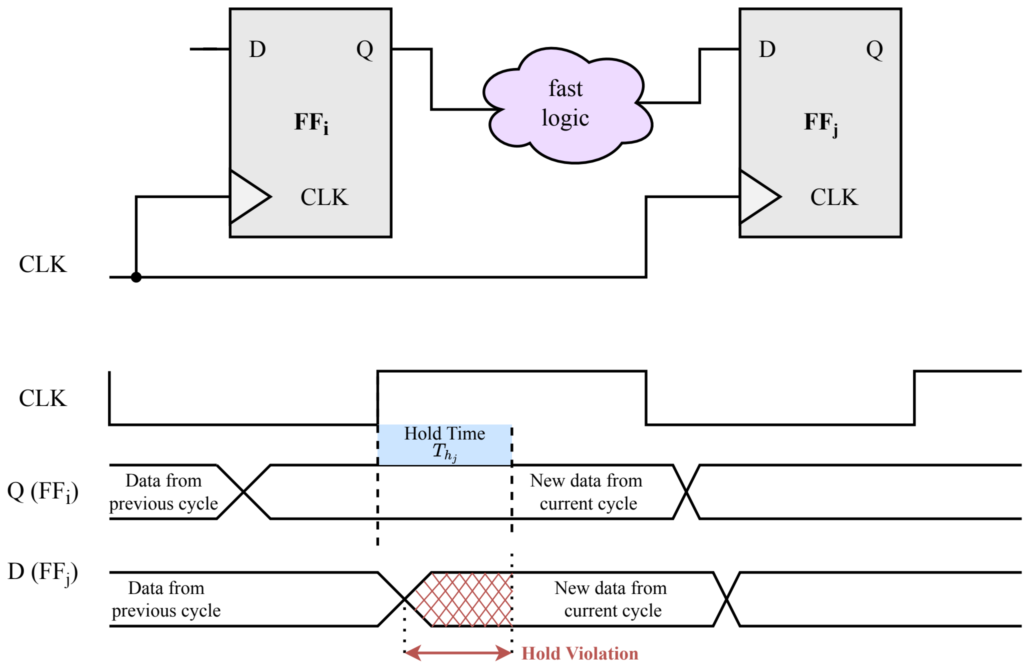


Figure 2.6: Example of Hold Violation between a pair of Flipflops

As **Arrival Time (AT)** is defined as the time when a signal arrives at an exact place at a specific time. **Required Arrival Time (RAT)** represents the specific time when the data is required to be present at the specific pin. As **Slack** is defined the difference between the Required Arrival Time (RAT) and the Arrival Time (AT):

$$slack = RAT - AT \tag{2.5}$$

2.2 Integrated Clock Gating Cells (ICG cells)

Clock gating is one of the most used power management methods in digital circuits. The insertion of clock gating cells into a design aims to reduce power consumption by suppressing the transitions from propagating to parts of the clock path [4].

There are various ways to implement an Integrated Clock Gating cell. The following architecture using a Negative Edge Triggered Latch and an AND gate (figure 2.7) is considered as the most optimal [5]. The other alternatives are latch-free clock gating architectures that usually consist of only an AND or OR gate (depending on the edge on which flip-flops are triggered) mentioned in [4]. These architectures, due to their simplicity, can cause glitches in the output signal which is the gated clock signal that will reach the sequential cells that consist of the fanout of the ICG cell, as can observed in figure 2.8.

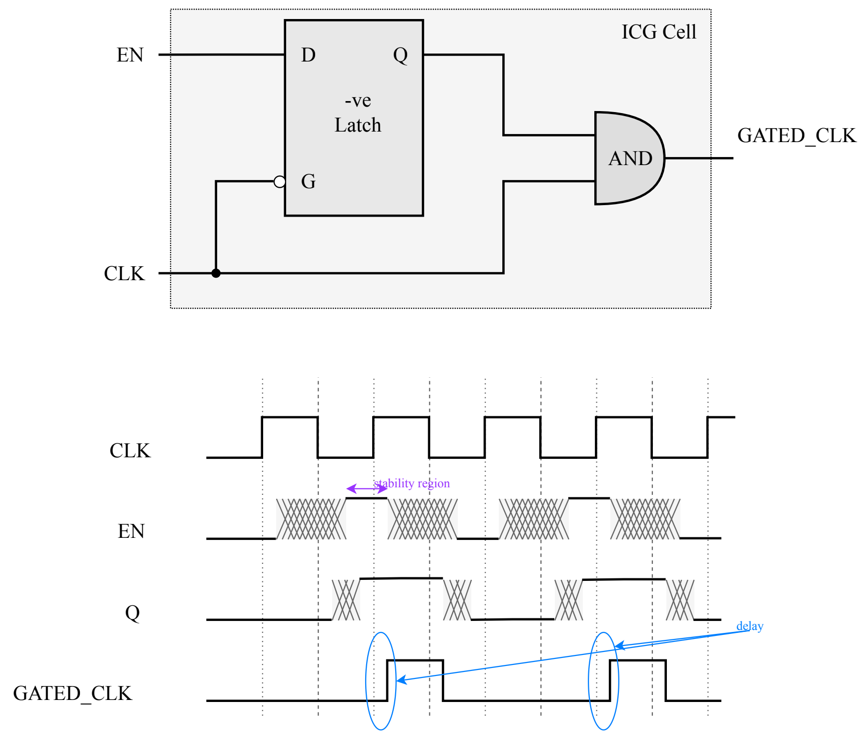


Figure 2.7: Integrated Clock Gating Cell and waveforms

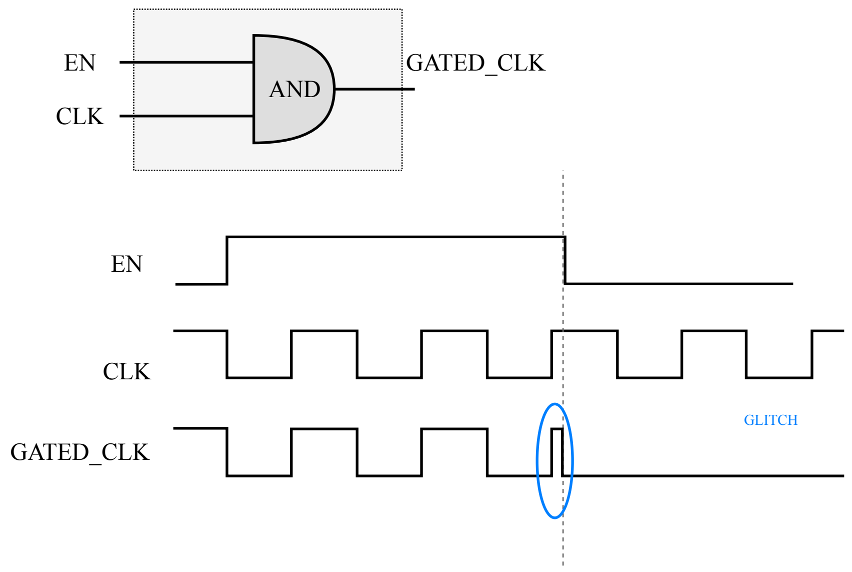


Figure 2.8: Clock Gating Cell with AND gate and the waveforms

As it can be observed from figure 2.8, a glitch is produced. This architecture requires that all enable signals must be held constant from the rising edge of the clock signal (CLK) until the falling edge of the clock signal such that the produced glitch is avoided. However, this is not always possible in the designs, and therefore the latch-based clock gating architecture in figure 2.7 is adopted for this thesis.

This architecture provides an extra "filtering" into the output signal such that the glitch is avoided. The latch captures and holds the enable signal (EN) until the complete clock pulse (CLK) has been generated. This leads to the output of the latch (Q) to be more stable (in comparison with the latch-free architectures) and further to the gated clock signal (GATED_CLK) to be glitch-free. In figure 2.7 it is shown that the gated clock is propagated only when both the clock signal and the output of the clock are high.

Chapter 3

Existing Work

The problem of Clock Tree Synthesis has been studied by many researchers over the years, and therefore there is a wide variety of algorithms that perform Clock Tree Synthesis. Many architectures of Clock Trees have been widely used during the procedure of Clock Tree Synthesis, each one of them aiming at a different target. However, there is not a wide variety of techniques or algorithms that perform zero-skew Clock Tree Synthesis while at the same time satisfying multiple constraints such as the maximum capacitance, and the maximum slew with an arbitrary placement of the sequential elements of the design.

The first algorithm described in [6] is based on the Geometric (Manhattan) distance of the sequential cells. The primary goal of the algorithm is to create a Binary Tree with the sequential cells being at the leaf nodes in a recursive Bottom-Up approach while achieving minimal skew. Initially, a net is defined at the clock input of each sequential cell. The algorithm aims to find a point to connect each pair of sequential elements, such that the delay from that point to each cell is equal. This procedure is repeated and in each iteration, the new point is defined as a new node of the Binary Tree. The algorithm terminates when the clock source is reached. Therefore, this algorithm can guarantee equal delays in each sub-tree of the binary tree, only if the connection point falls on a routable location. Otherwise, the clock tree cannot be considered as zero skew.

The algorithm presented in [7] explores the architecture of the H-Tree clock networks and the Spine or "Fishbone" clock network. This algorithm is mainly used in the Opensource tool, TritonCTS [8], in which they have introduced a new method for Clock Tree Synthesis that combines the architectures mentioned earlier, using a dynamic programming-based method. The algorithm of the presented work aims to construct a Generalized H-Tree while inserting buffers to minimize the clock power, subject to the given placed solution, the number of sink regions (such that each region contains less than 40 sinks), the maximum clock skew constraint, clock latency, transition time and capacitance constraint. The first step of their algorithm consists of formulating a dynamic programming problem to co-optimize branching patterns and buffering, using the total sink capacitance and layout region area and aspect ratio. The second step uses the K-means algorithm [9] and formulates an integer linear programming problem that determines the final positions of the clock buffers.

Another approach in [10] aims to satisfy tight slew constraints during the Clock Tree

Synthesis. This method is based on the Deferred Merge Embedding (DME) algorithm (which is further explained in section 4.2) and uses specific cost functions for the computation of the merging point to target the defined slew constraints while at the same time achieving the most optimal skew. More specifically, this work proposes a new method of computation for the merging point and a new method to simultaneously check and satisfy slew and skew constraints. The combination of these proposals, also, helps to reduce the power consumption of the clock network. According to the experimental results, the Slew-Driven Clock Tree Synthesis methodology has generated promising results, since the tight slew constraints can be satisfied and the clock power is reduced. However, the execution time of the algorithm is notably high because, after each buffer insertion, slew propagation is performed to ensure that the requirements are met.

A symmetrical Obstacle-Aware Clock Tree Synthesis methodology with integrated buffer insertion is presented in [11]. The algorithm using the total number of sequential cells of the design defines the levels of the generated clock network. Moreover, the node merging is performed using the Tilted Rectangular Regions (TRRs), which will be also explained in more detail in section 4.2. Regarding the Buffering strategy they propose, they aim to achieve an equal number of buffers at each path and they mainly use a Look-Up Table (LUT) in order to determine the most optimal buffer type for each case. Lastly, after having decided on the topology and the buffering of the clock network, this approach proposes an Obstacle-aware routing algorithm. Their Obstacle-Aware Routing methodology is based on vectors. Initially, the algorithm identifies all of the wires that intersect with obstacles and defines a direction vector, which begins from the parent node (the source of the clock tree) and has as an endpoint the sequential cells of the design. The presented flow offers favorable results, since based on the experimental results, the algorithm provides a reduction in skew and capacitance resource on average (compared with other equally important works). Furthermore, this symmetrical structure can offer the ability to resist OCV.

Another approach that is based on the Deferred Merge Embedding algorithm is presented in [12]. The proposed methodology combines simultaneous routing, wire sizing, and buffer insertion. They use a generalized concept of the merging segment mentioned in section 4.2, called the Merging Segment Set, mainly aiming for a more accurate generation of the merging point since the concept of the Merging Segment Set offers more zero-skew merging locations. The implemented Buffer Insertion algorithm aims to maintain an equal number of buffers in each root-to-leaf path of the binary tree. The proposed methodology achieves better wirelength, in comparison to the original Deferred Merge Embedding algorithm, and the clock delay is greatly reduced.

Studying the existing works that have been used for Clock Tree Synthesis, it can be observed that most algorithms use as the foundation for their methodology the algorithm of Deferred Merge Embedding (section 4.2). Therefore, this thesis has chosen this algorithm in order to construct the clock tree structure for each design. This methodology has plenty of room for optimizations in different stages of the Clock Tree Synthesis in order to improve the quality of the clock signal.

Chapter 4

Proposed Methodology

The problem this thesis is trying to solve is the construction of a Zero-Skew Clock Tree, even when Integrated Clock Gating cells are present in the design. The proposed methodology consists of three different algorithms that have as goal to distribute evenly the clock signal along the sequential cells of the design. The first algorithm is the Method of Means and Medians (MMM), which aims to create a binary tree that has as leaves the sequential cells of the design. The second algorithm is the Deferred Merge Embedding (DME) algorithm which determines the final positions of the inner nodes of the binary tree. Lastly, two heuristics are proposed in order to insert buffers into the design for better timing results. The last section of this chapter presents the flow that is essential for achieving zero skew in a design when Integrated Clock Gating cells are present.

4.1 Clock Tree Topology Generation

The most fundamental tree structure (used in the first step of Clock Tree Synthesis) is the H-tree. This structure demands the clock sinks to be placed symmetrically and no obstacles present in the chip layout. These two conditions render the structure impractical since the clock tree sinks (registers, Flip-Flops, latches, integrated clock gating cells etc.) are placed all over the chip in most designs. Therefore, many algorithms have been proposed for clock tree construction, having as the foundation the H-Tree.

4.1.1 Method of Means and Medians (MMM)

The algorithm of Method of Means and Medians [13] is an iterative algorithm that divides/splits (horizontally or vertically) the number of sinks into two sub-circuits until each sub-circuit contains only one sink. Let $S = \{s_1, s_2, s_3, ..., s_k\}$ be the sinks of a circuit that needs the clock signal to reach, where each sink is represented using its (x, y) coordinates. The complexity of the algorithm is $O(klogk)$, where k is the total number of clock sinks.

The algorithm, in each iteration, defines the center of mass of the set of sinks $(x_c(S), y_c(S))$:

$$x_c(S) = \frac{\sum_{i=1}^{k} x_i}{k}, \tag{4.1}$$

$$y_c(S) = \frac{\sum_{i=1}^{k} y_i}{k}, \tag{4.2}$$

that defines the coordinates of the merging point generated during the split (for the selected sub-circuit). Then, the sinks of the sub-circuit are sorted (in increasing order) based on their x-coordinates or y-coordinates (depending on the direction of the chosen split) and they are divided on the median sink. The algorithm arbitrarily decides the direction of the first cut (horizontal or vertical).

Let the algorithm split a sub-set S_x based on the x-coordinates of the sinks of S_x. After the division of the sinks, two sub-sets will be generated $S_i, S_j \in S_x$, for which the following statements hold: $x_i \leq x_j$ if $i < j$.

Considering all the above, having a set of sinks S, the algorithm defines the following four rules, such that four sub-circuits can be generated:

$$S_L(S) \equiv \{s_i \in S_x | i \leq \lceil k/\mathbf{2} \rceil\} \tag{4.3}$$

$$S_R(S) \equiv \{s_i \in S_x | \lceil k/\mathbf{2} \rceil < i \leq k\} \tag{4.4}$$

$$S_T(S) \equiv \{s_i \in S_y | \lceil k/\mathbf{2} \rceil < i \leq k\} \tag{4.5}$$

$$S_B(S) \equiv \{s_i \in S_y | i \leq \lceil k/\mathbf{2} \rceil\} \tag{4.6}$$

where S_L and S_R are generated when the sinks are split based on the x-coordinates and S_T and S_B are generated based on the y-coordinates. Based on the algorithm, the following inequalities should hold for the two sub-regions (left, right, and top, bottom):

$$||S_L| - |S_R|| \leq \mathbf{1} \text{ and } ||S_B| - |S_T|| \leq \mathbf{1} \tag{4.7}$$

As mentioned, the algorithm (presented in algorithm 1) determines the exact coordinates (coordinates of the center of mass) of the inner nodes in each iteration. However, these generated nodes cannot be realistically defined in a circuit, due to the obstacles (other cells or macros) that may be present and therefore they do not guarantee zero-skew for the clock signal distribution (if some adjustment of their locations must be done). Moreover, even if the generated inner nodes do not overlap with the already placed cell, the algorithm does not guarantee minimum wirelength and it is highly possible to generate a non-rectilinear Steiner Tree.

Therefore, in this thesis, this algorithm was mainly used in order to divide the clock sinks and create the structure of a binary tree that will be later used as input for the Deferred-Merge Embedding (DME) Algorithm, explained in section 4.2.

Algorithm 1 Method of Means and Medians (MMM) Algorithm

```
if $S \leq 1$ then
    return
end if
$x_0 = x_c(S); y_0 = y_c(S);$
$x_{left} = x_c(S_L(S)); y_{left} = y_c(S_L(S));$
$x_{right} = x_c(S_R(S)); y_{right} = y_c(S_R(S));$
route from $(x_0, y_0)$ to $(x_{left}, y_{left})$ and $(x_{right}, y_{right})$
$x_{bot\_left} = x_c(S_B(S_L(S)));$
$y_{bot\_left} = y_c(S_B(S_L(S)));$
$x_{top\_left} = x_c(S_T(S_L(S)));$
$y_{top\_left} = y_c(S_T(S_L(S)));$
$x_{bot\_right} = x_c(S_B(S_R(S)));$
$y_{bot\_right} = y_c(S_B(S_R(S)));$
$x_{top\_right} = x_c(S_T(S_R(S)));$
$y_{top\_right} = y_c(S_T(S_R(S)));$
route from $(x_{left}, y_{left})$ to $(x_{bot\_left}, y_{bot\_left})$ and $(x_{top\_left}, y_{top\_left})$
route from $(x_{right}, y_{right})$ to $(x_{bot\_right}, y_{bot\_right})$ and $(x_{top\_right}, y_{top\_right})$
$MMM(S_B(S_L(S)));$
$MMM(S_T(S_L(S)));$
$MMM(S_B(S_R(S)));$
$MMM(S_T(S_R(S)));$
```

Example using the Method of Means and Medians

An example is presented to better understand the algorithm and how it was used for the initial Binary Tree Generation. The figure 4.1 depicts a schematic of a circuit, in which, the placed cells (FF1, FF2, FF3, ..., FF10) are considered as clock tree sinks. The algorithm, in this implementation, initially sorts the sinks based on their x-coordinates (in increasing order) and the following sorted list is obtained :

$$\text{Sorted sinks} = \{FF4, FF5, FF6, FF7, FF2, FF8, FF1, FF3, FF9, FF10\}$$

Having the sorted list of the sinks, the algorithm divides the set of sinks into two sub-sets, in the positions of the median sink, as it was mentioned, (FF2 - FF8), resulting in the two following sets:

$$S_1 = \{FF4, FF5, FF6, FF7, FF2\} \text{ and } S_2 = \{FF8, FF1, FF3, FF9, FF10\}$$

The same procedure is followed for the two subsets, with the only difference being that in this iteration, the sinks are sorted in ascending order based on their y-coordinate.

The following figures show in detail the steps of the algorithm.

- The figure 4.2 depicts the first cut/split done in the circuit that creates the two sub-circuits S_1 and S_2 mentioned before. After this split is performed, the Root node of the binary tree presented in figure 4.4 is generated.
- The figure 4.3 shows the generated circuits and their division based on the y-coordinates of the sinks. The sinks of the sub-circuits S_1 and S_2 are sorted based

on their y-coordinates and then the median is found. Four new sub-circuits are generated, S_3 and S_4, which are derived from S_1 and S_5 and S_6, derived from the sub-circuit S_2. The division of S_1 generates the Inner Node 1 and the division of S_2 generates the Inner Node 2, as it is depicted in the Final Binary Tree in figure 4.4).

As evidenced by observations and experiments, the generated binary tree from Method of Means and Medians [13] will always be a balanced binary tree $T(S)$, with a total number of nodes equal to $\mathbf{2} \cdot k - \mathbf{1}$, where k is the number of clock sinks.

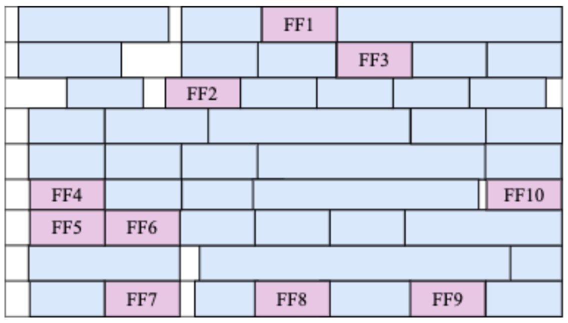


Figure 4.1: Initial Design for Method of Means and Medians

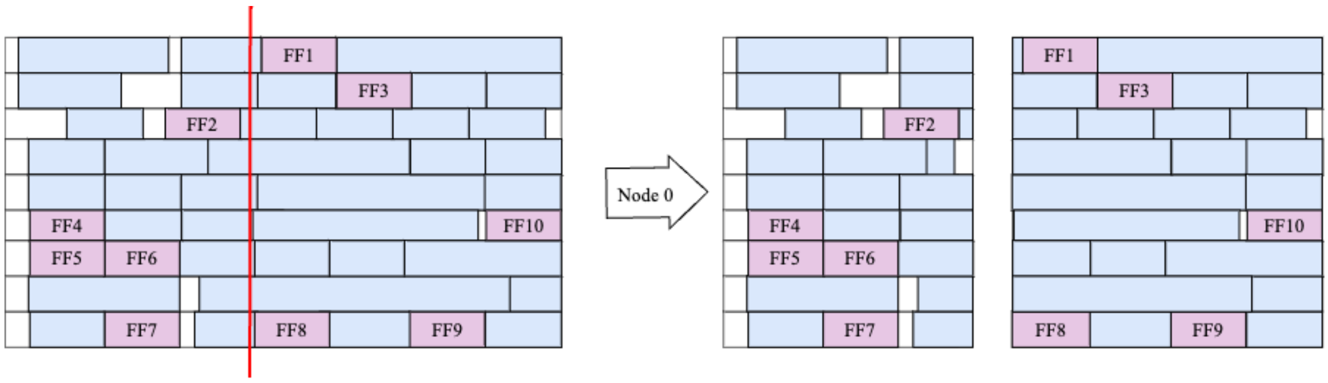


Figure 4.2: Generation of Root Node 0

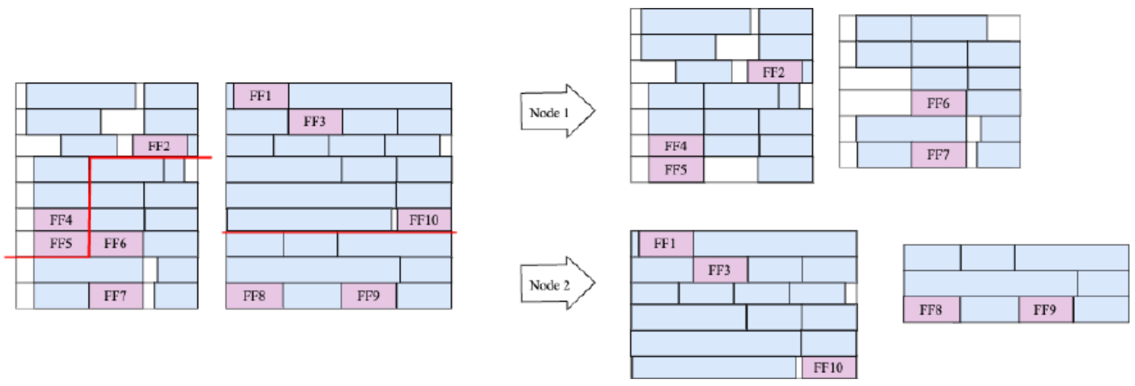


Figure 4.3: Generation of Inner Nodes 1 and 2

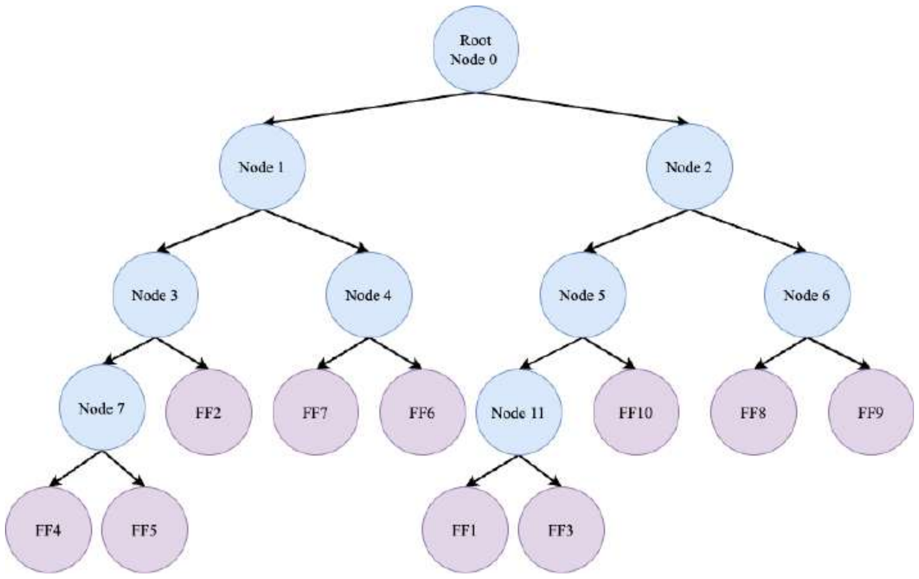


Figure 4.4: Binary Tree Generated by MMM

4.2 Deferred-Merge Embedding (DME) Algorithm

The next step of the Clock Tree Synthesis is the clock tree routing having as input an abstract clock tree topology. This stage of the algorithm is crucial since it attempts to minimize the skew or accomplish zero-skew while achieving the minimum possible wirelength. This problem is addressed by the algorithm of Deferred-Merge Embedding (DME) [14], which consists of two phases (Bottom Up and Top Down, explained in section 4.2.1 and section 4.2.2 accordingly). The algorithm, for a given binary tree representing the clock tree $T(S)$, finds the accurate location of the inner nodes, such that the wirelength and the delay at the sinks of the clock tree are minimal. Furthermore, **the results of the algorithm highly depend on the input binary tree**. In this thesis, the input binary tree is derived by the Method of Means and Medians (MMM) explained in section 4.1.1.

Before explaining the steps of each phase, it is important to mention some fundamental terms and assumptions used in the algorithm. To begin with, the algorithm, in order to find the accurate positions of the inner nodes, uses two main terms, the **Merging Segment (MS)** and the **Tilted Rectangular Region (TRR)**. A merging segment (MS) is defined as the set of all possible locations of the inner node such that minimal wirelength and zero-skew are achieved. Therefore, the merging segment is a line segment with a slope $+1$ or -1, as shown below in figure 4.5, and each inner node is associated with one merging segment. A Tilted Rectangular Region (TRR) is defined as the set of all of the points that have an equal distance from the merging segment. An illustration of a Merging Segment and a Tilted Rectangular Region is given in figure 4.5 for a better understanding of the terms.

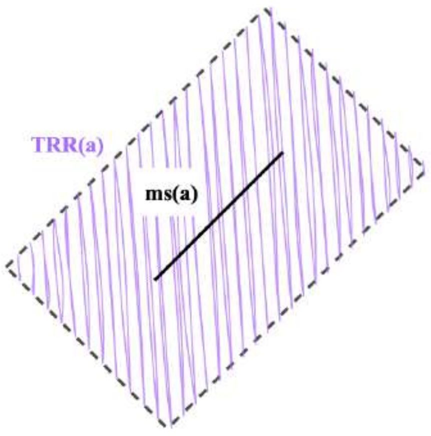


Figure 4.5: Example of Merging Segment (ms) and Tilted Rectangular Region(TRR)

Another important term of the algorithm is the **edge length** ($edge\_length_a$). Each inner node is associated with an edge length and it represents the Manhattan distance from the current node to the parent node such that the delay and wirelength are minimal.

The delay calculation from the clock root to the clock sinks is achieved using the Elmore Delay Model. The implemented algorithm uses two different representations of the wire between the clock source and the clock sinks of the circuits as shown in figure 4.6. The first model (a) is the RC model and the second (b) is the Pi model.

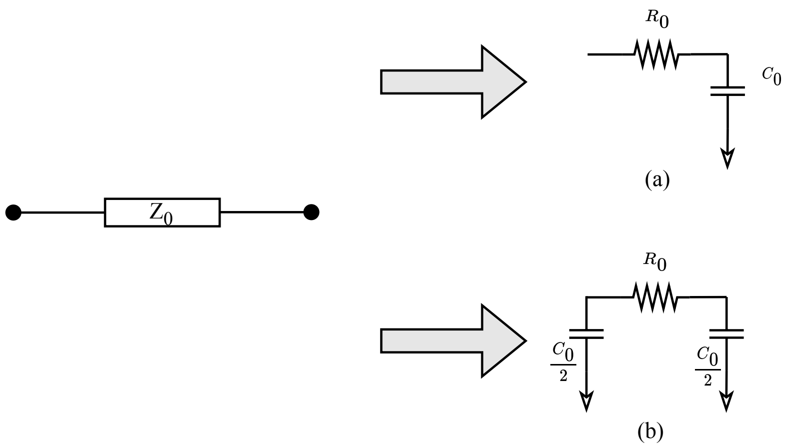


Figure 4.6: Wire Representations: (a) RC Model and (b) Pi Model

In both cases, the algorithm applies the Elmore delay, in order to achieve the balancing of the delays. Let the merging point of subtree 1 and subtree 2 be an inner node of the binary tree generated by MMM that has as children subtree 1 and subtree 2. The algorithm, in order to guarantee zero skew and minimum wirelength between the two children, aims to find the optimal wirelength between the inner node and each child. The following formula with the delays is applied in both models:

$$t_{p1} = t_{p2} \implies t_{Z_1} + t_1 = t_{Z2} + t_2, \tag{4.8}$$

where t_{Z_1} and t_{Z_2} are the delays of the impedance in figure 4.7 and figure 4.8 and t_1 and t_2 are the delays of each sub-tree 1 and 2.

Applying the Elmore Delay to the RC model, we get the following formulas:

$$t_{Z_1} = \ln \mathbf{2} \cdot r_1 \cdot c_1 \text{ and } t_{Z_2} = \ln \mathbf{2} \cdot r_2 \cdot c_2, \tag{4.9}$$

$$r_1 = r \cdot x \cdot l, \tag{4.10}$$
$$r_2 = r \cdot (\mathbf{1} - x) \cdot l, \tag{4.11}$$

$$c_1 = c \cdot x \cdot l, \tag{4.12}$$
$$c_2 = c \cdot (\mathbf{1} - x) \cdot l, \tag{4.13}$$

where r and c are resistance and capacitance per unit accordingly.

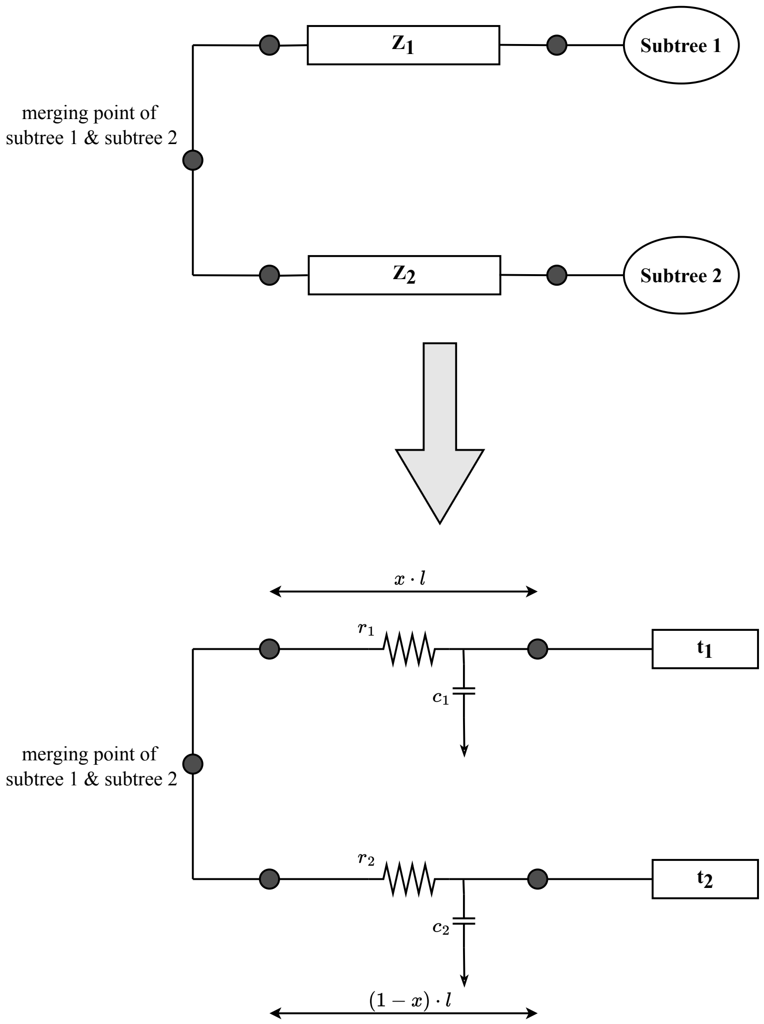

Z1
Subtree 1
merging point of subtree 1 & subtree 2
Z2
Subtree 2
$x \cdot l$
r_1
c_1
t1
merging point of subtree 1 & subtree 2
r_2
c_2
t2
$(1 - x) \cdot l$

Figure 4.7: Example of an inner node and the children of the Binary Tree with RC-model

The second approach uses a distributed RC circuit using the Pi model (figure 4.6 (b)). This model is used because it provides better accuracy for the delay calculation.

The between wire is modeled using the Pi type (figure 4.8) and the Elmore Delay is applied such that the delay from the parent (inner node/merging point) to each of the children is equal. Applying the Elmore Delay to the Pi model, we get the following formulas:

$$t_{Z_1} = r_1 \cdot (\frac{c_1}{2} + C_1) \text{ and } t_{Z_2} = r_2 \cdot (\frac{c_2}{2} + C_2),$$

$$r_1 = r \cdot x \cdot l,$$

$$r_2 = r \cdot (1 - x) \cdot l,$$

$$c_1 = c \cdot x \cdot l,$$

$$c_2 = c \cdot (1 - x) \cdot l,$$

where r and c are resistance and capacitance per unit accordingly.

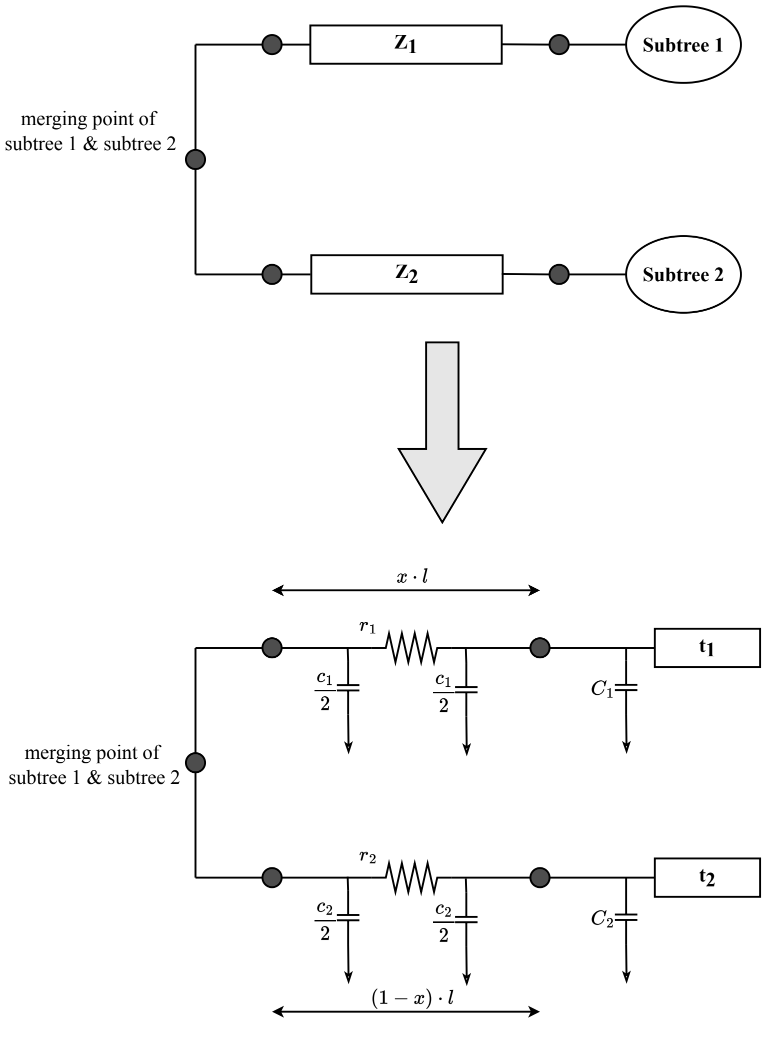


Figure 4.8: Example of an inner node and the children of the Binary Tree with Pi-model

4.2.1 Bottom-Up Phase: Generation of Merging Segments

The algorithm presented in algorithm 2, during the Bottom-Up Phase, generates the Manhattan arcs or merging segments (ms) of the inner nodes. The generation of the merging segments is done in a Bottom-Up traversal of the binary tree because the computation of the merging segment of each node is related to the merging segments of the children of the node.

As it was mentioned above, each node of the binary tree is associated with a merging segment. If the node of the tree is a sink (flipflop, register, latch, integrated clock gating cell, etc.), then the merging segment is the location of the cell. Otherwise, if the node of the tree is an inner node, then the merging segment is computed based on the merging segments of its left and right children. Let v be an inner node and a and b be the left and right children accordingly. We define as $ms(a)$ and $ms(b)$ the merging segments of the children and as $|e_a|$ and $|e_b|$ the edge lengths of each child node. For each node v, any point on the $ms(v)$ should allow a and b to be merged with minimum wiring cost $|e_a|+|e_b|$. Each node is associated with an edge length that is calculated using the Elmore Delay model as it will be mentioned below.

Let L be the Manhattan distance between the children a and b. If the nodes a and b are CTS sinks, then L is equal to the Manhattan distance between the two cells. Otherwise, if one or both of the children are inner nodes, then L is equal to the Manhattan distance between the merging segments and it is defined as $L = min\{d(p_1, p_2)|p_1 \in ms(a), p_2 \in ms(b)\}$. As it was mentioned above, the goal of DME is to achieve zero skew, therefore the Elmore Delay from the parent node v to child a and to child b should be equal.

The next two sub-sections present mathematical formulas for the wirelength between the parent node and the children nodes, for each wire model.

Algorithm 2 Bottom-Up Phase of Deferred-Merge (DME) Embedding Algorithm

```
  for for each node v in G (bottom-up order) do
      if v is a sink node then
          ms(v) ← {$pl(v)$}
    else
        Let a and b be the children of v
        Calculate_Edge_Lengths($|e_a|, |e_b|$)
        Create TRRs $trr_a$ and $trr_b$ as follows:
        $core(trr_a) \leftarrow ms(a)$
        $radius(trr_a) \leftarrow |e_a|$
        $core(trr_b) \leftarrow ms(b)$
        $radius(trr_b) \leftarrow |e_b|$
        ms(v) ←trr$_a \cap trr_b$
    end if
end for
```

Bottom-Up Generation of Merging Segments using RC Model

The algorithm in this step uses the defined formula for the delays mentioned in the previous section ($t_{Z_1} + t_1 = t_{Z_2} + t_2$) to determine the exact wirelength such that the delays are equal. After expanding this formula for the RC model, the algorithm solves for x:

$$t_{Z_1} = \ln \mathbf{2} \cdot r_1 \cdot c_1 \text{ and } t_{Z_2} = \ln \mathbf{2} \cdot r_2 \cdot c_2 \implies \tag{4.14}$$

$$x = \frac{\ln \mathbf{2} \cdot r \cdot c \cdot l^2 - t_1 + t_2}{\mathbf{2} \cdot \ln \mathbf{2} \cdot r \cdot c \cdot l^2} \tag{4.15}$$

According to the value of x, there are three cases for the merging segment construction:

- **Case 1:** $0 \leq x \leq 1$

  In this case, the minimum merging cost while achieving zero-skew can be computed. Therefore, the edge length of sub-tree 1 is equal to $x \cdot l$ and the edge length of the sub-tree 2 is equal to $(1 - x) \cdot l$.

- **Case 2:** $x < 0$

  In the case where x is a negative number, it implies that $t_1 > t_2$. In order to achieve zero-skew, the algorithm does not insert a wire from the merging point to sub-tree 1, but only to sub-tree 2, such that the delays are balanced. The inserted wirelength (x_{new}) is computed as:

  $$t_1 = t_2 + \ln \mathbf{2} \cdot r \cdot c \cdot l^2 \cdot x_{new}^2 \implies x_{new} = \sqrt{\frac{t_1 - t_2}{\ln \mathbf{2} \cdot r \cdot c \cdot l^2}} \tag{4.16}$$

  The edge length of sub-tree 1 is equal to 0 and the edge length of the sub-tree is equal to $x_{new} \cdot l$.

- **Case 3:** $x > 1$

  In the case where x is larger than 1, it implies that $t_1 < t_2$. Similarly to case 2, in order to achieve zero-skew, the algorithm does not insert a wire from the merging point to sub-tree 2, but only to sub-tree 1. The inserted wirelength (x_{new}) is computed as:

  $$t_1 + \ln \mathbf{2} \cdot r \cdot c \cdot l^2 \cdot x_{new}^2 = t_2 \implies x_{new} = \sqrt{\frac{t_2 - t_1}{\ln \mathbf{2} \cdot r \cdot c \cdot l^2}} \tag{4.17}$$

  The edge length of sub-tree 1 is equal to $x_{new} \cdot l$ and the edge length of the sub-tree is equal to 0.

The following table 4.1 gathers the cases based on the value of x using the RC Model.

Case 1: $0 \leq x \leq 1$	Case 2: x < 0	Case 3: x > 1
$x = \frac{\ln 2 \cdot r \cdot c \cdot l^2 - t_1 + t_2}{2 \cdot \ln 2 \cdot r \cdot c \cdot l^2}$ $edge\_length_1 = x \cdot l$ $edge\_length_2 = (1 - x) \cdot l$	$x_{new} = \sqrt{\frac{t_1 - t_2}{\ln 2 \cdot r \cdot c \cdot l^2}}$ $edge\_length_1 = 0$ $edge\_length_2 = x_{new} \cdot l$	$x_{new} = \sqrt{\frac{t_2 - t_1}{\ln 2 \cdot r \cdot c \cdot l^2}}$ $edge\_length_1 = x_{new} \cdot l$ $edge\_length_2 = 0$

Table 4.1: Equations for edge length in Bottom Up DME with **RC Model**

Bottom-Up Generation of Merging Segments using Pi-Model

The following equation ($t_{Z_1} + t_1 = t_{Z_2} + t_2$) is applied in order to find the wirelength between the merging point and the sub-tree 1 and 2 using the Pi model. The algorithm solves the delay equations for x as it is shown below:

$$t_{Z_1} = r_1 \cdot (\frac{c_1}{2} + C_1) \text{ and } t_{Z_2} = r_2 \cdot (\frac{c_2}{2} + C_2) \tag{4.18}$$

$$x = \frac{(t_2 - t_1) + \Big(r \cdot l \cdot (C_1 + c \cdot \frac{l}{2})\Big)}{r \cdot l \cdot \Big((c \cdot l) + C_1 + C_2\Big)} \tag{4.19}$$

According to the value of x, there are three cases for the merging segment construction:

- **Case 1:** $0 \leq x \leq 1$

  In this case, the minimum merging cost while achieving zero-skew can be computed. Therefore, the edge length of sub-tree 1 is equal to $x \cdot l$ and the edge length of the sub-tree 2 is equal to $(1 - x) \cdot l$.

- **Case 2:** $x < 0$

  In the case where x is a negative number, it implies that $t_1 > t_2$. In order to achieve zero-skew, the algorithm does not insert a wire from the merging point to sub-tree 1, but only to sub-tree 2. The inserted wirelength (x_{new}) is computed as:

  $$t_1 = t_2 + r \cdot x_{new} \cdot (\frac{1}{2} \cdot c \cdot x_{new} + C_2) \implies x_{new} = \frac{\sqrt{(r \cdot C_2)^2 + 2 \cdot r \cdot c \cdot (t_1 - t_2)} - r \cdot C_2}{r \cdot c} \tag{4.20}$$

  The edge length of sub-tree 1 is equal to 0 and the edge length of the sub-tree is equal to x_{new}.

- **Case 3:** $x > 1$

  In the case where x is larger than 1, it implies that $t_1 < t_2$. In order to achieve zero-skew, the algorithm does not insert a wire from the merging point to sub-tree 2, but only to sub-tree 1. The inserted wirelength is computed as:

  $$t_1 + r \cdot x_{new} \cdot (\frac{1}{2} \cdot c \cdot x_{new} + C_1) = t_2 \implies x_{new} = \frac{\sqrt{(r \cdot C_1)^2 + 2 \cdot r \cdot c \cdot (t_2 - t_1)} - r \cdot C_1}{r \cdot c} \tag{4.21}$$

  The edge length of sub-tree 1 is equal to 0 and the edge length of the sub-tree is equal to x_{new}.

Similarly, the following table 4.2 gathers the cases based on the value of x using the Pi Model.

Case 1: $0 \leq x \leq 1$	Case 2: x $<$ 0	Case 3: x $>$ 1
$x = \frac{(t_2 - t_1) + \Big(r \cdot L \cdot (C_1 + c \cdot \frac{l}{2})\Big)}{r \cdot l \cdot \Big((c \cdot l) + C_1 + C_2\Big)}$	$x_{new} = \frac{\sqrt{(r \cdot C_2)^2 + 2 \cdot r \cdot c \cdot (t_1 - t_2)} - r \cdot C_2}{r \cdot c}$	$x_{new} = \frac{\sqrt{(r \cdot C_1)^2 + 2 \cdot r \cdot c \cdot (t_2 - t_1)} - r \cdot C_1}{r \cdot c}$
$edge\_length_1 = x \cdot l$	$edge\_length_1 = 0$	$edge\_length_1 = x_{new}$
$edge\_length_2 = (1 - x) \cdot l$	$edge\_length_2 = x_{new}$	$edge\_length_2 = 0$

Table 4.2: Equations for edge length in Bottom Up DME with **Pi Model**

Example of Bottom Up Phase using Pi Model

In the following section, the generation of the merging segment of Inner Node 11 is presented for a better understanding of the Bottom-Up Stage of the DME Algorithm using the Pi Model (figure 4.8). As it is shown in figure 4.4, the children of node 11 are FF1 and FF3.

Let $r = 10^3, c = 10^{-12}, C_{FF1} = C_{FF3} = 0.0018pF$ and $l = 11.78$, the Manhattan distance between FF1 and FF3. In this example, $t_{FF1} = t_{FF3} = 0$, since FF1 and FF3 are flipflops, and not inner nodes. Using the initial formula for x from table 4.2, we have the following

$$x = \frac{(t_{FF3} - t_{FF1}) + \Big(r \cdot L \cdot (C_{FF1} + c \cdot \frac{l}{2})\Big)}{r \cdot l \cdot \Big((c \cdot l) + C_{FF1} + C_{FF3}\Big)} \implies x = 0.5$$

Since $0 \leq x \leq 1$, the algorithm uses the formulas for the edge length from <u>Case 1</u> and they are computed as $e_{FF1} = x \cdot l \implies e_{FF1} = 5.89$ and $e_{FF3} = (1-x) \cdot l \implies e_{FF3} = 5.89$. The capacitance seen at Inner Node 11 is calculated

$$C_{11} = C_{FF1} + C_{FF3} + (c \cdot (e_{FF1} + e_{FF3})) \implies C_{11} = 0.0036pF,$$

and the delay at the same node is equal to

$$t_{11} = t_{FF1} + (r \cdot e_{FF1} \cdot (C_{FF1} + \frac{c \cdot e_{FF1}}{2})) \implies t_{11} = 10.5549ns$$

The next step is the construction of the TRRs of FF1 and FF3. Starting from the center of FlipFlop 1, the algorithm creates a rhombus (in case the child is a sink), the TRR mentioned above, with an edge equal to the computed edge length. The same is done for FlipFlop 3, as it is presented into figure 4.9. The intersection of the TRRs is equal to the Merging Segment of the Inner Node 11.

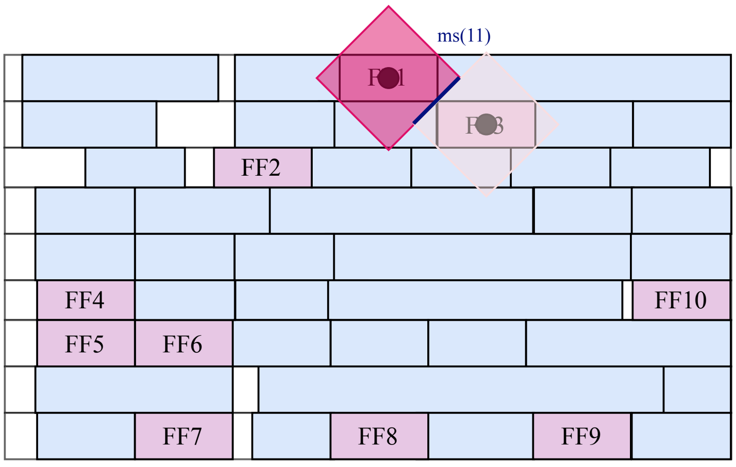


Figure 4.9: Generation of Merging Segment of Inner Node 11

Then, the algorithm traverses the Binary Tree (figure 4.4) in a bottom-up manner and computes the merging segments for the Inner Nodes 7 and 6. The procedure is the same as the computation of the merging segments of the Inner Node 11, and therefore, it is not expanded further. The next merging segment to be computed is the merging segment of the Inner Node 5. The left child of that node is the Inner Node 11 and the right child is the FlipFlop 10 (as can be observed by the Binary Tree representation in figure 4.4).

Initially, the Manhattan Distance between the merging segment and the FlipFlop 10 is calculated. Since the Inner Node 11 is a merging segment, two Manhattan Distances are calculated, with the first one being the one between the upper point of the generated merging segment and the center of FlipFlop 10 and the second one being between the lower point of the generated merging segment and the center of the FlipFlop 10. The minimum distance between these two nodes is equal to $37.01um$.

Then, using the formula defined for x we have:

$$x = \frac{(t_{FF10} - t_{11}) + \Big(r \cdot L \cdot (C_{11} + c \cdot \frac{l}{2})\Big)}{r \cdot l \cdot \Big((c \cdot l) + C_{11} + C_{FF10}\Big)} \implies x = 0.2814$$

Since $x < 1$, then the edges of the children of Inner Node 5 are computed using the formulas from <u>Case 1</u> (table 4.2): $e_{11} = x \cdot l \implies e_{11} = 10.3733$ and $e_{FF10} = (1 - x) \cdot l \implies e_{FF10} = 26.6367$. Then, as the algorithm states, the TRR of each node is constructed. In this case, the left child (Inner Node 11) is a merging segment. In order to construct the TRR, the upper point of the merging segment is expanded by the amount of the computed edge length in the upper and right direction, and the lower point of the merging segment is expanded by the amount of the computed edge length in the lower and left direction, as it is shown in figure 4.10. The TRR of the FlipFlop 10 is constructed as well, and the intersection of both TRRs constitutes the merging segment of Inner Node 5 *(bold blue line in figure 4.10).*

The capacitance seen at Inner Node 5 is equal to

$$C_5 = C_{11} + C_{FF10} + (c \cdot (e_{11} + e_{FF10})) \implies C_5 = 0.0054pF$$

and the delay at the same node is equal to

$$t_5 = t_{FF10} + (r \cdot e_{FF10} \cdot (C_{FF10} + \frac{c \cdot e_{FF10}}{2})) \implies t_{11} = 47.94ns$$

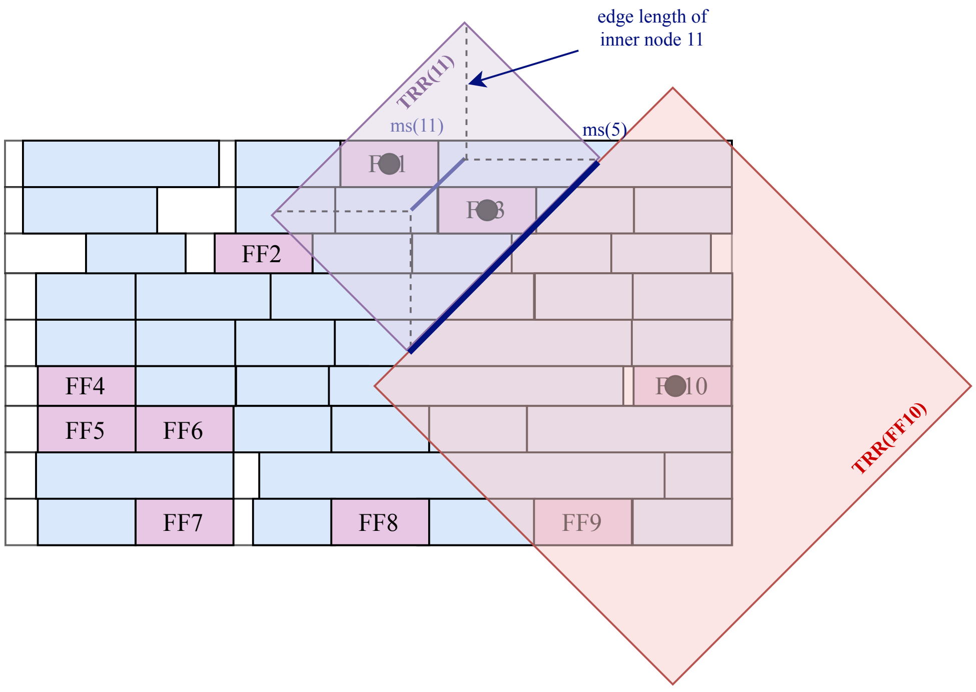


Figure 4.10: Generation of Merging Segment of Inner Node 5

4.2.2 Top-Down Phase: Embedding of Nodes

The second step of the Deferred Merge Embedding (DME) Algorithm, presented in algorithm 3, defines the exact locations of the Inner Nodes in a Top-Down manner. Initially, for the Root Node of the Binary Tree, the algorithm chooses any point of the merging segment of the root node that was generated during the Bottom-Up Phase described in section 4.2.1. If the node is an Inner node, then the exact location depends on its edge length, that was computed during the construction of the merging segment, and the location of its parent node.

Let v be an inner node of the binary tree and p the parent of that node. The goal of this step as it was mentioned, is to determine the exact location of the node v. The algorithm constructs a TRR that has as center/core the exact location of parent node p (the x,y-coordinates of this node was determined in the previous step of the Top-Down Phase) and as edge the edge length of the node v (that was computed during the Bottom-Up Phase). The final position of the node v is the point where the constructed TRR and the merging segment of node v intersect. In the case that there are many points of the intersection, the algorithm chooses the point that is closer to the location of the parent node p. This procedure is continued until the leaf nodes (clock sinks) of the binary tree are reached.

Algorithm 3 Top-Down Phase of Deferred-Merge (DME) Embedding Algorithm

```
for for each internal node v in G (top-down order) do
    if v is the root then
        Choose any $pl(v) \in ms(v)$
    else
        Let p be the parent node of v
        Construct $trr_p$ as follows:
        $core(trr_p) \leftarrow pl(p)$
        $radius(trr_p) \leftarrow |e_v|$
        Choose any $pl(v) \in ms(v) \cap trr_p$
    end if
end for
```

Example of Top Down Phase

The following section, presents the steps while determining the exact location of the inner node 11, for a better understanding of the Top-Down Stage of the DME algorithm. In this step, the final position of the Root Node and Inner Nodes 1-7 have been determined but in figure 4.11 only the final position of Inner Node 5 is shown, since node 5 is the parent node of the node 11.

The algorithm creates a TRR having as core the final position of the parent node 5 and edge length equal to **10.3733** (the edge length of the Inner Node 11 as it was computed in the example of section 4.2.1. Then, the final location of the Inner Node 11 is defined as the point where the TRR and the merging segment of node 11 intersect, as shown in figure 4.11.

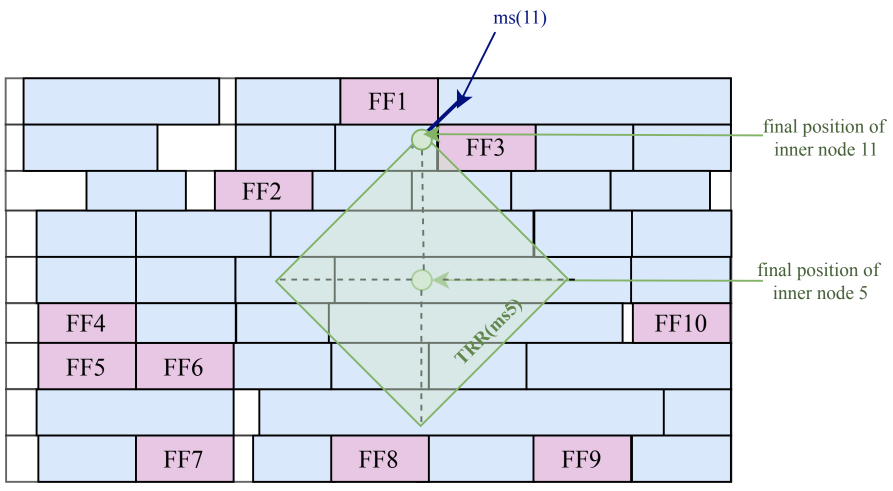

ms(11)
FF1
FF3
FF2
FF4
FF10
FF5
FF6
FF7
FF8
FF9
TRR(ms5)
final position of inner node 11
final position of inner node 5

Figure 4.11: Computation of Final Position of Inner Node 11

4.3 Buffering Algorithm

Buffer Insertion is an effective way to achieve minimum clock signal delay and skew. Having as input a routed clock tree that was generated using the algorithms Method Of Means and Medians (MMM) and Deferred Merge Embedding (DME), the algorithm aims to find the most optimal positions for the buffers to be inserted such that the clock skew and delay are minimized. Moreover, buffer insertion helps to maintain the integrity of the clock signal during its propagation from the clock source to the sinks (sequential elements). The greediest approach is to insert clock buffers at each generated Inner Node. However, the clock tree has the form of a balanced binary tree, resulting in $\mathbf{2} \cdot n - \mathbf{1}$, if we suppose that the binary tree has n leaves. Therefore, the greediest approach can lead to extremely high runtime and over-buffering of the design, introducing a big delay along the paths of the clock tree. Moreover, none of the implemented approaches insert a buffer at the clock root.

Lastly, in both proposed methodologies, after placing the buffers in the selected positions, the Legalization step is performed. This step is very important, since the generated locations of the inner nodes may overlap with already placed cells. During Clock Tree Synthesis, the clock buffers have higher priority, therefore the cells that may overlap with the buffers, are placed into new locations with the help of the Legalizer.

4.3.1 First Buffering Approach: Buffer Insertion at Clock Sinks

The first implemented buffer insertion algorithm inserts one buffer between the clock source and each clock sink. This approach maintains an equal number of buffers between each clock-to-sink path (one buffer) and therefore inserts the minimum clock delay. Using the generated binary tree from the Method of Means and Medians in figure 4.4, the following figure (figure 4.12) illustrates the implemented buffer insertion. However, this approach can also lead to over-buffering of the design in the case where the number of sequential cells is extremely large because the number of inserted buffers is equal to the clock sinks. Moreover, it is important to mention, that this algorithm does not insert any buffer in the paths that consist of the fanout of the Integrated Clock Gating Cells. The delay from the Integrated Clock Gating cells to their fanout highly depends on the delay imported from their between interconnect.

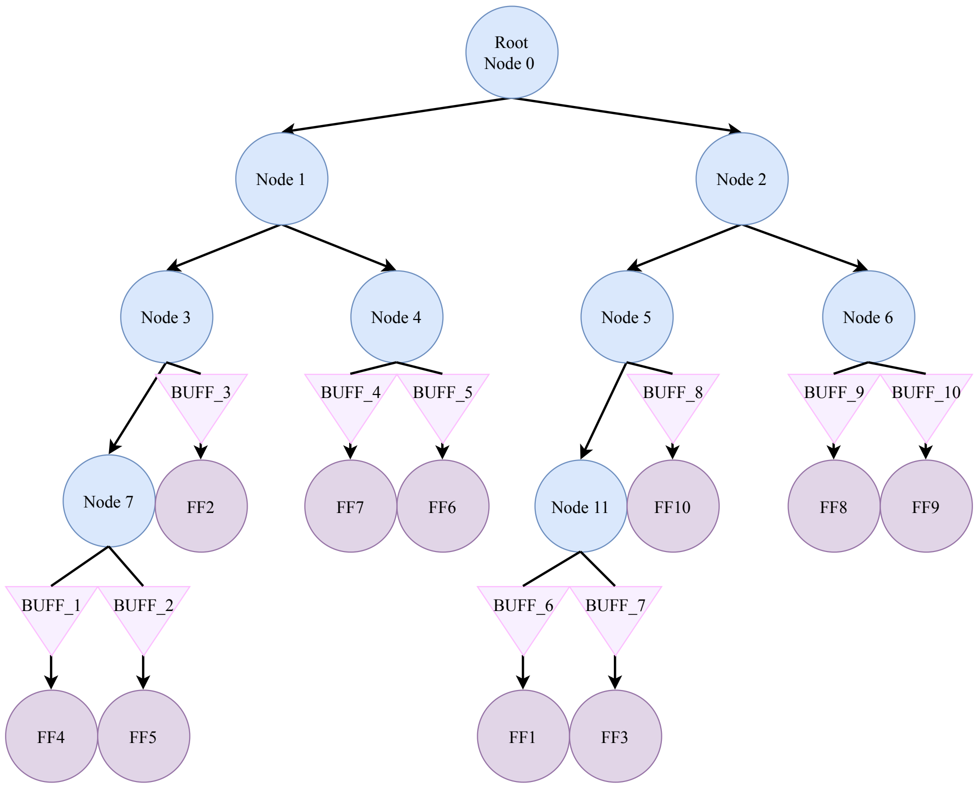


Figure 4.12: Example using the first implemeted buffering approach

4.3.2 Second Buffering Approach: Buffer Insertion at Selected Levels of the Clock Tree

The second approach is based on a heuristic found through a number of experiments. As it was mentioned, the buffer insertion at each sink or at each inner node negatively affects the delay of the clock tree and the execution time of the Buffer Insertion. Therefore, a new heuristic has been developed in order to achieve the requirements of a balanced clock tree. Using the generated binary tree from the Method of Means and Medians in figure 4.4, the following figure (figure 4.13) illustrates the implemented buffer insertion.

The heuristic inserts buffers, in a bottom-up approach, at the inner nodes that were generated from the Method of Means and Medians that have a level equal to max level $-\mathbf{3}$ and max level $-\mathbf{4}$, where max level is the maximum level of the binary tree. Implementing this heuristic, it is found that the maximum fanout of a buffer is 8. For example, let a design with 32 nodes generate a Binary Tree with 63 total nodes and the maximum level of the tree will be equal to 5. Based on the heuristic, the buffers will be inserted into the inner nodes that are on level 3 and on level 2.

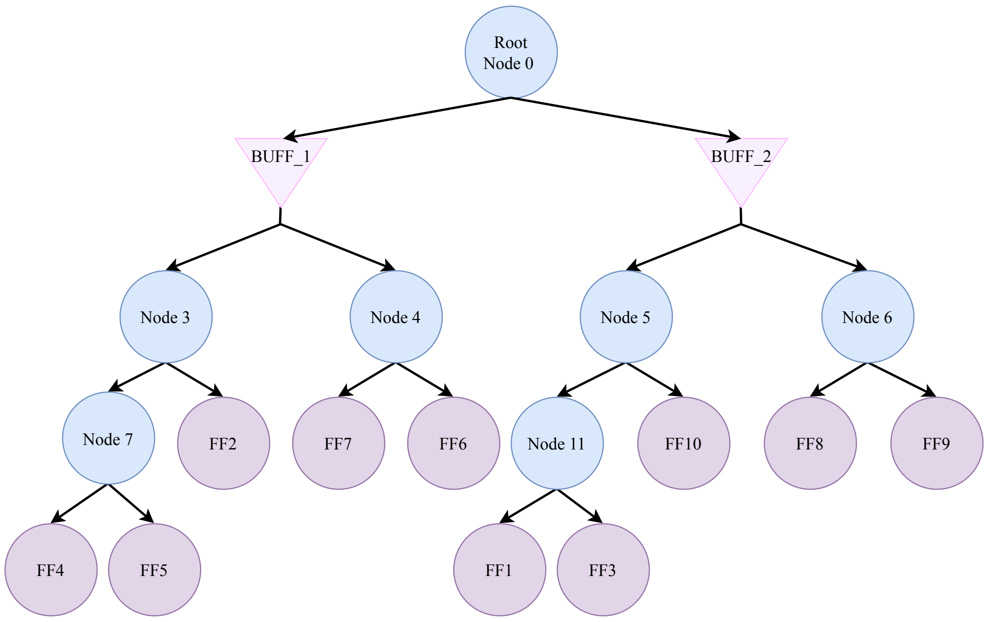


Figure 4.13: Example using the second buffering approach

Moreover, observing the segment of Binary Tree in figure 4.14 presents the main reason why the algorithm has chosen the specific levels of the binary tree for buffer insertion. Supposing that the circuit in figure 4.14 represents a part of the binary tree in figure 4.4, and more specifically the inner nodes Node 3 and Node 7, along with the children of these nodes ($FF\mathbf{4}$, $FF\mathbf{5}$ and $FF\mathbf{2}$), we try inserting buffer at the last two levels of the binary tree. However, since the sinks (or Flip Flops in this example) do not belong on the same level of the tree, it is expected that there will be a noticeable difference in the Arrival Time (AT) of the clock signal to each one of the Flip Flops. Moreover, the number of buffers will not be always equal in all of the clock source to Flip Flop paths. The path from the clock source to $FF\mathbf{2}$ will have only $\mathbf{1}$ buffer, while the path from the clock source to $FF\mathbf{4}$ and the path to $FF\mathbf{5}$, will have $\mathbf{2}$ buffers.

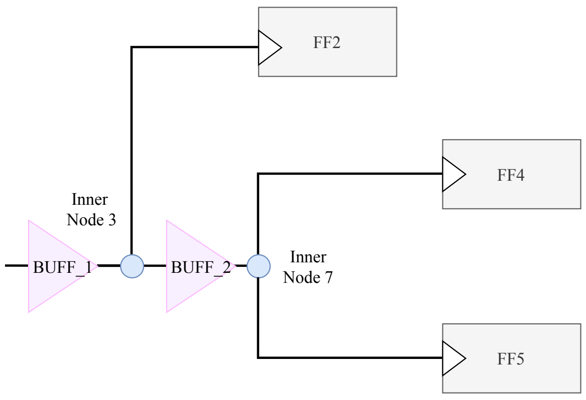


Figure 4.14: buffering example

4.4 Clock Tree Synthesis in the presence of Integrated Clock Gating Cells

Integrated Clock Gating Cells are very common in designs, especially when minimum power is one of the main requirements of the circuit. However, zero-skew is more difficult to achieve during Clock Tree Synthesis. As it was mentioned in section 2.2, integrated clock gating cells disable the clock signal in some areas of the design when these areas are not in use since this helps highly reduce the switching activity of the sequential cells. The biggest problem during the procedure of Clock Tree Synthesis arises when these areas are disabled and the Integrated Clock Gating cells are treated as clock sinks. Clock Tree Synthesis must achieve minimum delay difference at both integrated clock gating cells and all clock sinks, such that the skew is very low whether the clock gating technique is enabled or disabled.

The proposed methodology is presented in the following schematic figure 4.15. Initially, the Integrated clock gating cells are located in the design and they are marked as clock sinks (along with the rest sequential elements of the design). The fanout of the Integrated Clock Gating cells is located also and ignored during the Clock Tree Synthesis procedure. This is done because the main goal is to achieve minimum or zero skew between the Integrated Clock Gating Cells and the rest sequential elements that do not have a gated clock signal as input. The next steps of the methodology remain the same as they are mentioned in section 4.1.1 (Method of Means and Medians), section 4.2 (Deferred Merge Embedding) and section 4.3 (Buffering Algorithm) when Integrated Clock Gating cells are not present in the design. Moreover, it is worth mentioning that no buffers are inserted into the fanout of the Integrated Clock Gating cells because the main goal is to achieve the minimum delay to these cells.

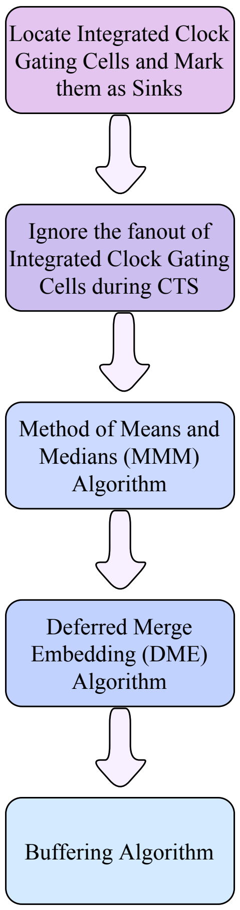

Locate Integrated Clock Gating Cells and Mark them as Sinks
Ignore the fanout of Integrated Clock Gating Cells during CTS
Method of Means and Medians (MMM) Algorithm
Deferred Merge Embedding (DME) Algorithm
Buffering Algorithm

Figure 4.15: Proposed Methodology when ICG cells are present during Clock Tree Synthesis

Chapter 5

Experimental Evaluation

The following Chapter represents the experimental results after implementing the Proposed Methodology mentioned in chapter 4, in an internal Place and Route (PnR) Tool with an integrated Static Timing Analysis (STA) Engine.

Initially, table 5.1 represents the benchmarks that were used in order to evaluate the validity and accuracy of the algorithms. The netlist technology for the designs AES, APB, and PID is $130um$ IHP. A wide variety of experiments has been performed using all the possible combinations, regarding the first cut of the Method Of Means and Medians algorithm and the type of interconnect models in the Deferred Merge Embedding Algorithm.

In terms of comparison, reference clock tree solutions have been constructed using TritonCTS, an open-source tool that performs Clock Tree Synthesis [8], and an Industrial Tool. The table 5.2 and table 5.3 show the total number of inserted buffers and the skew of the designs, after performing the proposed methodology in chapter 4, as well as the produced results using the OpenSource tool TritonCTS [8] and the Industrial tool.

Testcases	Number of Total Cells	Number of Flipflops	Number of ICG cells
AES	8057	670	-
APB	1021	910	-
PID	5301	397	-
SPI	1183	229	23

Table 5.1: Description of the Testcases

5.0.1 Experimental Results with RC-Model

The table 5.2 shows the total number of inserted buffers and the skew of the designs, after performing the proposed methodology in chapter 4, as well as the produced results using the OpenSource tool TritonCTS [8] and Industrial tool. More specifically, regarding the first buffering approach, the algorithm inserts buffers before each clock sink. On the other hand, the second buffering approach inserts buffers at specific levels of the binary tree for better overall results. The experiments of the following table were performed using the RC-model for the wire representation mentioned in section 4.2.1. Also, regarding the

Method of Means and Medians, the first cut/division of the sequential elements of the design was done in vertical order.

Testcases	Tool	Skew (ns)	Number of Buffers
AES	Prop. Methodology (1st Buff. Approach)	0.010047	670
	Prop. Methodology (2nd Buff. Approach)	0.029510	192
	TritonCTS	0.201001	127
	Industrial Tool	0.009323	104
APB	Prop. Methodology (1st Buff. Approach)	0.010708	910
	Proposed Methodology (2nd Buff. Approach)	0.049292	192
	TritonCTS	0.178750	255
	Industrial Tool	0.008716	114
PID	Prop. Methodology (1st Buff. Approach)	0.007160	397
	Prop. Methodology (2nd Buff. Approach)	0.016215	96
	TritonCTS	0.216431	63
	Industrial Tool	0.004644	46
SPI	Prop. Methodology (1st Buff. Approach)	0.046756	62
	Prop. Methodology (2nd Buff. Approach)	0.055578	12
	TritonCTS	0.19265	78
	Industrial Tool	0.02328	13

Table 5.2: Comparison between the Implemented Algorithm, TritonCTS and Industrial Tool (RC-Model)

As we can observe by the table, the implemented Clock Tree Synthesis algorithm combined with the 1st Buffering Approach described in section 4.3.1 inserts a very large number of buffers, compared with the other approaches. This can be explained because the first approach inserts as many buffers as the sinks. However, observing the results, the proposed methodologies insert to all of the designs more buffers than the TritonCTS and the Industrial Tool. Studying further the methodologies that both tools use, it is clear that they use a more developed heuristic for buffer insertion than the proposed ones. Moreover, both Clock Tree Synthesis methodologies perform many optimizations to the designs during the procedure.

Design: AES

The following pictures in figure 5.1 represent the design AES (with 670 FlipFlops). In the first picture (figure 5.1a), the highlighted cells correspond to the placement of the sequential elements of the design. The same placement is used as input for all three of the tools that were used to produce the results in table 5.2.

The second picture (figure 5.1b) depicts the placement of the buffers using the Proposed Methodology with the second Buffering Approach. The main reason why the First Buffering approach is not included in the figures is that this algorithm over-buffers the design and does not produce quality results. Moreover, the placement of the buffers is right next to the placement of the sinks.

In the third picture, figure 5.1c, the highlighted cells are the buffers that the TritonCTS has selected to insert into the design. Comparing the figure 5.1b with figure 5.1c, the placement of the buffers is similar, regarding that they are placed across the design with the only difference that TritonCTS inserts fewer buffers than the Proposed Methodology.

Lastly, the figure in figure 5.1d represents the buffers that were placed performing Clock Tree Synthesis using the Industrial Tool. The resulting layout is much different than the Proposed Methodology and the OpenSource Tool. The Industrial Tool changes the layout after the Buffer Placement, since some positions of the new buffers may overlap with already placed cells. In this case, the tool performs an incremental placement flow, such that the placement of the buffers has a higher priority than the placement of the already placed cells.

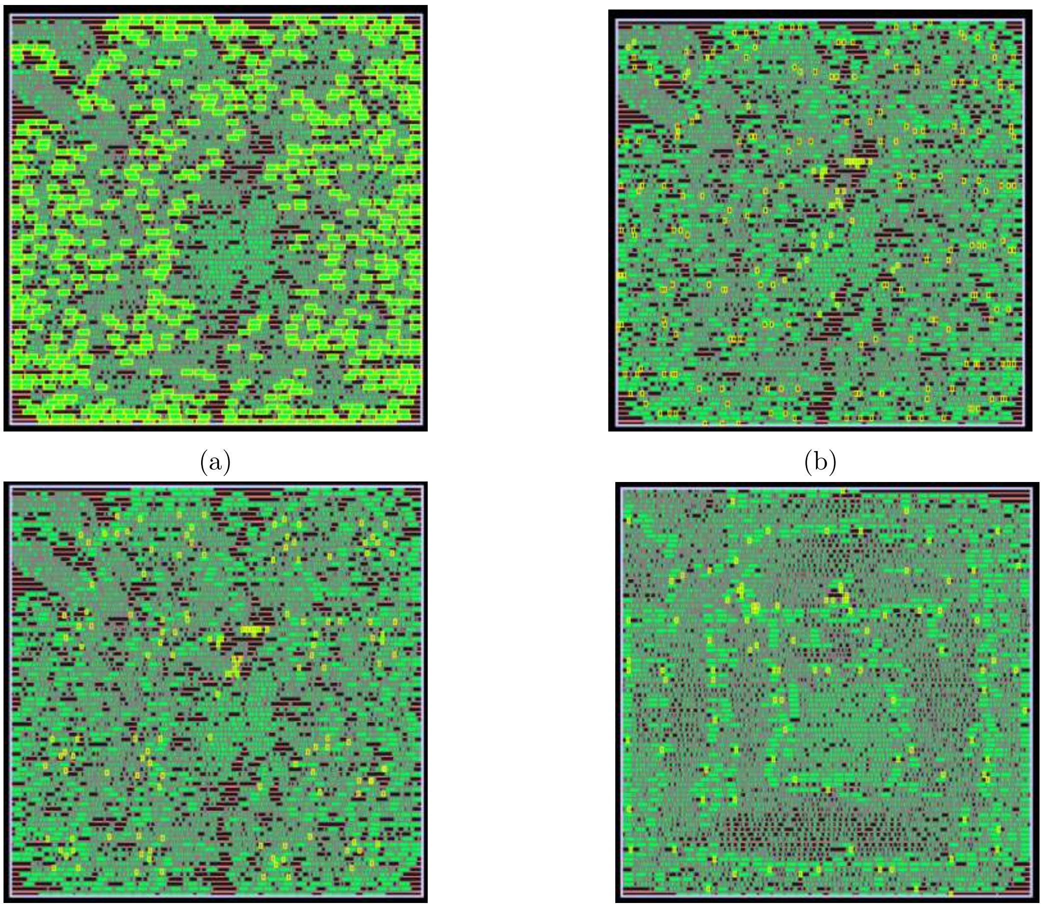


Figure 5.1: Screenshots from the Proposed Methodology regarding the AES design. (a) Sequential Elements of the design, (b) Placement of the Buffers generated by the Proposed Methodology, (c) Placement of the Buffers generated by TritonCTS, and (d) Placement of the Buffers generated by Industrial Tool

Design: APB

The layouts in figure 5.2 present the design APB that contains 1021 total cells, from which 910 are sequential cells (or Flipflops). In the first figure (figure 5.2a), the cells that are highlighted with yellow are the clock sinks (the sequential cells). The next three figures (figure 5.2b, figure 5.2c and figure 5.2d) represent the placement of the buffers using the proposed algorithm, the OpenSource tool TritonCTS and a Commercial Tool, accordingly. Comparing the Buffer Insertion performed by the Proposed Methodology (figure 5.2b) with the Buffer Insertion performed by TritonCTS (figure 5.2c), the number of buffers in the first tool is greater than the second one. Moreover, the placement of the layout changes only in the third case, using the commercial tool, since more optimizations are performed to achieve the desired requirements of the design. Also, observing the table 5.2, the number of inserted buffers using the proposed approach is 9x the buffers that the industrial tool uses.

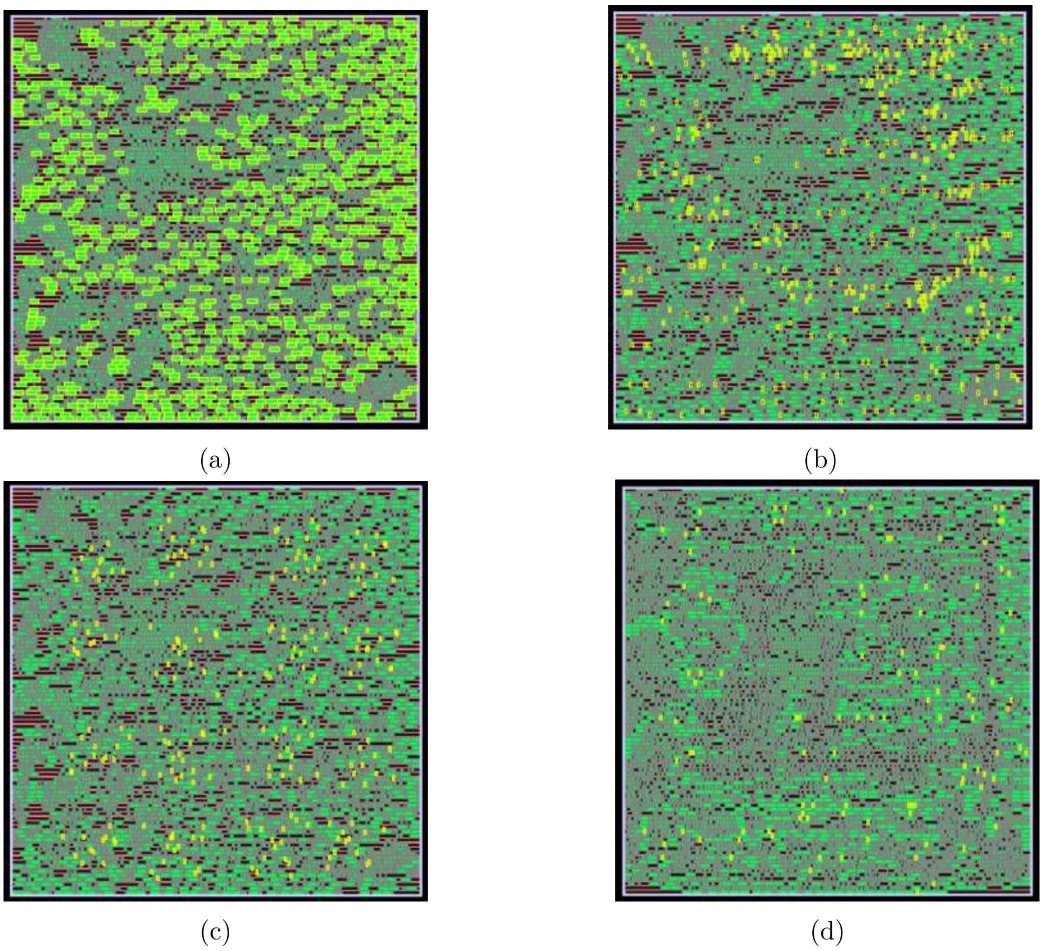

(a) (b) (c) (d)

Figure 5.2: Screenshots from the Proposed Methodology regarding the APB design. (a) Sequential Elements of the design, (b) Placement of the Buffers generated by the Proposed Methodology, (c) Placement of the Buffers generated by TritonCTS and (d) Placement of the Buffers generated by Industrial Tool

Design: PID

Similarly, the figures in figure 5.3 depict the design PID, which contains 397 sequential cells considered as clock sinks during Clock Tree Synthesis. The highlighted cells in the first layout (figure 5.3a) correspond to the 397 sequential cells of the design. The following three layouts (figure 5.3b, figure 5.3c and figure 5.3d) represent the buffer placement using the methodologies of the three different tools that were mentioned earlier. The results from table 5.2 indicate that the first buffering approach used in our Proposed Methodology offers better results regarding the Skew compared to the second buffering approach and the solution provided by the OpenSource tool, TritonCTS. However, this approach, as it was mentioned above, over-buffers the design and has a high execution time, due to continuous traversals of the Binary Tree. Moreover, the second buffering approach of the Proposed Methodology is comparable to the TritonCTS solution, since the number of inserted buffers is more rational while generating a more optimal skew.

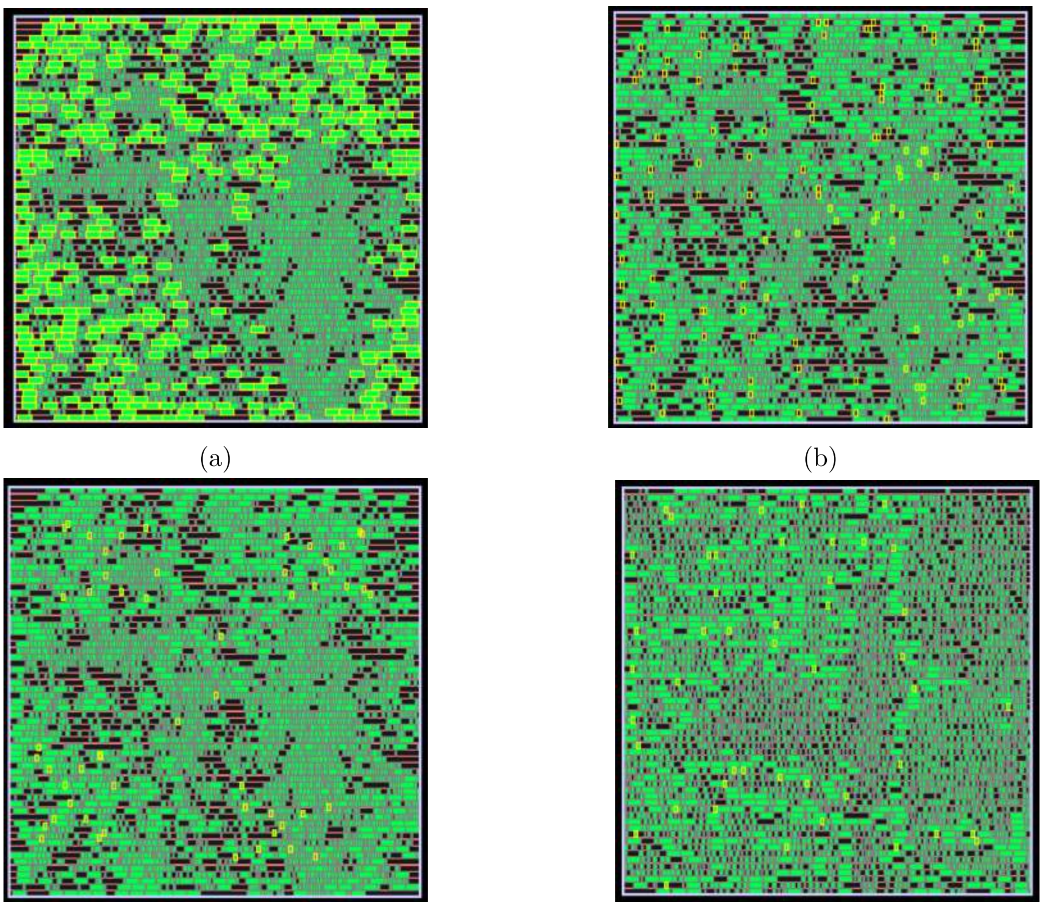


Figure 5.3: Screenshots from the Proposed Methodology regarding the PID design. (a) Sequential Elements of the design, (b) Placement of the Buffers generated by the Proposed Methodology, (c) Placement of the Buffers generated by TritonCTS and (d) Placement of the Buffers generated by Industrial Tool

Design: SPI

The following pictures in figure 5.4 depict the SPI design that consists of 1183 total cells, from which 229 are flipflops and 23 are Integrated Clock Gating cells. The flow proposed in section 4.4 will locate the flipflops and the ICG cells including their fanout.The next step is to set the fanout of the ICG as "ignore cells", since we are researching the case in which the ICG cell prevents the propagation of the clock signal. Having completed this analysis, the algorithm marks as clock sink, only 62 sequential cells, presented in figure 5.4a. The following three screenshots depict the buffer placement performed by the Proposed Methodology (figure 5.4b), the OpenSource tool TritonCTS (figure 5.4c) and the Industrial Tool (figure 5.4d). It can be observed that the buffers are evenly placed regarding the placement of the clock sinks.

Regarding the results in table 5.2, the Proposed Methodology produces in both buffering approaches a clock network with less skew than the OpenSource tool. Moreover, the implemented algorithm in the second buffering approach inserts to the design similar amount of buffers compared to the Industrial Tool while achieving worst but comparable skew.

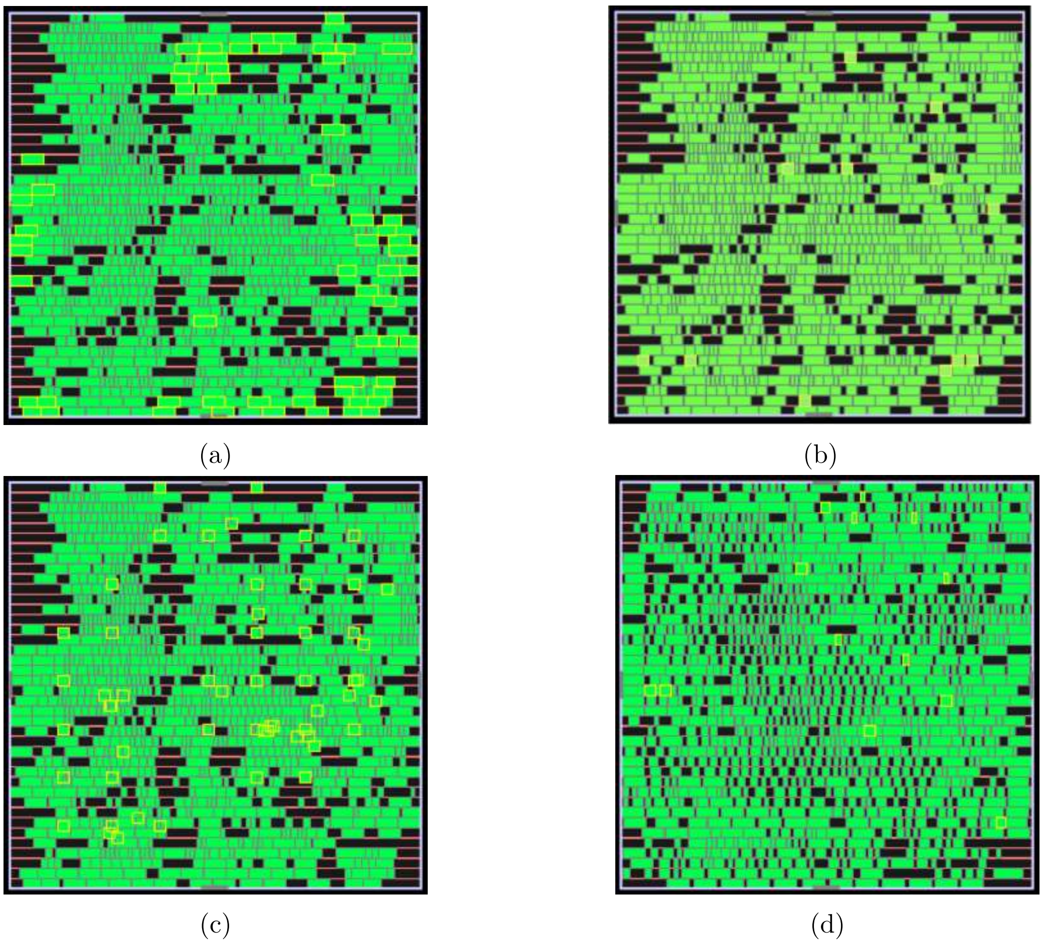


Figure 5.4: Screenshots from the Proposed Methodology regarding the SPI design. (a) Sequential Elements of the design, (b) Placement of the Buffers generated by the Proposed Methodology, (c) Placement of the Buffers generated by TritonCTS and (d) Placement of the Buffers generated by Industrial Tool

5.0.2 Experimental Results with Pi-Model

The experiments of the following table were performed using the Pi-model for the wire representation mentioned in section 4.2.1. Also, regarding the Method of Means and Medians, the first cut/division of the sequential elements of the design was done in vertical order.

Testcases	Tool	Skew (ns)	Number of Buffers
AES	Prop. Methodology (1st Buff. Approach)	0.1294	670
	Prop. Methodology (2nd Buff. Approach)	0.8963	192
	TritonCTS	0.201001	127
	Industrial Tool	0.009323	104
APB	Prop. Methodology (1st Buff. Approach)	0.1475	910
	Prop. Methodology (2nd Buff. Approach)	0.9743	192
	TritonCTS	0.17875	255
	Industrial Tool	0.008716	114
PID	Prop. Methodology (1st Buff. Approach)	0.09356	397
	Prop. Methodology (2nd Buff. Approach)	0.06438	96
	TritonCTS	0.216431	63
	Industrial Tool	0.004644	46
SPI	Prop. Methodology (1st Buff. Approach)	0.068643	62
	Prop. Methodology (2nd Buff. Approach)	0.085379	12
	TritonCTS	0.19265	78
	Industrial Tool	0.02328	13

Table 5.3: Comparison between the Implemented Algorithm, TritonCTS and Industrial Tool (Pi-Model)

Observing the resulting skew in table 5.2 and table 5.3, we can conclude that the results in the second case, using the Pi-Model representation of the wire resistance are worse. Even though the Model is considered to be more accurate than the RC model, this result can be due to the inaccurate formula of the capacitance of the inner node. The algorithm, after computing the edge length of the inner node (as it is shown in the example in section 4.2.1) uses the following formula for computing the capacitance at the inner node: $C_{node} = C_1 + C_2 + c \cdot (e_1 + e_2)$, where C_1 and C_2 correspond to the capacitance of the children nodes and e_1 and e_2 are the computed edge lengths of the nodes. However, this formula does not provide high accuracy in the values of the capacitance.

Testcases	Total Clock Sinks	Skew Difference (ns)	Number of Buffers Difference
AES	670	-0.866790	0
APB	910	-0.925008	0
PID	397	-0.048165	0
SPI	229	-0.029801	0

Table 5.4: Comparison between the RC-Model and Pi-Model using the second buffering approach

The table 5.4 shows the comparison between the two wire models while using the second buffering approach presented in section 4.3.2. The Skew Difference (ns) is defined as Skew using RC-Model – Skew using Pi-Model. As it is easily observed in the implemented algorithm the RC-Model offers a better accuracy, regarding the skew. The number of inserted buffers is the same in all of the testcases since this mainly depends on the generated binary tree by the Method of Means and Medians algorithm. Moreover, there is a correlation between the total number of clock sinks and the skew difference. The first two testcases (AES and APB) have a large skew difference while the designs PID and SPI have a smaller. This can be justified by the number of total clock sinks. As it was mentioned above, the disadvantage of the Pi-model is the inaccurate formula for the capacitance of the inner node. Circuits with a large number of sequential cells will lead to a bigger binary tree, therefore there are more inner nodes for which the capacitance must be calculated. Therefore, the error is increased based on the number of total clock sinks, regarding the Pi-Model.

5.0.3 Experimental Results with Different Initial First Cut in MMM

The results derived from the Deferred Merge Embedding algorithm highly depend on the input binary tree. In this thesis, the Binary Tree is generated using the algorithm Method of Means and Medians. The measurements presented in the above section are produced with the first division of the sequential cells done in vertical order (using the x-coordinates of the cells). However, some experiments have been implemented in order to determine how the first "cut" or division can affect the skew of each design. Therefore, the table 5.5 depicts the algorithm's results, dividing the clock sinks in horizontal order, based on their y-coordinates. As we can observe, the implemented algorithm still produces better results than the Opensource tool. However, due to the many optimizations performed, the Industrial tool generates the most optimal skew.

Testcases	Tool	Skew (ns)	Number of Buffers
AES	Prop. Methodology (1st Buff. Approach)	0.015435	670
	Prop. Methodology (2nd Buff. Approach)	0.03622	192
	TritonCTS	0.201001	127
	Industrial Tool	0.009323	104
APB	Prop. Methodology (1st Buff. Approach)	0.011266	910
	Prop. Methodology (2nd Buff. Approach)	0.042458	192
	TritonCTS	0.17875	255
	Industrial Tool	0.008716	114
PID	Prop. Methodology (1st Buff. Approach)	0.007150	397
	Prop. Methodology (2nd Buff. Approach)	0.015832	96
	TritonCTS	0.216431	63
	Industrial Tool	0.004644	46
SPI	Prop. Methodology (1st Buff. Approach)	0.026100	62
	Prop. Methodology (2nd Buff. Approach)	0.053146	12
	TritonCTS	0.19265	78
	Industrial Tool	0.02328	13

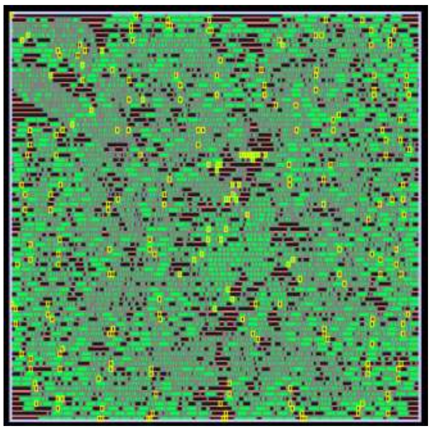
Table 5.5: Comparison between the Implemented Algorithm (with different first cut), TritonCTS, and Industrial Tool

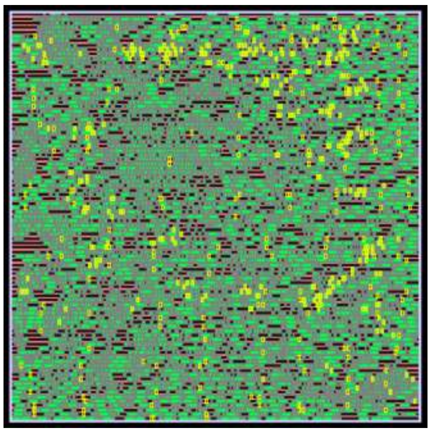
The table 5.6 compares the Skew of the benchmarks, based on the first division in the Method of Means and Medians Algorithm. The Skew Difference (ns) is defined as (Skew generated when the first cut is done based on the x-coordinates of the sinks) - (Skew generated when the first cut is done based on the y-coordinates of the sinks). The results signify that the second approach (division of the sinks based on the y-coordinates) produces a smaller skew regarding the designs APB, PID, and SPI, while the first approach produces better results for the design AES. This comparison indicates that the first cut can highly affect the quality of the results of the implemented algorithm and that the first division of the circuits depends on the placement of the cells. The comparison using the Pi-Model has been skipped, since due to the inaccurate formula of the capacitance, the results are not comparable.

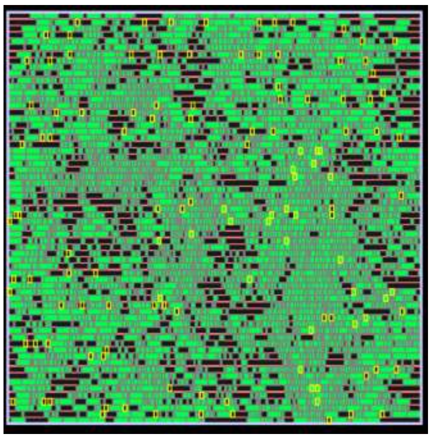
Testcases	Total Clock Sinks	Skew Difference (ns)
AES	670	-0.006710
APB	910	0.006834
PID	397	0.000383
SPI	62	0.002432

Table 5.6: Comparison between the First cut of MMM using the RC-Model and the second buffering approach

Moreover, the placement of the Buffers generated by the Proposed Methodology for each testcase is presented in figure 5.5. In all cases, the buffers are evenly distributed in the layout area. As it can be noted by the table 5.5, the number of inserted buffers is the same in all cases, since only the direction of the first division/"cut" is different and not the number of sequential cells that are considered as clock sinks for the Clock Tree Synthesis.


(a)


(b)


(c)

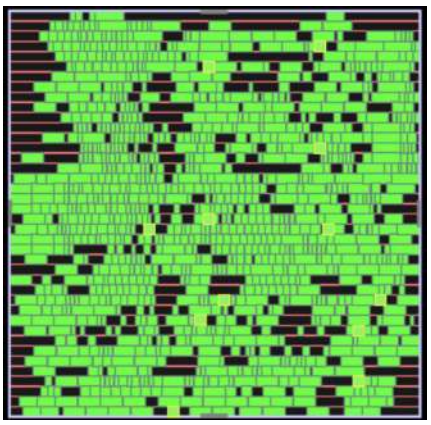

(d)

Figure 5.5: Buffer Placement using the Proposed Methodology with the first division in the MMM algorithm being horizontal (a) Buffer Placement of the AES design, (b) Buffer Placement of the APB design (c) Buffer Placement of the PID design, and (d) Buffer Placement of the SPI design

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Concluding, Clock Tree Synthesis is a very complex problem and there are many optimizations that can be applied prior, during, or after Clock Tree Synthesis. This thesis proposes a three-step methodology, that aims to achieve a balanced clock network in the presence of Integrated Clock Gating cells. The first step of the methodology consists of the generation of the binary tree, with the leaves being the clock sinks, using the algorithm of Method of Means and Medians. The second step aims to identify the final positions of the inner nodes of the binary tree with the help of the algorithm Deferred Merge Embedding and the last step uses two different heuristics to insert buffers to the design.

The results presented in Chapter 5 seem promising enough, since compared to other OpenSource Tools, our approach is proved to have better results regarding the Skew of the design. Moreover, two interconnect models have been used for the estimation of the wire delays (the RC model and the Pi model) for a wider variety of results. Additionally, some variations of the first implemented algorithm, Method of Means and Medians, have been developed that aim to acquire better results regarding the skew of the designs. However, the proposed methodology compared to the Industrial Tool, can indicate the lack of pre-CTS and post-CT optimizations, which are considered as future work. The following section presents some of the future work that should be implemented in order to produce even better results.

6.2 Future Work

The first way to improve further the results of the proposed flow is to extend the algorithm of Method of Means and Medians, such that using a heuristic the best first cut or "division" is performed. As we have observed in section 5.0.3, the direction of the first division of the sinks can highly affect the final skew of the design.

Another optimization that could be done is to extend the algorithm of the Deferred Merge Embedding, such that the finalized positions of the inner nodes are obstacle aware. In the implementation of this thesis, as it was mentioned, the step of Detailed Placement was performed when the buffer location overlapped with an already placed cell. This extension of the algorithm can help reduce the run-time and offer at the same time better results. Moreover, regarding the Deferred Merge Embedding, more research should be performed considering the formula for the capacitance of the inner node, using the Pi-model representation as it was mentioned in section 5.0.2.

Moreover, the heuristics used in the buffering approach could be developed more in order to consider other parameters too, such as maximum capacitance, maximum fanout, and slew. Also, a new heuristic should be introduced to decide the type and the size of the clock buffer that will be inserted during the Clock Tree Synthesis. Lastly, regarding the Integrated Clock Gating cells, there are many optimizations that can be performed. The most important should be a placement algorithm, that based on the location of the ICG cell, decides the locations of the sequential cells that belong to its fanout, such that their distance is minimal. This methodology can assure minimum delay from the ICG cell to its fanout, without affecting the other cells.

Appendix A

Acronyms

EDA Electronic Design Automation

IC Integrated Circuit

VLSI Very Large Scale Integration

ICG Integrated Clock Gating

GDSII Graphic Design System II

GP Global Placement

LG Legalization

DP Detailed Placement

CTS Clock Tree Synthesis

FF Flip-Flop

QoR Quality-of-Results

STA Static Timing Analysis

AT Arrival Time

RAT Required Arrival Time

OCV On Chip Variation

MMM Method of Means and Medians

DME Deferred Merge Embedding

MS Merging Segment

TRR Tilted Rectangular Region

PnR Place and Route

Bibliography

[1] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer, Eds., *Electronic design automation for IC implementation, circuit design, and process technology.* CRC Press, Feb. 2017.

[2] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[3] N. Maheshwari and S. S. Sapatnekar, "Timing analysis and optimization of sequential circuits," 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:58186380

[4] C. Piguet, *Low-power Electronics Design.* CRC, 2004. [Online]. Available: https://books.google.gr/books?id=Zl75swEACAAJ

[5] F. Emnett and M. M. Biegel, "Power reduction through rtl clock gating," 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:16164571

[6] J. Cong, A. Kahng, and G. Robins, "Matching-based methods for high-performance clock routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 8, pp. 1157–1169, 1993.

[7] K. Han, A. B. Kahng, and J. Li, "Optimal generalized h-tree topology and buffering for high-performance and low-power clock distribution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 2, pp. 478–491, 2020.

[8] "TritonCTS." [Online]. Available: https://github.com/The-OpenROAD-Project/TritonCTS

[9] X. Jin and J. Han, "K-means clustering," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Springer US, pp. 563–564. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_425

[10] W. Liu, C. Sitik, E. Salman, B. Taskin, S. Sundareswaran, and B. Huang, "Slects: Slew-driven clock tree synthesis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 4, pp. 864–874, 2019.

[11] M. Liu, Z. Zhang, W. Sun, and D. Wang, "Obstacle-aware symmetrical clock tree construction," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 515–518.

[12] I.-M. Liu, T.-L. Chou, A. Aziz, and D. F. Wong, "Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion," in *Proceedings of the*

2000 International Symposium on Physical Design, ser. ISPD '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 33–38. [Online]. Available: https://doi.org/10.1145/332357.332370

[13] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, "Clock routing for high-performance ics," in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, ser. DAC '90. New York, NY, USA: Association for Computing Machinery, 1991, p. 573–579. [Online]. Available: https://doi.org/10.1145/123186.123406

[14] K. Boese and A. Kahng, "Zero-skew clock routing trees with minimum wirelength," in *[1992] Proceedings. Fifth Annual IEEE International ASIC Conference and Exhibit*, 1992, pp. 17–21.