

Electronic Theses and Dissertations, 2020-

2021

Synthesis Methodologies for Robust and Reconfigurable Clock Networks

Necati Uysal
University of Central Florida

 Part of the [Computer and Systems Architecture Commons](#)
Find similar works at: <https://stars.library.ucf.edu/etd2020>
University of Central Florida Libraries <http://library.ucf.edu>

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Uysal, Necati, "Synthesis Methodologies for Robust and Reconfigurable Clock Networks" (2021).
Electronic Theses and Dissertations, 2020-. 941.
<https://stars.library.ucf.edu/etd2020/941>

SYNTHESIS METHODOLOGIES FOR ROBUST AND RECONFIGURABLE CLOCK
NETWORKS

by

NECATI UYSAL

M.S. University of Central Florida, 2017

B.S. University of Gaziantep, 2013

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2021

Major Professor: Rickard Ewetz

© 2021 Necati Uysal

ABSTRACT

In today's aggressively scaled technology nodes, billions of transistors are packaged into a single integrated circuit. Electronic Design Automation (EDA) tools are needed to automatically assemble the transistors into a functioning system. One of the most important design steps in the physical synthesis is the design of the clock network. The clock network delivers a synchronizing clock signal to each sequential element. The clock signal is required to be delivered meeting timing constraints under variations and in multiple operating modes. Synthesizing such clock networks is becoming increasingly difficult with the complex power management methodologies and severe manufacturing variations. Clock network synthesis is an important problem because it has a direct impact on the functional correctness, the maximum operating frequency, and the overall power consumption of each synchronous integrated circuit.

In this dissertation, we proposed synthesis methodologies for robust and reconfigurable clock networks. We have made three contributions to this topic. First, we have proposed a clock network optimization framework that can achieve better timing quality than previous frameworks. Our proposed framework improves timing quality by reducing the propagation delay on critical paths in a clock network using buffer sizing and layer assignment. Second, we have proposed a clock tree synthesis methodology that integrates the clock tree synthesis with the clock tree optimization. The methodology improves timing quality by avoiding to synthesize clock trees with topologies that are sensitive to variations. Third, we have proposed a clock network that can reconfigure the topology based on the active mode of operation. Lastly, we conclude the dissertation with future research directions.

This dissertation is first dedicated to my family and then to Bekir Yilmaz, a senior member of my family whom I have lost during my Ph.D. studies.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Rickard Ewetz. His guidance through my research journey with his expertise and knowledge was invaluable. In addition, I am grateful for his assistance and patience when I encounter difficulties. I would also like to extend my gratitude to Dr. Ronald F. DeMara, Dr. Mingjie Lin, Dr. Murat Yuksel, and Dr. Damian Dechev for their support, feedback, and their time to serve in my doctoral committee.

I would like to thank my friend, Baogang Zhang for his support during my PhD studies. Baogang was always there for me when I needed help. It was a great pleasure for me to work with him in the same research lab.

I would like to thank my dear friends, Ahmet Uzun, Ender Egri, Nazar Emirov, and Yunus Uzun. Without their support and advice, it would be really difficult for me to tackle many problems that I have faced in my life.

I would like to express my endless love to my parents, Turgut and Ayse Uysal. I am grateful to them for their devotion and sacrifices in providing a good life for me. I would also like to thank my dear sister and brother, Ayse Gul and Huseyin, for their continuous support and encouragement throughout my entire life.

I would like to thank the ECE department at UCF and NSF (CCF-1755825, CNS-1908471, and CNS-2008339 grants) for providing financial support during my Ph.D. studies.

TABLE OF CONTENTS

| | |
|---|----|
| LIST OF FIGURES | xi |
| LIST OF TABLES | xv |
| CHAPTER 1: INTRODUCTION | 1 |
| CHAPTER 2: PRELIMINARIES | 3 |
| 2.1 Timing constraints | 3 |
| 2.2 Variations in the timing | 4 |
| 2.3 Clock networks with different topologies | 5 |
| CHAPTER 3: REVIEW OF PREVIOUS WORKS | 7 |
| 3.1 Elmore delay model | 7 |
| 3.2 Clock tree synthesis | 9 |
| 3.2.1 Construction of clock trees using the Elmore delay model [26] | 10 |
| 3.2.2 Deferred merge embedding (DME) Algorithm | 12 |
| 3.3 Clock tree optimization | 13 |
| 3.3.1 Timing slacks under OCVs | 14 |

| | | |
|--|--|----|
| 3.3.2 | Predicted timing quality and CTO | 15 |
| 3.4 | Gate sizing and layer assignment | 16 |
| 3.4.1 | Van Ginneken’s algorithm | 16 |
| 3.5 | Techniques of handling OCVs | 18 |
| 3.6 | Clock network synthesis for multiple modes of operations | 19 |
| CHAPTER 4: LATENCY CONSTRAINT GUIDED BUFFER SIZING AND LAYER AS- | | |
| SIGNMENT FOR CLOCK TREES WITH USEFUL SKEW | | 21 |
| 4.1 | Introduction | 21 |
| 4.1.1 | Motivation and the overview of the BLU framework | 21 |
| 4.1.2 | Proposed framework | 23 |
| 4.2 | The BLU framework | 23 |
| 4.2.1 | Baseline of the BLU framework | 24 |
| 4.2.2 | Relaxing the latency constraints | 25 |
| 4.2.3 | Tightening the latency constraints | 26 |
| 4.3 | Methodology | 28 |
| 4.4 | Experimental evaluation | 29 |
| 4.4.1 | Evaluation of positive delay adjustments | 32 |

| | | |
|--|--|----|
| 4.4.2 | Evaluation of negative delay adjustments | 33 |
| 4.5 | Summary and conclusion | 34 |
| CHAPTER 5: AN OCV-AWARE CLOCK TREE SYNTHESIS METHODOLOGY | | 36 |
| 5.1 | Introduction | 36 |
| 5.2 | Motivation | 37 |
| 5.2.1 | Limitations of previous works | 37 |
| 5.2.2 | Proposed methodology | 37 |
| 5.3 | Methodology | 40 |
| 5.3.1 | The overview of the framework | 40 |
| 5.3.2 | Enumeration of top-level trees | 42 |
| 5.3.3 | Pruning of top-level trees | 43 |
| 5.3.4 | Construction of virtual topology | 44 |
| 5.3.5 | Insertion of non-uniform safety margins | 44 |
| 5.3.6 | Specification of latency ranges | 46 |
| 5.3.7 | Construction of USTs | 47 |
| 5.4 | Experimental Evaluations | 48 |
| 5.4.1 | Evaluation of framework configurations | 49 |

| | | |
|---|--|----|
| 5.4.2 | Comparisons with state-of-the-art | 52 |
| 5.5 | Summary and conclusion | 54 |
| CHAPTER 6: SYNTHESIS OF CLOCK NETWORKS WITH A MODE RECONFIGURABLE | | |
| | TOPOLOGY | 55 |
| 6.1 | Introduction | 55 |
| 6.2 | Preliminaries | 56 |
| 6.3 | Problem formulation | 58 |
| 6.4 | The limitations of the previous studies | 59 |
| 6.5 | Proposed MRT Structure | 61 |
| 6.5.1 | Overview of the MRT structure | 61 |
| 6.5.2 | Improving the robustness in high performance modes | 62 |
| 6.5.3 | Reducing power in low performance modes | 64 |
| 6.6 | Methodology | 66 |
| 6.6.1 | Insertion of OR-gates | 67 |
| 6.6.2 | Construction of top-level clock trees | 68 |
| 6.6.2.1 | Construction of Topology Relation Graph | 69 |
| 6.6.2.2 | Edge pruning | 70 |

| | | |
|-----------------------------------|--|----|
| 6.6.2.3 | TRG guided Tree construction | 71 |
| 6.6.3 | Construction of the reconfigurable topology | 72 |
| 6.6.4 | Clock network optimization | 73 |
| 6.6.5 | Supply voltage selection | 74 |
| 6.7 | Experimental Results | 74 |
| 6.7.1 | Evaluation of MRT design configurations | 77 |
| 6.7.1.1 | Selection of the number of top-level trees | 77 |
| 6.7.1.2 | Evaluation of the TRG guided top-level tree construction | 78 |
| 6.7.1.3 | Evaluation of topology reconfiguration | 79 |
| 6.7.1.4 | Evaluation of the negative-edge of the clock signal | 81 |
| 6.7.2 | Evaluation of MRT structures | 82 |
| 6.8 | Summary and conclusion | 86 |
| CHAPTER 7: FUTURE WORKS | | 87 |
| LIST OF REFERENCES | | 88 |

LIST OF FIGURES

| | |
|--|----|
| Figure 2.1: Setup and hold time constraints between a pair of flip-flops. | 4 |
| Figure 2.2: Clock networks with a) tree, b) near-tree and c) non-tree topology. | 5 |
| Figure 3.1: a) A clock tree topology and b) The RC representation of the clock tree topology in (a). | 7 |
| Figure 3.2: a) A pair of subtrees and the embedding location are illustrated. b) The illustration of embedding locations as merging segments (<i>ms</i>) of the corresponding nodes in (a) for a zero skew merge. c) The illustration of embedding locations as merging regions (<i>mr</i>) of the corresponding nodes in (a) for a non-zero skew merge. | 12 |
| Figure 3.3: The illustration of the path that introduces OCV into the timing constraints between FF_i and FF_j | 14 |
| Figure 3.4: Extension of van Ginneken's algorithm [36]. | 17 |
| Figure 4.1: (a) A specified delay adjustment. (b) Delay adjustment realized by buffer insertion. (c) Proposed realization of delay adjustments using buffer sizing and layer assignment. | 22 |
| Figure 4.2: (a) SG. (b) P_{ins} and P_{wns} and delay adjustments. (c) latency constraints. (d) offsets. (e) van Ginneken's algorithm. (f) selected candidate solution. | 24 |
| Figure 4.3: (a) Latency constraints. (b) Relaxed latency constraints. (c) Method to find l_i^r | 25 |

| | |
|--|----|
| Figure 4.4: (a) Delay adjustments specified by Eq (3.9). (b) Clock tree after van Ginneken's algorithm. l_{max} is violated by the path marked with a red dashed line. (c) Generation of a delay adjustment constraint. | 26 |
| Figure 4.5: Proposed flow for the BLU framework. | 28 |
| Figure 5.1: (a) State-of-the-art CTS+CTO based synthesis flow. (b) Proposed OCV-aware clock tree synthesis flow. | 38 |
| Figure 5.2: Detailed flow of proposed OCV-aware synthesis methodology. | 40 |
| Figure 5.3: (a) Multiple candidate top-level topologies are enumerated. (b) The top-level topology is determined after pruning the candidate topologies. (c) The top-level clock tree and the virtual topology are constructed. (d) The latencies to the clock sinks are estimated using the virtual topology. (e) Non-uniform safety margins are inserted in the skew constraints and a latency range is specified for each clock sinks. (f) Bottom-level subtrees are iteratively constructed by inserting safety margins that are tailored to the topology. If the latency constraints are not satisfied, the construction process returns to (c) with improved timing predictions. | 41 |
| Figure 5.4: The illustration of (a) Uniform H-tree and (b) Non-uniform H-tree | 42 |
| Figure 5.5: Evaluation of the framework in terms of (a) Latency. (b) Capacitance. (c) TNS. (d) WNS. | 49 |
| Figure 5.6: Sensitivity to scaling parameter c_s on <i>ecg</i> | 50 |

| | |
|--|----|
| Figure 5.7: Evaluation of top-level tree construction using uniform H-tree vs. non-uniform H-tree. | 51 |
| Figure 6.1: Clock networks with different topologies and the illustration of short circuit current on each topology. | 59 |
| Figure 6.2: The arrival time distributions (for the positive-edge of the clock signal) at different points with respect to an OR-gate are shown in (a), (b) and (c). The variance of the arrival time distribution at the output of the OR-gate is tighter than that of the variance at the inputs, which demonstrates that OR-gates improve the robustness to variations. (d) The proposed MRT structure is constructed using OR-gates to provide robustness to variations. | 62 |
| Figure 6.3: (a) Flow for constructing MRT structures. (b) Flow for construction of the core top-level tree. (c) Flow for construction of the $N - 1$ subsequent top-level trees. | 66 |
| Figure 6.4: Example flow for the construction of an MRT structure when $N = 2$. (a) First stage subtrees. (b) Driver devices are inserted. (c) Core top-level clock tree is constructed. The driver devices used to construct the core-top level tree are colored in green. (d) The Topology Relation Graph (TRG) is constructed with respect to the topology. (e) The TRG after edge pruning is applied. (f) The second top-level clock tree is constructed using TRG. (g) The clock network after the construction of top-level trees shown in (c) and (f). (h) The clock network after constructing the reconfigurable topology. | 68 |
| Figure 6.5: The reconfigurable topology | 73 |

| | |
|---|----|
| Figure 6.6: Evaluation of MRT structures with different number of top-level trees on the circuits (a) <i>dma</i> and (b) <i>aes</i> . The performance is evaluated in terms of power and robustness to variations. The clock network structure with 1 top-level tree is equivalent to the traditional clock tree. | 77 |
| Figure 6.7: Evaluation of guiding the top-level tree construction using different cost metrics. The evaluation is performed in terms of normalized power and normalized robustness (D). | 79 |
| Figure 6.8: Evaluation of DVFS vs reconfiguration combined with DVFS in terms of average (a) supply voltage and switching capacitance and (b) total circuit power for different ratios of C_{comb} and C_{clk} . The experimental results shown in the figure are the average values for the benchmarks in Table 6.3. | 80 |
| Figure 6.9: Evaluation of the MRT structure for the negative-edge and the positive-edge of the clock signal. | 81 |
| Figure 6.10: Histogram of skews from Monte Carlo simulations of (a) Tree and (b) MRT structures on <i>usbf</i> | 85 |

LIST OF TABLES

| | |
|--|----|
| Table 4.1: Circuits in [31]. | 29 |
| Table 4.2: Evaluation of various tree structures in terms of total capacitance. | 31 |
| Table 4.3: Evaluation of various tree structures in terms of runtime. | 31 |
| Table 4.4: Evaluation of negative delay adjustments. | 35 |
| Table 5.1: Holistic guidelines and objectives for different parts of a clock tree. | 39 |
| Table 5.2: Evaluation of the clock tree structures in terms of performance and synthesis time. | 52 |
| Table 6.1: Comparison between clock networks with tree, near-tree, and non-tree topolo- gies. | 60 |
| Table 6.2: Comparison between DVFS and topology reconfiguration combined with DVFS. | 64 |
| Table 6.3: Benchmarks in [7]. | 75 |
| Table 6.4: The properties of the structures. | 75 |
| Table 6.5: Evaluation of structures in terms of power consumption, supply voltage and timing quality. | 82 |

CHAPTER 1: INTRODUCTION

Sequential VLSI circuits consist of combinational logic and sequential elements. The sequential elements such as registers and flip-flops require a synchronization signal to capture and propagate the correct data signal. The synchronization is facilitated using a clock network that delivers a clock signal from a clock source to all sequential elements. There exists timing constraints between each sequential element that are separated by combinational logic in the data and control paths. For the functional correctness of the circuit, the clock signal must be delivered meeting the timing constraints.

Clock skew is the difference in the arrival time between a pair of flip-flops. Ideally, the clock signal is required to be delivered to the flip-flops at the same time to meet the timing constraints. Therefore, early studies focused on synthesizing clock networks to meet zero-skew [1, 3–5, 23, 41] or bounded-skew [6, 51, 52] constraints in order to achieve minimum possible clock skew. However, this may not be necessary because the timing constraints imposed by the combinational logic between certain pairs of clock sinks may actually be looser. In such cases, the timing margins can be allowed to be utilized by the clock skew. This is advantageous because there is a trade-off between the clock skew and the hardware cost of a clock network. Consequently, lower cost clock networks can be synthesized by utilizing available timing margins, which can be achieved by constructing useful skew trees (USTs) [22, 31, 34, 53, 70].

Modern VLSI circuits are required to meet tight timing constraints under variations while operating at high clock frequencies. Constructing clock networks with small nominal skews is not too difficult [1, 3, 22, 23]. However, it is very challenging to meet tight skew constraints while a clock network is under the influence of process, voltage and temperature (PVT) variations [2, 20, 21, 64, 65, 68]. The main reason is that the variations result in deterioration in the timing of

the clock network, which introduces timing violations. To meet the timing constraints, the effects of variations must be accurately accounted when synthesizing clock networks.

When the clock signal propagates through the interconnects of a clock network, the transition time of the clock signal may severely degrade. To preserve the functionality of a circuit, the clock signal must be delivered with a sharp transition. This can be ensured by meeting a maximum transition time constraint when delivering the clock signal. To achieve this, typically clock buffers and clock inverters are required to be inserted at the appropriate locations when constructing the clock networks.

In today's advanced technology nodes, modern VLSI circuits are required to operate in low and high performance modes to cater to variable frequency and power requirements. Consequently, the clock networks for such circuits must be synthesized meeting drastically different timing constraints in the different modes. This is particularly more challenging under the influence of variations. To meet both the timing constraints and the power budgets, advanced structures and synthesis methodologies are required.

This dissertation focuses on techniques and methodologies to construct robust clock networks that achieve high timing quality under the influence of variations. Clock network construction problems are formulated and solved using various algorithms, clock network structures, and optimization techniques. In this dissertation, an overview of the backgrounds and the previous works on the construction of clock networks are reviewed in Chapter 2 and Chapter 3, respectively. In Chapter 4, a buffer sizing and layer assignment framework is proposed to improve the timing quality of a constructed clock tree. In Chapter 5, an On-chip variation-aware clock tree synthesis methodology is proposed. Finally, a methodology to construct clock networks that can reconfigure its topology based on the active mode of operation is proposed. The dissertation is concluded with the future research directions in Chapter 7.

CHAPTER 2: PRELIMINARIES

In this chapter, the preliminaries are reviewed.

2.1 Timing constraints

Synchronous circuits are synchronized by delivering a clock signal from the clock source to flip-flops (or clock sinks). There is a setup and hold time constraint between any pair of flip-flops (FFs) that are only separated by combinational logic in the data and control paths, which is shown in Figure 2.1. The clock signal must be delivered meeting setup and hold time constraints for the functional correctness of the circuit. The setup and hold time constraints between the launching flip-flop FF_i and the capturing flip-flop FF_j are formulated, as follows [26]:

$$t_i + t_i^{CQ} + t_{ij}^{max} + t_j^S \leq t_j + T, \quad (2.1)$$

$$t_i + t_i^{CQ} + t_{ij}^{min} \geq t_j + t_j^H, \quad (2.2)$$

where t_i and t_j are the arrival times of the clock signal to the FF_i and FF_j , respectively. t_i^{CQ} is the clock to output delay of the capturing flip-flop; T is the clock period; t_j^S and t_j^H are the setup time and hold time of FF_j , respectively. t_{ij}^{max} (t_{ij}^{min}) is the maximum (minimum) delay through the combinational logic between FF_i and FF_j . The clock skew between the pair of clock sinks is defined to be equal to $skew_{ij} = t_i - t_j$. Using $ub_{ij} = T - t_i^{CQ} - t_{ij}^{max} - t_j^S$ and $lb_{ij} = t_i^{CQ} + t_{ij}^{min} - t_j^H$, the setup and hold time constraints in Eq (2.1) and Eq (2.2) can be respectively reformulated as explicit skew constraints, as follows:

$$lb_{ij} \leq skew_{ij} \leq ub_{ij} \quad (2.3)$$

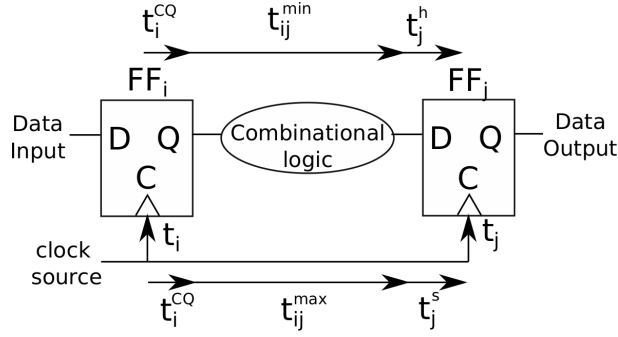


Figure 2.1: Setup and hold time constraints between a pair of flip-flops.

2.2 Variations in the timing

The intrinsic variability in the semiconductor fabrication and fluctuations in the environmental operating conditions introduces variations in the timing. The clock network that synchronizes each circuit is naturally vulnerable to these variations because it spans across each chip. Consequently, the variations in the timing must be accounted for when constructing clock networks.

The variations are introduced by the variability in the manufacturing process such as differences in the channel lengths, wire widths, and oxide thickness; and the changes in the environmental conditions such as temperature, supply voltage, and cross-talk noise.

The variations within a single die are spatially correlated and commonly referred to as On-Chip variations (OCV). However, the variations between each die are directly dependent on the parameters of the process at the time of manufacturing. These process parameters are captured using process corners. To account for the OCVs at a process corner, spatially correlated variations are applied to the process parameters across the chip. Consequently, the timing quality of a clock network can be evaluated by performing simulations for each corner using the process parameters that are obtained after applying OCVs.

Clock networks can easily be constructed to meet the timing constraints under the nominal conditions. However, it is challenging to meet the timing constraints under variations because variations may severely shrink the timing margins. The variations in the timing can be integrated into the explicit skew constraints in Eq (2.3), as follows:

$$lb_{ij} + (\delta_i + \delta_j) \leq skew_{ij} \leq ub_{ij} - (\delta_i + \delta_j) \quad (2.4)$$

where δ_i and δ_j are the timing deteriorates at FF_i and FF_j , respectively. The timing deteriorates are directly correlated with the distance between FF_i and FF_j in the topology of the clock network [17, 35].

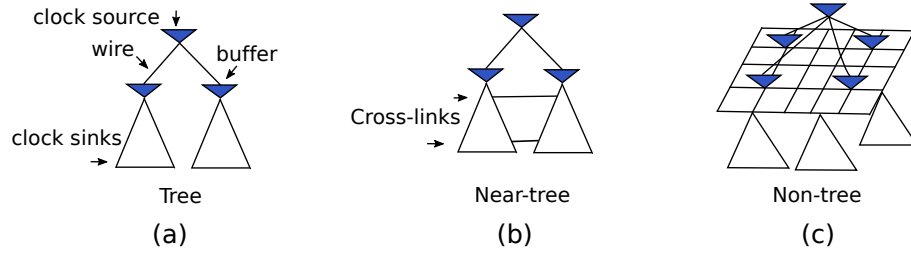


Figure 2.2: Clock networks with a) tree, b) near-tree and c) non-tree topology.

2.3 Clock networks with different topologies

The overall power consumption and robustness to variations of a clock network is determined by the topology, which can be in the form of a tree, near-tree, or non-tree, which are illustrated in Figure 2.2. Clock trees consume the least power but are vulnerable to variations. Clock networks with a non-tree or near-tree topology have multiple paths from the clock source to the clock sinks. When inserted appropriately, the alternative paths neutralize the negative impact of variations. However, clock networks with a non-tree topology (as clock meshes) may consume 3X-5X more

power than a clock tree [19, 25, 27]. Near-tree structures provide high robustness to variations, while power consumption is only slightly higher than the power consumption of a clock tree. Therefore, clock network structures in the form of a near-tree topology [9, 13, 15, 16, 24, 35, 69] has been the focus of many recent studies. More details about the properties of the clock network topologies are provided in Chapter 6.

CHAPTER 3: REVIEW OF PREVIOUS WORKS

In this chapter, we present an overview of the backgrounds and the previous works.

3.1 Elmore delay model

The clock network synthesis process requires a delay model to propagate the clock signal in order to evaluate the timing performance of a clock network. The interconnects of a clock network can be represented as a lumped Resistor-Capacitor (RC) network. The delay of any node in an RC network can be obtained by formulating and solving differential equation problems. In general, Elmore delay model [50] is extensively used to obtain the delay in an RC network, which is based on the first order approximation of the delay. There exists complex delay models that are used to more accurately compute the delay in an RC network [14, 44–47, 60]. However, these techniques are computationally expensive compared to the Elmore delay model.

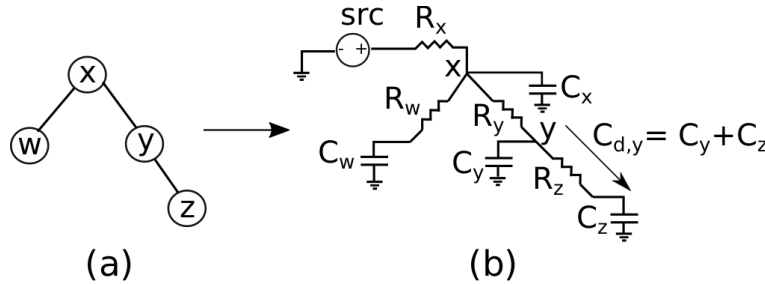


Figure 3.1: a) A clock tree topology and b) The RC representation of the clock tree topology in (a).

A clock network with a tree topology and its equivalent RC representation is illustrated in (a) and (b) of Figure 3.1, respectively. The delay of a specific node in the RC network shown in Figure 3.1(a) can be computed using the equation, as follows [50]:

$$t_{v_s, v_i} = \sum_{\forall j \in \text{path}(v_s, v_i)} R_j \cdot C_{d,j} \quad (3.1)$$

where v_i is a node in the topology; t_{v_s, v_i} is the propagation delay from the clock source (v_s) to v_i ; $\text{path}(v_s, v_i)$ is the path from v_s to v_i ; R_j is the total resistance of a wire segment that connects two nodes on the $\text{path}(v_s, v_i)$; $C_{d,j}$ is the total downstream capacitance on v_j , i.e., the total capacitance of the subtree rooted at v_j .

The voltage at a certain time (t) on each node in the RC network can be computed using the equation as follows:

$$V_i(t) = V_{dd} \cdot (1 - e^{-t/t_{v_s, v_i}}) \quad (3.2)$$

where $V_i(t)$ is the voltage at node i ; V_{dd} is the supply voltage. The delay is generally defined as the duration for the output capacitance to be charged from 0V to $V_{dd}/2$. Using the Eq (3.2), the total elapsed time until the voltage reaches to $V_{dd}/2$ can be derived, as follows:

$$t_{d, v_i} = \ln(2) \cdot t_{v_s, v_i} \quad (3.3)$$

where t_{d, v_i} is the delay of the clock signal that propagates from v_s to v_i . The switching performance of the interconnects can be evaluated using the transition time at each node. The transition time at a certain node is the elapsed time that the voltage changes between $0.1V_{dd}$ and $0.9V_{dd}$ [26]. The transition time at any internal node in an RC network can be formulated using Eq (3.2), as follows [54]:

$$t_{tr, v_i} = \ln(9) \cdot t_{v_s, v_i} \quad (3.4)$$

where t_{tr,v_i} is the transition time of the clock signal at v_i . Using the Eq (3.3) and Eq (3.4), the delay and the transition time of the clock signal can be respectively computed to evaluate the timing of the clock network.

3.2 Clock tree synthesis

Clock tree synthesis (CTS) is a process that is performed to construct a clock tree that delivers a clock signal from a clock source to each clock sinks. The inputs to this process are x-y coordinates of clock sinks and a technology library. Given these inputs, a clock tree is constructed to connect the clock source to the clock sinks using wires and devices from the technology library. Moreover, the clock signal must be delivered meeting the skew constraints in Eq (2.3) for the functional correctness of the circuits.

Early studies on CTS focused on the construction of zero skew trees (ZSTs) [5, 23, 56] and bounded skew trees (BSTs) [6]. Later, the construction of useful skew trees (USTs) [22, 28, 32, 34] was explored to meet the explicit (arbitrary) skew constraints imposed between each pair of sequential elements (or clock sinks) that are only separated by combinational logic in the control logic and data paths.

In the remaining of this section, the backgrounds for the clock tree synthesis process is presented. We mainly focused on explaining the construction of the ZSTs. However, the details of the modifications required to synthesize BSTs and USTs are provided for different steps of the CTS process.

3.2.1 Construction of clock trees using the Elmore delay model [26]

In this section, it is explained how clock trees can be constructed using the Elmore delay model in Section 3.1, which is described more detailed in [26]. Zero skew tree construction is proposed in [5, 23, 56]. The construction is based on iteratively merging two subtrees (or clock sinks) into a large subtree such that a target skew constraint is guaranteed to be satisfied under the Elmore delay model. This is achieved by propagating the downstream delay of each subtree and finding a tapping point where two subtrees are required to be merged to meet the skew constraint.

Let u and v be the two subtrees; r be the newly formed subtree that is constructed after merging subtrees u and v ; $t_{u,r}^{max}$ ($t_{v,r}^{max}$) be the maximum downstream delay through the path from r to u (v). $t_{u,r}^{max}$ and $t_{v,r}^{max}$ can be respectively computed, as follows:

$$\begin{aligned} t_{u,r}^{max} &= t_u^{max} + x \cdot L \cdot R_{wire} \cdot (x \cdot L \cdot C_{wire} + C_u) \\ t_{v,r}^{max} &= t_v^{max} + (1 - x) \cdot L \cdot R_{wire} \cdot ((1 - x) \cdot L \cdot C_{wire} + C_v) \end{aligned} \quad (3.5)$$

where t_u^{max} and t_v^{max} are the maximum downstream delay of subtrees u and v , respectively; L is the shortest manhattan distance between u and v ; x is the ratio of the wirelength between u and tapping point r to the L . C_u and C_v are the downstream capacitance at u and v , respectively. R_{wire} and C_{wire} are the resistance and capacitance per unit distance, respectively.

To construct zero skew trees, two subtrees are required to be merged such that $t_{u,r}^{max} = t_{v,r}^{max}$ is satisfied. Consequently, a zero skew merge can be achieved by solving the following equation to find the required x , as follows [23, 26]:

$$x = \frac{t_v^{max} - t_u^{max} + R_{wire} \cdot L \cdot (C_{wire} \cdot L + C_v)}{R_{wire} \cdot L \cdot (C_{wire} \cdot L + C_u + C_v)} \quad (3.6)$$

After finding x for a zero skew merge using Eq (3.6), the distance from u to the tapping point can be computed as xL , which is shown in Figure 3.2(a). The embedding locations for a zero skew merge can be represented as merging segments (ms), which is shown in Figure 3.2(b). Note that a detour wire is required to be inserted to balance the delay of the subtrees when $x < 0$ or $x > 1$ conditions are obtained. Next, the maximum propagation delay of the subtree r is updated to be the maximum of $t_{u,r}^{max}$ and $t_{v,r}^{max}$, i.e., $t_r^{max} = \max\{t_{u,r}^{max}, t_{v,r}^{max}\}$. Consequently, a zero skew clock tree can be constructed by iteratively performing a zero-skew merge on a pair of subtrees until a single subtree is obtained.

Similar to the zero skew merges, merges for non-zero skew constraints can be performed to construct BSTs and USTs under the Elmore delay model. This can be achieved by propagating both the maximum and minimum downstream delay of the subtrees as in Eq (3.5). Let $t_{u,r}^{min}$ and $t_{v,r}^{min}$ be the minimum downstream delay through the path from r to u and v , respectively. Using the maximum and minimum downstream delay of subtrees, multiple equations as in Eq (3.6) can be formulated and solved to determine the possible embedding locations that ensure to meet $|t_{u,r}^{max} - t_{v,r}^{min}| < skew_{u,v}$ and $|t_{v,r}^{max} - t_{u,r}^{min}| < skew_{v,u}$. Let x^+ and x^- be the maximum and minimum possible x value that meets the skew constraints. While performing merge operations in the construction of BSTs and USTs, there exists a range of possible x values instead of a single x value due to that a non-zero skew is allowed to be utilized. Therefore, the skew constraints are guaranteed to be satisfied by selecting any x in the range of $[x^-, x^+]$. The embedding locations for a non-zero skew merge are represented using merging regions (mr), which is shown in Figure 3.2(c).

In the next section, we review a well-known algorithm to construct clock trees with buffers, which guarantees to meet both the skew constraints and the transition time constraint under the Elmore delay model.

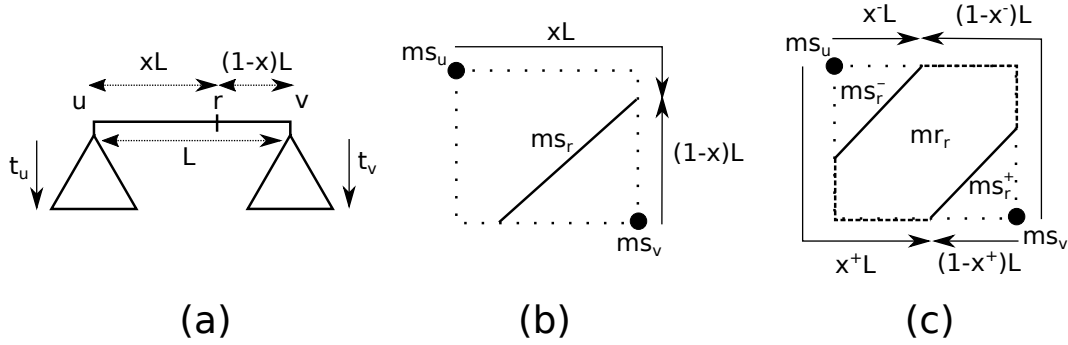


Figure 3.2: a) A pair of subtrees and the embedding location are illustrated. b) The illustration of embedding locations as merging segments (ms) of the corresponding nodes in (a) for a zero skew merge. c) The illustration of embedding locations as merging regions (mr) of the corresponding nodes in (a) for a non-zero skew merge.

3.2.2 Deferred merge embedding (DME) Algorithm

In [1, 4, 23], DME algorithm is proposed to construct ZSTs. The DME algorithm is extended to construct BSTs [6] and USTs [22]. Given a clock tree topology and a set of clock sinks, the DME algorithm constructs a clock tree while ensuring to meet the skew constraints under the Elmore delay model. The algorithm is based on first iteratively performing a bottom-up merging of subtrees phase and insertion of buffers phase. Next, a top-down embedding phase is performed to determine the exact locations of the internal nodes in each subtree.

Merging of subtrees: The bottom-up merging of subtrees phase is based on iteratively merging the pair of subtree (or clock sinks) that requires minimum wirelength to be joined while meeting the skew constraint under the Elmore delay model. The embedding locations for each pair of subtree is determined using the procedure explained in Section 3.2.1 and the corresponding pair of subtree is merged at the embedding location. The merging process is facilitated by a nearest neighbour graph (NNG) [5]. An NNG is used to capture the subtrees and the wirelength distances between the pairs of subtrees. Iteratively, the pair of subtrees that require minimum amount of wirelength

to be joined is selected to be merged. The transition time is evaluated after each pair of subtrees have been merged. If the transition time constraint is violated, subtrees are unmerged and locked for further merging operations. Merging operations are performed until all the subtrees are locked.

Insertion of buffers: The input to the insertion of buffers step is a set of locked subtrees from the merging of subtrees step. For each subtree, a minimum sized buffer that can drive the locked subtree (without violating the transition time constraint) is inserted at the root. Next, a stem wire is inserted between the buffer and the subtree [4].

The merging of subtrees and the insertion of buffers steps are repeated until there is one single subtree left. Finally, a top-down embedding phase is performed to determine the exact locations (x-y coordinates) of each internal node within each subtree. More details about the DME algorithm and its variants are in [1, 4, 6, 22, 23, 26].

3.3 Clock tree optimization

Clock tree optimization (CTO) is a process that is performed to eliminate the timing violations that are obtained after the CTS process. State-of-the-art CTO techniques are based on specifying and realizing delay adjustments using buffers and detour wires to remove the timing violations [30, 38, 39]. In this section, we first present how the slacks in the timing constraints are captured (see Section 3.3.1). Next, we explain how delay adjustments are specified to remove the timing violations (see Section 3.3.2).

3.3.1 Timing slacks under OCVs

After a clock tree has been constructed, the slack in the constraints can be computed, as follows:

$$setup_slack_{ij} = t_j - t_i + T - t_j^S - t_{ij}^{CQ} - t_{ij}^{max} - \delta_i - \delta_j, \quad (3.7)$$

$$hold_slack_{ij} = t_i - t_j + t_{ij}^{CQ} + t_{ij}^{min} - t_j^H - \delta_j - \delta_i, \quad (3.8)$$

where $setup_slack_{ij}$ and $hold_slack_{ij}$ are respectively the slacks in the setup and hold time constraints in Eq (2.1) and Eq (2.2). A negative slack implies a violation of a timing constraint while a positive slack implies available margins in a timing constraint. δ_i and δ_j are the timing deteriorates that are introduced by OCV.

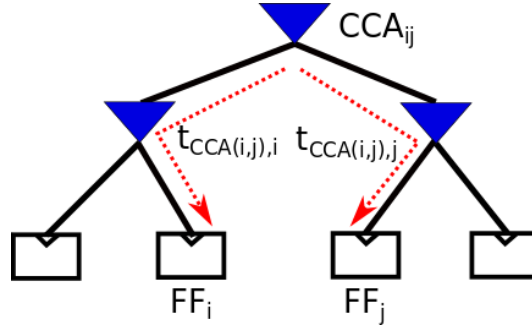


Figure 3.3: The illustration of the path that introduces OCV into the timing constraints between FF_i and FF_j .

The timing deteriorates δ_i and δ_j are dependent on the distance between FF_i and FF_j in the clock tree topology. A clock tree containing the flip-flops FF_i and FF_j is illustrated in Figure 3.3. Let the closest common ancestor (CCA) between FF_i and FF_j in the clock tree be denoted $CCA(i,j)$ [35]. Based on the model in [39], δ_i and δ_j are equal to $c_{ocv} \cdot t_{CCA(i,j),i}$ and $c_{ocv} \cdot t_{CCA(i,j),j}$, respectively. $t_{CCA(i,j),i}$ and $t_{CCA(i,j),j}$ are the propagation delays from the $CCA(i,j)$ to FF_i and FF_j , respectively. c_{ocv} is a parameter determined by circuit simulations.

The slack in the timing constraints can be captured in a slack graph (SG) [30]. In an SG $G = (V, E)$, the vertices V represent clock sinks and the edge weights E represent the slack in the timing constraints. An edge e_{ij} with weight $w_{ij} = \text{setup_slack}_{ij}$ is added for each setup time constraint. An edge e_{ji} with weight $w_{ji} = \text{hold_slack}_{ij}$ is added for each hold time constraint. The timing quality of a clock tree is measured in total negative slack (TNS) and worst negative slack (WNS), i.e., the sum and the maximum of the negative timing slacks in Eq (3.7) and Eq (3.8).

3.3.2 Predicted timing quality and CTO

Timing violations in constructed clock trees are typically eliminated by realizing non-negative delay adjustments [12, 30, 38, 39]. A delay adjustment is a change of the propagation delay through a branch in the clock tree. Note that the use of delay adjustments is equivalent to specifying latency constraints in the form of points [38, 39] or ranges [30] for the clock sinks.

Let $\Delta_k \geq 0$ be a delay adjustment at a location k in a clock tree. Delay adjustments are typically restricted to locations where buffers are placed in the topology to avoid disrupting the overall timing [39]. Next, the final timing quality is predicted by specifying a set of delay adjustments using an LP formulation, as follows [30, 39]:

$$\begin{aligned}
\min \quad & c_t \sum_{k \in B} \Delta_k + c_{wns} P_{wns} + c_{tns} P_{tns} \\
s.t. \quad & \sum_{k \in \text{path}(CCA(i,j),i)} (1 + c_{ocv}) \Delta_k - \sum_{h \in \text{path}(CCA(i,j),j)} (1 - c_{ocv}) \Delta_h - s_{ij} \leq w_{ij}, \\
& s_{ij} \leq P_{wns}, \quad (i, j) \in E, \\
& \sum_{(i,j) \in E} s_{ij} = P_{tns},
\end{aligned} \tag{3.9}$$

where $\text{path}(i, CCA(i, j))$ and $\text{path}(j, CCA(i, j))$ respectively denote the buffers on the paths from

$CCA(i, j)$ to FF_i and FF_j . w_{ij} is the weight of an edge in the SG. $s_{ij} \geq 0$ is a timing violation that is not eliminated by realizing the specified delay adjustments. P_{tns} and P_{wns} are respectively the predicted TNS and WNS that is achieved by realizing the specified delay adjustments. The c_t , c_{wns} and c_{tns} parameters are used to balance the different terms in the objective function. The $(1 + c_{ocv})$ and $(1 - c_{ocv})$ factors account for the timing deteriorates introduced by the specified delay adjustments.

3.4 Gate sizing and layer assignment

Gate sizing and layer assignment can be performed to save power while meeting constraints on the maximum latency (or propagation delay). Buffer sizing and layer assignment for zero skew and bounded skew clock trees has been studied in [29, 37, 41, 48]. Buffer sizing for USTs was performed using a Taylor expansion and sequential linear programming in [33, 48]. Nevertheless, it is difficult to handle discrete buffer sizes and layer assignments using linearization. In Section 3.4.1, we explain an algorithm that is commonly used to perform discrete gate sizing and layer assignment.

3.4.1 Van Ginneken's algorithm

Van Ginneken's algorithm is a well known dynamic programming algorithm that minimizes the latency of an RC tree using buffer sizing and layer assignment under the Elmore delay model [43]. In [36], the algorithm was extended to find all Pareto optimal solutions in terms of power consumption and latency while considering slew propagation. Moreover, it is straightforward to set different latency constraints for different clock sinks.

The algorithm solves the problem of selecting a buffer size for each buffer and a layer assignment

for each wire in a clock tree by propagating candidate solutions from the leaf nodes to the source node, which is illustrated in Figure 3.4. Each candidate c_k stores the maximal downstream delay d_k , non-shielded downstream capacitance cap_k , and cost in terms of total capacitance $cost_k$, i.e., $c_k = (d_k, cap_k, cost_k)$ [36]. First, a candidate solution with zero maximum downstream delay is created at each clock sink. Both the non-shielded capacitance and the cost are set equal to the sink capacitance, which is illustrated in the bottom-left of Figure 3.4.

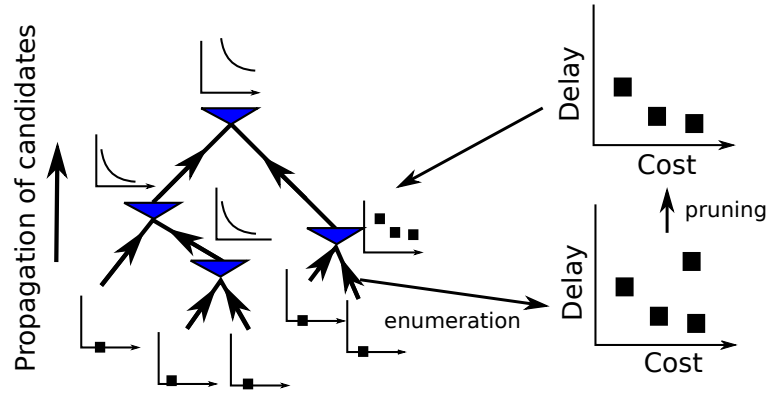


Figure 3.4: Extension of van Ginneken's algorithm [36].

The candidate solutions at the sinks are then propagated up towards the root of the tree. When a candidate solution is propagated through an edge (buffer or wire), all possible realizations (buffer sizes or layer assignments) are enumerated as candidates to realize the edge, shown in the bottom-right of Figure 3.4. New candidates are also formed when two branches in the clock tree are joined. Pruning is applied to eliminate non-Pareto optimal solutions, which is shown in the top-right of Figure 3.4. Next, the minimum cost candidate solution that meets a defined latency requirement is selected at the root. Lastly, the size for each buffer and the layer assignment for each wire is determined based on the selected candidate.

3.5 Techniques of handling OCVs

In this section, an overview of previous works on the construction of clock trees under the influence of OCVs is presented. Most common approaches to handle the OCVs in the literature are: i) inserting safety margins into the timing constraints before (or during) the clock tree synthesis, ii) applying a clock tree optimization phase to an initially constructed clock tree in order to eliminate the timing violations, and iii) reconstructing the topology of a clock tree to reduce the effects of OCVs.

Techniques of handling OCVs by inserting safety margins before (or during) the clock tree synthesis process have been explored in [22, 32, 58]. The insertion of uniform safety margins was investigated in [22, 32]. The limitation of that approach is that the required safety margins depend on the clock tree topology and are therefore non-uniform. Consequently, the use of non-uniform safety margins results in that many timing constraints will have excessive (or inadequate) timing margins inserted, which translates into substantial hardware overheads (or timing violations). To reduce the overheads, the magnitude of the safety margins can be tailored to the clock tree topology during the synthesis process [31, 58]. Unfortunately, these techniques still result in clock trees with timing violations [58] or unacceptable overheads [31].

The state-of-the-art methodology for synthesizing clock networks consists of a CTS phase and a CTO phase, which are detailed in Section 3.2 and Section 3.3, respectively. In the CTS phase, an initial clock tree is first constructed. Next, the impact of the OCVs and the associated timing violations are determined, which is explained in Section 3.3.1. An aggressive CTO phase is subsequently applied to eliminate all timing violations [12, 30], which is based on specifying and realizing delay insertions as explained in Section 3.3.2. The delay insertions improve timing by redistributing timing margins from satisfied to unsatisfied timing constraints. While CTO is capable of significantly improving the timing quality of most clock trees, there is no guarantee that

the optimization process will converge to a solution without timing violations. In particular, it may be impossible to close timing if the quality of the initial clock tree is poor. Advanced CTO techniques have recently been investigated to solve this challenge. The reconstruction of the clock tree topology with the objective of minimizing latency or placing certain clock sinks close in the topology was explored in [39, 42, 55]. Nevertheless, CTO flows are still time consuming and often require costly manual intervention. This stems from that the state-of-the-art design flows consider the impact of OCVs too late in the synthesis process.

In Chapter 4, a framework that can realize both positive and negative delay adjustments using buffer sizing and layer assignment is proposed. In addition, a clock tree synthesis methodology that can account for the effects of OCVs early in the synthesis flow is proposed in Chapter 5.

3.6 Clock network synthesis for multiple modes of operations

Meeting tight skew constraints under the influence of PVT variations is particularly challenging for circuits that are required to operate in low and high performance modes. In each mode, the timing constraints are drastically different. For the functional correctness of the circuit, clock networks must be synthesized such that the clock signal is delivered meeting timing constraints in both modes of operation under variations.

Many recent studies are focused on constructing clock trees that utilize a combination of guard-bands and useful skew to satisfy timing constraints under the variations in multiple modes [34]. However, for circuits with strict requirements on the clock frequency (in the high performance modes), it may be impossible to satisfy the timing constraints using a clock network with a tree topology.

As described in Section 2.2, the robustness to variations of a clock network is highly dependent on

the topology. The clock networks with a near-tree structure provides robustness to variations while the power consumption is similar to that of a clock tree. In the past decade, clock networks with near-tree topologies have been investigated, which promise significant improvements in robustness while the power consumption is similar to that of a clock tree [9, 13, 15, 16, 24, 35]. In [13, 15, 24], near-tree structures were constructed by inserting cross-links. In [35], multiple tree structures were fused together in order to create a multilevel fusion tree. In [9], a locally-merged structure was formed by fusing subtrees at internal nodes of a clock tree. The fusion was performed at internal nodes instead of at the clock sinks to reduce hardware overheads. A drawback of all near-tree structures is that there are multiple gates driving the same net of wires, which may result in short circuit current. Moreover, the same topology is used in every operational mode, despite that the robustness provided by a near-tree is only required in the high performance modes. It is easy to understand that a clock network with a tree topology could easily meet the timing constraints with much smaller power in the low performance modes.

In this dissertation, we addressed the problem of constructing clock networks for multiple modes of operations under variations, and proposed a new synthesis methodology to solve this problem in Chapter 6.

CHAPTER 4: LATENCY CONSTRAINT GUIDED BUFFER SIZING AND LAYER ASSIGNMENT FOR CLOCK TREES WITH USEFUL SKEW

4.1 Introduction

Closing timing using CTO is a tremendously challenging problem that may require designer intervention. The timing constraints must be satisfied even while the circuit is under the influence of OCVs. In addition, power consumption is a key design constraint in the advanced technology nodes. To meet both timing constraints and power constraints, it is essential to utilize every available timing margin by exploiting useful skew. In this chapter, we present a latency constraint guided buffer sizing and layer assignment framework for clock trees with useful skew, called the (BLU) framework. The proposed framework is applied after an initial clock tree has been constructed, and before traditional CTO is applied. The key idea is to perform a CTO by realizing delay adjustments using buffer sizing and layer assignment. The proposed framework can realize negative delay adjustments and handle discrete buffer sizes.

4.1.1 Motivation and the overview of the BLU framework

A delay adjustment is illustrated in Figure 4.1(a). The specified delay adjustments are traditionally realized by inserting delay buffers (see Section 3.3), which is shown in Figure 4.1(b). In contrast, this chapter proposes to realize delay adjustments using buffer sizing and layer assignment, which is illustrated in Figure 4.1(c). Compared with buffer insertion, layer assignment is a more gentle method of realizing delay adjustments that may result in lower power consumption.

This chapter is based on the paper that is published at 2019 Asia and South Pacific Design Automation Conference [59] © 2019 ACM.

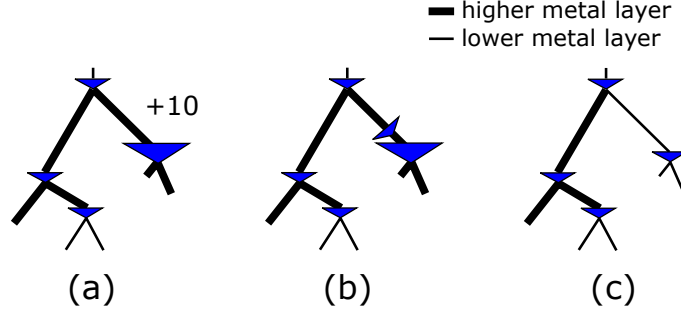


Figure 4.1: (a) A specified delay adjustment. (b) Delay adjustment realized by buffer insertion. (c) Proposed realization of delay adjustments using buffer sizing and layer assignment.

The BLU framework specifies a set of delay adjustments and predicts the final timing quality (P_{tns} , P_{wns}) that would be achieved using traditional CTO. P_{tns} and P_{wns} are respectively the predicted TNS and WNS. Next, the delay adjustments are translated into latency constraints without degrading P_{tns} and P_{wns} . Using the specified latency constraints, buffer sizing and layer assignment is conducted using an extension of van Ginneken’s algorithm, i.e., delay adjustments are realized while reducing the total capacitive cost. To further reduce power consumption, the BLU framework attempts to relax each point constraint into a latency range without degrading P_{tns} and P_{wns} . Moreover, a method of improving P_{tns} and P_{wns} by specifying tight latency constraints using negative delay adjustments is proposed. Lastly, traditional CTO is applied to realize remaining delay adjustments such that TNS and WNS are reduced to P_{tns} and P_{wns} .

Compared with in [33, 48], the BLU framework allows buffer sizing and layer assignment to be performed while utilizing discrete buffer and interconnect libraries. We consider the BLU framework to be orthogonal to the techniques of realizing negative delay adjustments by reconstructing the topology of a clock tree in [39, 42].

The experimental results demonstrate that the BLU framework is capable of reducing total capacitance, TNS and WNS with 13%, 58%, and 20%, respectively.

4.1.2 Proposed framework

Non-negative delay adjustments are realized by inserting buffers and detour wires during CTO, which translates into overhead in terms of total capacitance. In contrast, van Ginneken's algorithm (with the extension in [36]) is capable of trading-off maximum delay for total capacitive cost. The key idea of the BLU framework is to use buffer sizing and layer assignment to realize delay adjustments. Consequently, the proposed framework has the potential to improve power consumption while reducing TNS and WNS to P_{tns} and P_{wns} , respectively. Extensions to further improve performance are presented in Section 4.2.

Van Ginneken's algorithm requires maximum delay (or latency) constraints. The latency constraints are obtained from the delay adjustments specified using Eq (3.9). The obtained constraints are in the form of points. Therefore, each delay adjustment is required to be realized exactly. Let l_i denote the upper bound of the latency constraint to sink i . The constraints requires each arrival time t_i to be equal to the latency constraint l_i . However, van Ginneken's algorithm only ensures that $t_i \leq l_i$. Nevertheless, it is expected that the arrival times t_i will be close to l_i , as increasing the arrival time (or delay) typically results in a reduction of total capacitive cost. Moreover, the difference between t_i and l_i can be realized through traditional CTO after the proposed framework has been applied.

4.2 The BLU framework

The baseline of the BLU framework is presented in Section 4.2.1. In Section 4.2.2, point constraints are relaxed into range constraints to save capacitive cost. In Section 4.2.3, the latency constraints are tightened to improve P_{tns} and P_{wns} .

4.2.1 Baseline of the BLU framework

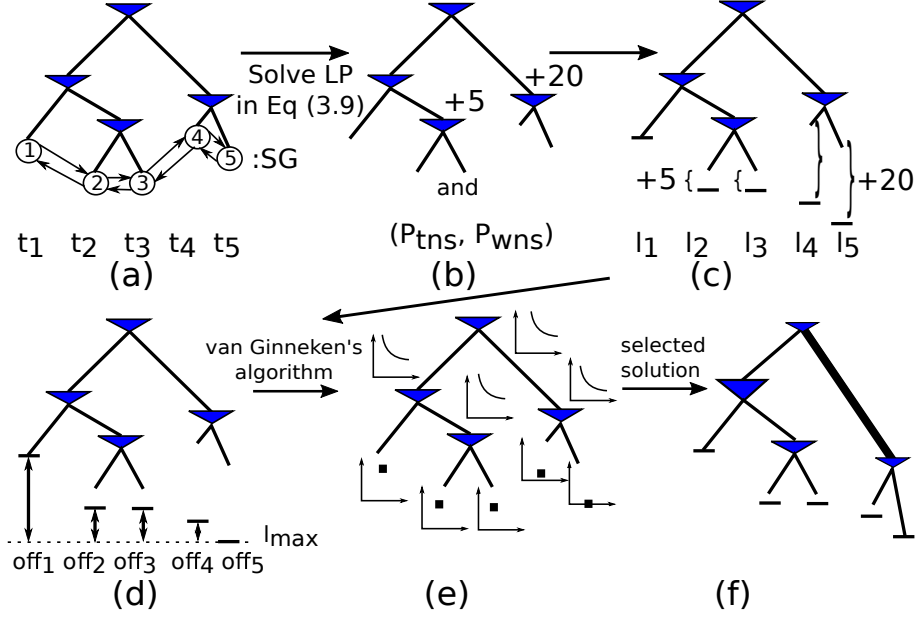


Figure 4.2: (a) SG. (b) P_{tns} and P_{wns} and delay adjustments. (c) latency constraints. (d) offsets. (e) van Ginneken's algorithm. (f) selected candidate solution.

The BLU framework is illustrated with an example in Figure 4.2. First, an SG is formed based on the timing and the topology of the initial clock tree, as illustrated in Figure 4.2(a). Next, the LP formulation in Eq (3.9) is solved to specify delay adjustments and to predict P_{tns} and P_{wns} , which is illustrated in Figure 4.2(b). The latency constraint l_i for each clock sink i is obtained by floating down each delay adjustments to the clock sinks and combining the adjustments with the current arrival time of the clock signal, which is shown in Figure 4.2(c). The maximum latency constraint is denoted l_{max} . Next, an offset off_i equal to, $l_{max} - l_i$, is introduced for each clock sink, as illustrated in Figure 4.2(d). A candidate solution c_i (in van Ginneken's algorithm) is created at each clock sink i with a maximum downstream delay equal to off_i , which is illustrated in Figure 4.2(e). Subsequently, van Ginneken's algorithm is applied and the minimum cost candidate solution that satisfies l_{max} is selected at the root. The latency constraint l_{max} at the root ensures that

each arrival time t_i is smaller than l_i . Moreover, it is guaranteed that at least one candidate solution will satisfy l_{max} at the root of the clock tree, i.e., the initial clock tree. Figure 4.2(f) shows the clock tree after buffer sizing and layer assignment has been performed with respect to the selected candidate solution. Compared with the clock tree in 4.2(a), the clock tree in Figure 4.2(f) will have the same predicted timing quality but smaller capacitive cost.

4.2.2 Relaxing the latency constraints

In this section, the BLU framework is extended to enable further savings in capacitive cost by relaxing the point constraints into latency range constraints. The latency constraints specified by Eq (3.9) are shown in Figure 4.3(a). The relaxed latency range constraints are shown in Figure 4.3(b). The BLU framework specifies the range constraints while guaranteeing that P_{tns} and P_{wns} are not degraded if every arrival time t_i is within the respective latency range. l_i^r is the relaxed latency constraint for clock sink i and $l_i \leq l_i^r$.

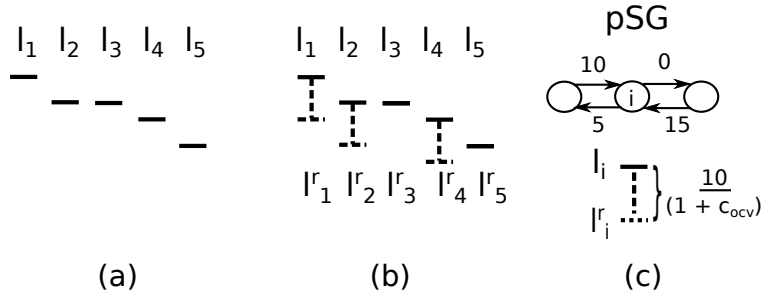


Figure 4.3: (a) Latency constraints. (b) Relaxed latency constraints. (c) Method to find l_i^r .

The method used to find the latency range constraint is illustrated in Figure 4.3(c). First, a predicted slack graph (pSG) is formed [42]. A pSG, captures the predicted slacks in the timing constraints after the delay adjustments specified by Eq (3.9) are realized. Moreover, the predicted slack violations s_{ij} are added to the respective edges in the pSG such that all edges in the pSG are

non-negative. Next, a relaxed latency constraint l_i^r is found, as follows:

$$l_i^r = l_i + \frac{w_{ki}^{min}}{(1 + c_{ocv})}, \quad (4.1)$$

where w_{ki}^{min} is the edge with the minimum weight of all the fan-in edges of node i in the pSG, i.e., $e_{ki} \in E$. The $(1 + c_{ocv})$ factor used to compensate for the increased timing deteriorates δ_i and δ_j .

4.2.3 Tightening the latency constraints

In this section, the BLU framework is extended by allowing negative delay adjustments to be specified in the clock tree, which may improve P_{wns} and P_{tns} , as illustrated in Figure 4.4.

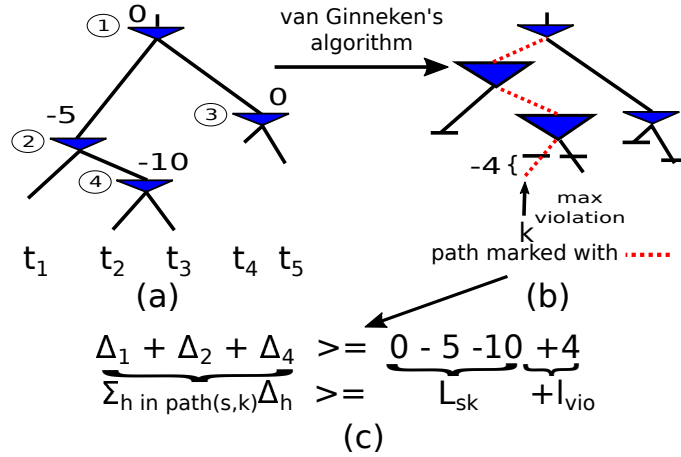


Figure 4.4: (a) Delay adjustments specified by Eq (3.9). (b) Clock tree after van Ginneken's algorithm. l_{max} is violated by the path marked with a red dashed line. (c) Generation of a delay adjustment constraint.

Compared with non-negative delay adjustments that are relatively easy to realize, it may be impossible to physically realize negative adjustments [39]. Consequently, it may be impossible to satisfy the latency constraint l_{max} using Van Ginneken's algorithm, which results in that TNS and

WNS cannot be reduced to the predicted P_{tns} and P_{wns} using traditional CTO. The BLU framework solves this challenge by generating *delay adjustment constraints* to the LP formulation in Eq (3.9), which ensures that only negative delay adjustments that can be realized are specified.

First, Eq (3.9) is solved while allowing both non-negative and negative delay adjustments, which is shown in Figure 4.4(a). Negative delay adjustments are facilitated by replacing the expression $c_t \sum_{k \in B} \Delta_k$ with $c_p \sum_{k \in B} \Delta^+ - c_n \sum_{k \in B} \Delta^-$ in Eq (3.9), where $\Delta^+ \geq 0$ and $\Delta^- \geq 0$ are non-negative and negative delay adjustments, respectively. c_p and c_n are user defined parameters. Next, the remainder of the BLU framework is applied and the resulting clock tree is shown in Figure 4.4(b). If any candidate solution satisfies the latency constraint l_{max} at the root, the same flow as for non-negative delay adjustments is applied (see Section 4.2.1).

If no candidate solution meets the latency constraint l_{max} , the BLU framework selects the candidate solution that is closest to satisfying the latency constraint. Let l_{vio} be the violation of the latency constraint and let $path(s, k)$ be the path from the source to clock sink k that created the largest latency violation, which is illustrated with a dashed red line in Figure 4.4(b). It is straightforward to find the path based on backtracking the candidate solutions generated by van Ginneken's algorithm. Next, a delay adjustment constraint is generated to force the delay adjustments on the $path(s, k)$ to be l_{vio} larger than currently specified, as follows:

$$\sum_{h \in path(s, k)} \Delta_h \geq L_{sk} + l_{vio}, \quad (4.2)$$

where L_{sk} is the sum of the delay adjustments on the path from the source to sink k in the current solution of Eq (3.9). Δ_h are variables in Eq (3.9).

A new set of latency constraints are specified by solving the LP formulation in Eq (3.9) in combination with the delay adjustment constraints in Eq (4.2), which is shown in Figure 4.4(c). The

process is iteratively repeated until the latency constraint l_{max} is satisfied. P_{tns} and P_{wns} are increased in each iteration as additional constraints are introduced to the LP formulation. Latency constraints are also introduced at internal nodes to speed up the convergence process. Violations of latency constraints at internal nodes are accounted for by modifying the $cost_k$ of a candidate to be equal to $cost_i = cap^{tot} + c_{vio} \cdot l_{vio}^{tot}$, where l_{vio}^{tot} is the sum of the violations in the downstream subtree and c_{vio} is a user specified parameter.

4.3 Methodology

The flow of the framework is illustrated in Figure 4.5. First, an initial clock tree is constructed using CTS [30, 34], which is the input to the BLU framework. Next, the BLU framework is performed, as described in Section 4.2. Lastly, CTO is performed to reduce TNS and WNS to P_{tns} and P_{wns} using the techniques in [30]. The high level flow for the BLU framework is outlined below.

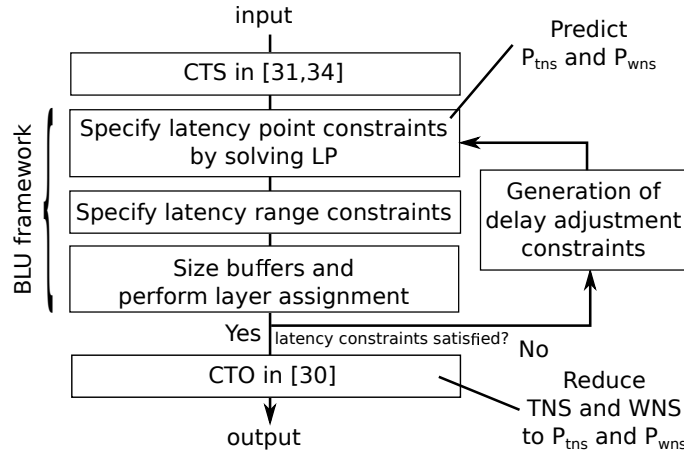


Figure 4.5: Proposed flow for the BLU framework.

A *specify latency point constraints* step is performed to predict P_{tns} and P_{wns} and to specify a set of latency constraints in the form of points. The point constraints are extended into latency ranges

in a *specify latency range constraints* step. Next, *buffer sizing and layer assignment* step is performed using an extension of van Ginneken’s algorithm that utilizes three-dimensional sampling and transition time constraints [40]. If the latency constraints are satisfied, buffer sizing and layer assignment is performed. If the latency constraints are not satisfied, a *generation of delay adjustment constraints* step is performed to introduce delay adjustment constraints. Next, the framework returns to the specify point constraints step. The process is iteratively repeated until all the latency constraints are satisfied.

4.4 Experimental evaluation

The experimental evaluation is performed on a quad core 3.4 GHz Linux machine with 32GB of memory. The proposed algorithms are implemented in C++. IBM ILOG CPLEX is used to solve the LP formulations in the framework [66].

Table 4.1: Circuits in [31].

| Circuit (name) | Sinks (num) | Skew constraints (num) |
|-------------------|----------------|---------------------------|
| s1423 | 74 | 78 |
| s5378 | 179 | 175 |
| s15850 | 597 | 318 |
| msp | 683 | 44990 |
| fpu | 715 | 16263 |
| usbf | 1765 | 33438 |
| dma | 2092 | 132834 |
| pci bridge32 | 3578 | 141074 |
| ecg | 7674 | 63440 |
| des peft | 8808 | 17152 |
| eht | 10544 | 450762 |
| aes | 13216 | 53382 |

The evaluation is performed using the framework proposed in [31], which is an extension of the problem formulation used in the ISPD 2010 contest [20]. The properties of the buffers and the

wires are obtained from the 45 nm technology used in the ISPD 2010 contest. The non-uniform skew constraints and the sink locations are generated using Synopsys DC and ICC. Moreover, there is a transition time constraint at each buffer and clock sink. A summary of the circuits is shown in Table 4.1. We construct and compare eight different tree structures to evaluate the BLU framework.

(1) The UST structure is a clock tree constructed using the CTS engine provided by the authors in [31]. (2) The UST-CTO structure is the structure obtained by applying the CTO in [30] to the UST structure. (3) The UST-P structure is obtained by applying the BLU framework to the UST structure using point constraints, i.e., the framework presented in Section 4.2.1. (4) The UST-P-CTO is the structure obtained by applying CTO to the UST-P structure. (5) The UST-R structure is the UST-P structure obtained by relaxing the point constraints into latency ranges, i.e., the method described in Section 4.2.2. (6) The UST-R-CTO structure is the structure obtained by applying CTO to the UST-R structure. (7) The UST-RT structure is the UST-R structure combined with the technique of tightening the constraints proposed in Section 4.2.3. (8) The UST-RT-CTO structure is obtained by applying CTO to the UST-RT structure.

We evaluate the tree structures in terms of total capacitance, timing performance and run-time. It is well known that the power consumption of a clock tree is highly correlated with the total capacitance. The timing quality is evaluated using TNS and WNS, which are computed using Eq (3.7) and Eq (3.8); P_{tns} and P_{wns} are obtained from solving the Eq (3.9). The arrival times t_i and t_j are obtained using NGSPICE simulations. All tree structures in the experimental results satisfy the same transition time constraints as in [31]. In Section 4.4.1, we evaluate the BLU framework on the clock trees in [31], which only requires non-negative delay adjustments. In Section 4.4.2, we evaluate the BLU framework on the clock trees with strict timing constraints, which utilizes both non-negative and negative delay adjustments.

Table 4.2: Evaluation of various tree structures in terms of total capacitance.

| Circuit | Capacitance (pF) | | | | | |
|---------|------------------|--------------|--------|-----------|--------|-------------|
| (name) | UST [31] | UST-CTO [31] | UST-P | UST-P-CTO | UST-R | UST-R-CTO |
| s1423 | 3.43 | 3.43 | 3.02 | 3.02 | 2.96 | 2.96 |
| s5378 | 5.65 | 5.65 | 5.04 | 5.04 | 4.87 | 4.87 |
| s15850 | 18.09 | 18.85 | 15.84 | 16.86 | 15.77 | 16.62 |
| msp | 1.41 | 1.41 | 1.35 | 1.35 | 1.20 | 1.20 |
| fpu | 1.60 | 1.60 | 1.52 | 1.52 | 1.35 | 1.35 |
| usbf | 4.55 | 4.55 | 4.14 | 4.14 | 4.07 | 4.07 |
| dma | 5.06 | 5.17 | 4.49 | 4.65 | 4.44 | 4.56 |
| pci | 7.65 | 7.65 | 7.02 | 7.06 | 6.71 | 6.71 |
| ecg | 23.44 | 23.66 | 20.39 | 20.96 | 20.54 | 20.84 |
| des | 18.82 | 18.84 | 16.58 | 16.62 | 16.62 | 16.64 |
| eht | 20.14 | 20.14 | 17.85 | 17.85 | 17.62 | 17.62 |
| aes | 151.70 | 152.91 | 132.91 | 135.28 | 132.91 | 135.60 |
| Norm. | 0.99 | 1.00 | 0.89 | 0.90 | 0.87 | 0.87 |

Table 4.3: Evaluation of various tree structures in terms of runtime.

| Circuit | Runtime (mins) | | | | | |
|---------|----------------|--------------|-------|-----------|-------|-----------|
| (name) | UST [31] | UST-CTO [31] | UST-P | UST-P-CTO | UST-R | UST-R-CTO |
| s1423 | 0.0 | 0.1 | 0.0 | 0.1 | 0.1 | 0.1 |
| s5378 | 0.0 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| s15850 | 0.2 | 2.3 | 0.3 | 14.7 | 0.2 | 2.0 |
| msp | 0.0 | 0.0 | 0.5 | 0.0 | 3.7 | 0.1 |
| fpu | 0.0 | 0.2 | 0.7 | 0.0 | 1.2 | 0.0 |
| usbf | 1.0 | 0.2 | 0.4 | 0.2 | 2.4 | 0.2 |
| dma | 1.0 | 2.1 | 1.1 | 2.5 | 10.3 | 2.1 |
| pci | 2.0 | 0.2 | 2.3 | 0.8 | 15.8 | 0.2 |
| ecg | 8.0 | 11.3 | 1.7 | 15.5 | 5.0 | 12.8 |
| des | 4.0 | 1.1 | 1.3 | 0.9 | 3.9 | 0.4 |
| eht | 8.0 | 0.6 | 28.6 | 0.4 | 107.2 | 0.4 |
| aes | 45.0 | 110.5 | 7.2 | 65.7 | 15.4 | 84.3 |
| Norm. | 0.30 | 1.00 | 0.60 | 1.10 | 1.20 | 1.70 |

4.4.1 *Evaluation of positive delay adjustments*

The total capacitance and the runtime of each structure are shown in Table 4.2 and Table 4.3, respectively. The normalized performances with respect to the UST-CTO structures are shown in the row labeled as ‘Norm’. The run-times in Table 4.3 are the run-times of individual synthesis steps. The normalized run-times are the cumulative run-times. The timing performance is not shown because there are no timing violations after CTO.

First, we apply traditional CTO to the UST structures. All timing violations are eliminated at the expense of an average 1% increase in capacitive cost. Compared with the UST structures, the UST-P structures have 10% lower capacitance. The capacitance reduction stems from that van Ginneken’s algorithm assigns interconnects to lower metal layers and that buffers are downsized while still meeting the transition time constraints. Next, CTO is applied to the UST-P structures. The UST-P-CTO structures have 10% lower capacitance than the UST-CTO structures, as the CTO phase only resulted in a small increase in total capacitance. The capacitive improvements come at an expense of a 10% increase in run-time.

Next, we compare UST-R structures with UST-P structures. The table shows that the UST-R structures have 2% lower capacitance than the UST-P structures. The improvement in capacitance is a result of the relaxation of point constraints into range constraints. Ideally, the UST-R structures should have better capacitive performance than the UST-P structures on all circuits. However, the sampling in van Ginneken’s algorithm may result in minor capacitance fluctuations (see circuits *des* and *aes*). After CTO is applied, it can be observed that the UST-R-CTO structures have 3% lower total capacitance than the UST-P-CTO structures. The average run-time of the UST-R-CTO structures is 1.5X higher than the UST-P-CTO structures. The UST-R structures demonstrate that the BLU framework is capable of performing buffer sizing and layer assignment to reduce capacitive cost without degrading timing performance. The improvements are achieved at the

expense of overhead in run-time.

4.4.2 Evaluation of negative delay adjustments

In this section, we focus on how the BLU framework performs on clock trees with non-zero P_{tns} and P_{wns} . The TNS, WNS, P_{tns} , and P_{wns} are respectively labeled ‘TNS’, ‘WNS’, ‘ P_{tns} ’ and ‘ P_{wns} ’ in Table 4.4. The evaluation is performed on the circuits *ecg* and *aes*. The synthesis tool in [31] is used to generate clock trees with different guard bands (labeled ‘M’), which regulates a trade-off between total capacitance and timing performance.

First, traditional CTO is applied to the UST structures. The UST-CTO structures have 72%, 67% lower TNS and WNS than the UST structures, respectively. It can be observed that TNS and WNS after CTO is strongly correlated with P_{tns} and P_{wns} before CTO. On the other hand, the timing improvement result in a 10% increase in total capacitance, which stems from the insertion of delay buffers.

Compared with the UST structure, the UST-R structures have 13%, 23%, 12%, lower total capacitance, P_{tns} , and P_{wns} , respectively. The P_{tns} , and P_{wns} improvements stem from that the layer assignment of interconnects under the bottom most buffers. The TNS and WNS may be improved or degraded. However, the timing performance is expected to be recovered after CTO, as P_{tns} and P_{wns} are improved.

Compared with the UST-CTO structures, the UST-R-CTO structures have 10% lower capacitance and 5% shorter run-time. The TNS is 29% lower and WNS is 13% higher. Compared with the UST-R structures, the UST-RT structures have 43% and 28% lower P_{tns} and P_{wns} , respectively. The timing improvements come from realizing negative delay adjustments. The total capacitance of the UST-RT structures are 8% higher than the UST-R structures. The increase in total capaci-

tance stems from realizing negative delay adjustments using large buffers. It is expected that the UST-RT structures and UST-R structures have similar performances on the clock trees with zero P_{tns} and P_{wns} , as no negative delay adjustments are specified. Compared with the UST-R-CTO structures, the UST-RT-CTO structures have 41%, 29%, lower TNS and WNS, respectively. The timing improvements of the UST-RT-CTO structures come at an expense of a 6% increase in total capacitance and a 27% increase in run-time.

The UST-R-CTO structures and UST-RT-CTO structures demonstrate that the BLU framework is capable of exploring a trade-off between timing quality and capacitive cost using negative delay adjustments. Moreover, the results of the UST-R-CTO structures and UST-RT-CTO structures are notably better than the UST-CTO structures in [31].

4.5 Summary and conclusion

In this chapter, a latency constraint guided buffer sizing and layer assignment framework is proposed. The proposed framework is capable of handling discrete buffer sizes and layer assignments while utilizing useful skew under the influence of variations.

Table 4.4: Evaluation of negative delay adjustments.

| Circuit (name) | M (ps) | Structure (name) | TNS (ps) | WNS (ps) | P_{tns} (ps) | P_{wns} (ps) | Cap (pF) | Run-time (min) |
|----------------|----------|------------------|-------------|-------------|----------------|----------------|-------------|----------------|
| ecg | 30 | UST [31] | 643 | 18 | 0 | 0 | 23.4 | 8.0 |
| | | UST-CTO [31] | 0 | 0 | 0 | 0 | 23.7 | 11.3 |
| | | UST-R | 352 | 18 | 0 | 0 | 20.5 | 5.0 |
| | | UST-R-CTO | 0 | 0 | 0 | 0 | 20.8 | 12.8 |
| | | UST-RT | 414 | 15 | 0 | 0 | 20.5 | 8.0 |
| | | UST-RT-CTO | 0 | 0 | 0 | 0 | 20.8 | 7.1 |
| | 15 | UST [31] | 2218 | 19 | 50 | 2 | 19.7 | 9.2 |
| | | UST-CTO [31] | 929 | 3 | 709 | 2 | 21.3 | 23.4 |
| | | UST-R | 4286 | 24 | 23 | 1 | 17.7 | 6.1 |
| | | UST-R-CTO | 150 | 4 | 86 | 3 | 19.5 | 17.6 |
| | | UST-RT | 1544 | 18 | 0 | 0 | 18.7 | 29.9 |
| | | UST-RT-CTO | 18 | 1 | 0 | 0 | 20.1 | 16.8 |
| | 0 | UST [31] | 15059 | 33 | 5299 | 22 | 17.6 | 4.5 |
| | | UST-CTO [31] | 6259 | 25 | 5919 | 23 | 20.6 | 42.9 |
| | | UST-R | 14865 | 41 | 6121 | 25 | 15.0 | 5.0 |
| | | UST-R-CTO | 7136 | 28 | 6615 | 26 | 17.9 | 21.3 |
| | | UST-RT | 16553 | 35 | 2882 | 19 | 17.4 | 21.3 |
| | | UST-RT-CTO | 3332 | 20 | 3928 | 19 | 20.6 | 35.4 |
| aes | 50 | UST [31] | 1315 | 22 | 0 | 0 | 151.7 | 45.0 |
| | | UST-CTO [31] | 0 | 0 | 0 | 0 | 152.9 | 172.5 |
| | | UST-R | 3045 | 26 | 0 | 0 | 132.9 | 15.4 |
| | | UST-R-CTO | 0 | 0 | 0 | 0 | 135.6 | 84.3 |
| | | UST-RT | 3028 | 26 | 0 | 0 | 132.9 | 19.0 |
| | | UST-RT-CTO | 0 | 0 | 0 | 0 | 135.5 | 86.4 |
| | 40 | UST [31] | 9897 | 33 | 2604 | 12 | 135.6 | 12.1 |
| | | UST-CTO [31] | 3208 | 16 | 3064 | 14 | 145.8 | 110.6 |
| | | UST-R | 25873 | 44 | 2200 | 11 | 119.1 | 21.6 |
| | | UST-R-CTO | 2972 | 17 | 2500 | 14 | 128.9 | 143.4 |
| | | UST-RT | 14118 | 36 | 2151 | 11 | 123.5 | 51.2 |
| | | UST-RT-CTO | 2498 | 14 | 2413 | 13 | 132.8 | 19.2 |
| | 30 | UST [31] | 16041 | 32 | 7095 | 14 | 112.1 | 24.9 |
| | | UST-CTO [31] | 8448 | 18 | 7950 | 15 | 121.6 | 139.0 |
| | | UST-R | 36367 | 47 | 4478 | 13 | 97.8 | 9.2 |
| | | UST-R-CTO | 5697 | 19 | 5186 | 15 | 111.8 | 73.5 |
| | | UST-RT | 15685 | 36 | 2636 | 11 | 102.2 | 56.5 |
| | | UST-RT-CTO | 3569 | 16 | 3330 | 14 | 113.4 | 189.2 |
| Norm. | - | UST [31] | 3.56 | 3.04 | 1.00 | 1.00 | 0.91 | 0.13 |
| | | UST-CTO [31] | 1.00 | 1.00 | - | - | 1.00 | 1.00 |
| | | UST-R | 6.33 | 3.84 | 0.77 | 0.88 | 0.79 | 0.26 |
| | | UST-R-CTO | 0.71 | 1.13 | - | - | 0.90 | 0.95 |
| | | UST-RT | 4.24 | 3.19 | 0.44 | 0.63 | 0.85 | 0.60 |
| | | UST-RT-CTO | 0.42 | 0.80 | - | - | 0.95 | 1.21 |

CHAPTER 5: AN OCV-AWARE CLOCK TREE SYNTHESIS METHODOLOGY

5.1 Introduction

In this chapter, we propose an OCV-aware clock tree synthesis methodology that aims to rethink how to account for OCVs. The key idea of the methodology is to predict the impact of the OCVs early in the synthesis process. This will allow the OCVs to be compensated for using non-uniform safety margins during the initial tree construction. The goal is to only leverage CTO to eliminate minor timing violations arising from modeling errors, which results in that the synthesis flow is almost correct-by-design. In contrast, the state-of-the-art methodologies account for OCVs using aggressive CTO, which results in unpredictable optimization and timing results.

The proposed OCV-aware clock tree synthesis methodology consists of a top-down phase and a bottom-up phase. In the top-down phase, numerous top-level tree topologies are enumerated and pruned. Next, a virtual clock tree is formed to estimate the timing and variations within every timing constraint. Subsequently, non-uniform safety margins are inserted to account for the variations. In the bottom-up phase, each virtual clock tree is refined into a real clock tree by constructing subtrees that connect the clock sinks to the top-level tree. If the constructed subtrees meet a set of latency constraints imposed by the virtual clock tree, the timing constraints are guaranteed to be satisfied by design. Otherwise, the virtual clock tree is updated with improved timing predictions and the process is iteratively repeated. The experimental results demonstrate that the proposed OCV-aware synthesis flow reduces the average TNS and WNS by 90% and 75%, respectively. Moreover, the run-time of the proposed flow is 47% shorter when compared to the CTO based

This chapter is based on the paper that is published at 2021 International Conference on Computer Aided Design conference [72] © 2021 IEEE.

flow.

The remainder of the chapter is organized as follows: Motivation is highlighted in Section 5.2. Methodology is explained in Section 5.3. Experimental results are presented in Section 5.4. The chapter is concluded in Section 5.5.

5.2 Motivation

In this section, we highlight the limitations of the previous works and provide an overview of the proposed methodology.

5.2.1 *Limitations of previous works*

The state-of-the-art synthesis methodology for clock networks is shown in Figure 5.1(a). As the OCVs depend on the timing and topology of the clock tree, uniform safety margins are first inserted into the timing constraints. Next, an initial clock tree is constructed. Given the topology and timing of the initial clock tree, the negative impact of OCVs and the associated timing violations are calculated. Finally, aggressive CTO is iteratively applied to eliminate timing violations. If the timing cannot be closed, expensive manual feedback is performed using ECOs. It is not surprising that ECOs are commonly required because the initial clock tree was constructed without accounting for OCVs.

5.2.2 *Proposed methodology*

The flow of the proposed OCV-aware clock tree synthesis methodology is shown in Figure 5.1(b). The methodology consists of a top-down phase and a bottom-up phase. In the top-down phase,

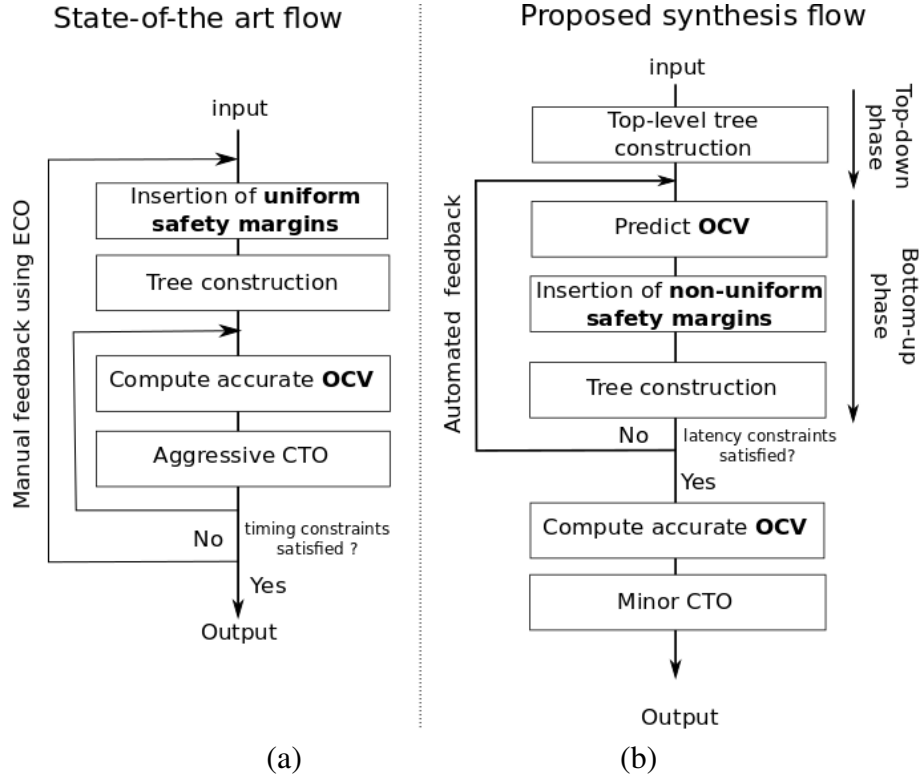


Figure 5.1: (a) State-of-the-art CTS+CTO based synthesis flow. (b) Proposed OCV-aware clock tree synthesis flow.

a top-level tree is constructed. (In the proposed framework, numerous top-level tree topologies are enumerated to explore a larger solution space.) Next, a virtual clock tree is constructed based on the top-level tree to estimate the non-uniform safety margin required in each timing constraint. In the bottom-up phase, subtrees are constructed to connect the clock sinks to the top-level tree. If the subtrees meet the constraints imposed by the virtual tree, the timing is satisfied by design. Otherwise, the timing and impact of OCVs within the virtual tree are updated, and the bottom-up phase is repeated. As the flow accounts for the impact of OCVs early in the synthesis process, there is no need for costly manual intervention using ECOs. The subsequent CTO is only used to

eliminate minor violations introduced by modeling errors¹.

Table 5.1: Holistic guidelines and objectives for different parts of a clock tree.

| Part of Topology | Tree properties | | Objective | Construction method |
|------------------|------------------|---------------|------------------|---------------------|
| | Timing & latency | Hardware cost | | |
| Top-level | Majority | Minority | Minimize latency | Top-down |
| Bottom-level | Minority | Majority | Minimize cost | Bottom-up |

The proposed flow is inspired by the holistic guidelines for clock tree synthesis shown in Table 5.1. A clock tree can be decomposed into a top-part and a bottom-part. The table shows that the top-part of a clock tree stands for a very small portion of the total capacitance and hardware overheads. At the same time, it defines the overall timing of the clock tree. In contrast, the bottom part of the clock tree accounts for a majority of the hardware resources and capacitance. On the other hand, it has a limited impact on the timing. Therefore, we speculate that it would be advantageous to construct the top-part of the clock tree first to define the timing and the negative impact of the OCVs. While fixating the top-level tree (a restriction) may introduce some hardware overheads, the overheads are expected to be small as the cost of the top-part of the clock tree is minor compared with the bottom-part. Next, the bottom-part of the clock tree can be constructed in a cost efficient manner bottom-up. The main challenge to this approach is that it is difficult to select the ideal top-level tree. The larger the specified top-level tree is, the more predictable the OCVs become. On the other hand, the introduced overheads become larger when the top-level tree is larger. Consequently, designs with tight (loose) timing constraints require a larger (smaller) top-level tree. To solve this challenge, the methodology enumerates and prunes out several different top-level tree topologies.

¹The construction of clock trees is traditionally guided by less accurate delay models that can be evaluated quickly. In contrast, CTO is commonly performed using computationally expensive delay models that are more accurate. We refer to the difference in accuracy between the two models as modeling errors.

5.3 Methodology

In this section, we provide the details of the proposed OCV-aware clock synthesis methodology.

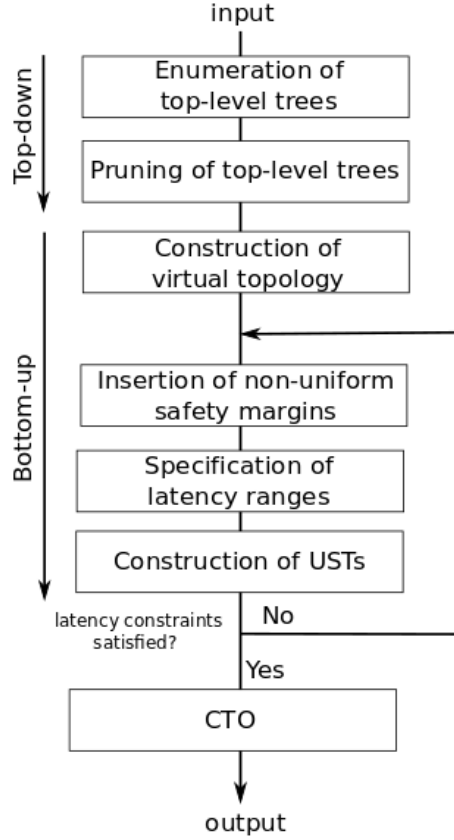


Figure 5.2: Detailed flow of proposed OCV-aware synthesis methodology.

5.3.1 The overview of the framework

The flow of the proposed methodology is shown in Figure 5.2 and illustrated with an example in Figure 5.3. The flow consists of a top-down phase and a bottom-up phase. In the top-down phase, a top-level tree is constructed by enumerating and pruning top-level tree topologies, which is shown in (a) and (b) of Figure 5.3. The details are provided in Section 5.3.2 and Section 5.3.3. In the

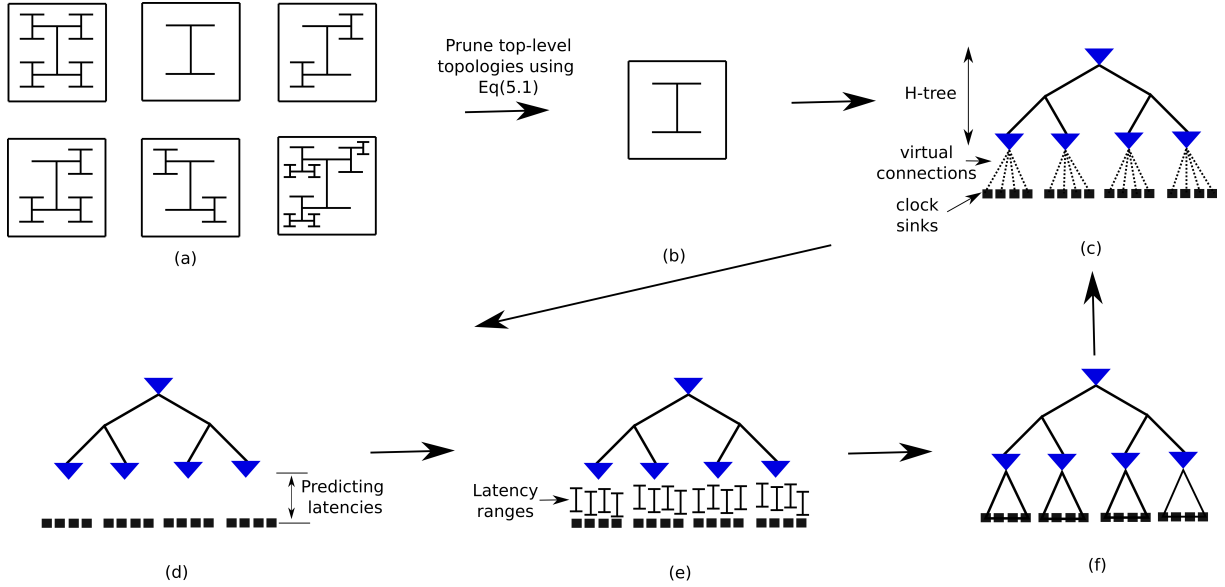


Figure 5.3: (a) Multiple candidate top-level topologies are enumerated. (b) The top-level topology is determined after pruning the candidate topologies. (c) The top-level clock tree and the virtual topology are constructed. (d) The latencies to the clock sinks are estimated using the virtual topology. (e) Non-uniform safety margins are inserted in the skew constraints and a latency range is specified for each clock sinks. (f) Bottom-level subtrees are iteratively constructed by inserting safety margins that are tailored to the topology. If the latency constraints are not satisfied, the construction process returns to (c) with improved timing predictions.

bottom-up phase, a virtual clock tree is first formed based on the top-level tree, which is shown in Figure 5.3(c) and detailed in Section 5.3.4. Next, the timing and OCV impact is estimated based on the virtual tree. The estimates are used to insert non-uniform safety margins in the timing constraints, as shown in Figure 5.3(d) and explained in Section 5.3.5. Next, the timing constraints are converted into latency constraints, which is shown in Figure 5.3(e) and explained in Section 5.3.6. A latency constraint is a lower and upper bound on the arrival time of the clock signal to a clock sink. The conversion is motivated by that it is typically easier to construct clock trees based on latency constraints than skew constraints. Finally, subtrees are constructed bottom-up to connect the clock sinks to the leaf nodes of the top-level tree as shown in Figure 5.3(f) and Section 5.3.7.

After bottom-level subtrees have been constructed, we compute the maximum latency to each clock sink. If all the latency constraints are satisfied, the synthesis process has converged and timing is closed in the presence of OCVs. Otherwise, the synthesis process returns to the insertion of non-uniform safety margins step. However, the timing of the virtual tree is updated with timing from the constructed subtrees. Hence, the timing and the impact of the OCVs can be computed more accurately. In our future work, we plan to explore adjusting the topology of the top-level tree based on feedback from the construction process.

5.3.2 Enumeration of top-level trees

In this section, it is explained how different top-level trees are enumerated. The top-level trees in the proposed methodology are in the form of uniform and non-uniform H-trees [54]. While any type of top-level trees could be used, we select H-trees because they have short latency, which in turn results in smaller OCVs. The H-trees divide the die into multiple rectangular regions. Each leaf node of the top-level tree is expected to drive all the clock sinks in the corresponding region.

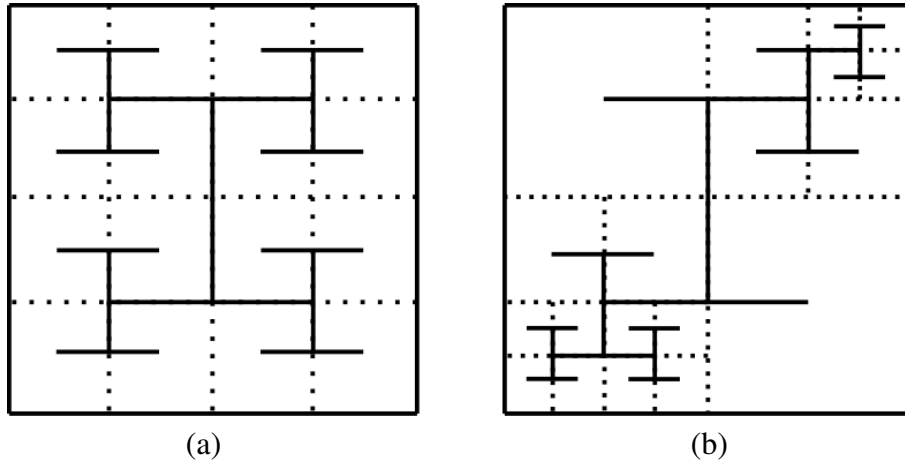


Figure 5.4: The illustration of (a) Uniform H-tree and (b) Non-uniform H-tree

In the proposed framework, all possible uniform and non-uniform top-level H-trees with a maximum of three levels are enumerated. A uniform H-tree have the same amount of levels from the root node to each of the leaf node, which divides the die into a set of uniform regions as shown in Figure 5.4(a). On the other hand, a non-uniform H-tree divides the die into a set of non-uniform regions by constructing irregular level of H-trees, which is illustrated in Figure 5.4(b). In our framework, we primarily use non-uniform H-trees to better adapt the top-level tree to the properties of the input circuit. Note that we consider uniform H-trees are considered a subset of non-uniform H-trees. We also limit the maximum level of an H-tree to be three. The maximum level was selected to balance a trade-off between runtime and solution quality.

5.3.3 *Pruning of top-level trees*

In this section, we prune down the enumerated top-level tree topologies into a single top-level tree topology. An alternative is to keep multiple promising topologies to improve the solution space exploration at the expense of longer synthesis time. The pruning is performed by defining a cost metric for each region. Next, the smallest top-level tree that satisfies a cost constraint on each region is selected.

The cost metric attempts to limit the size of the subtrees that will be connected to each leaf node of the top-level tree. If the subtrees become too large, the magnitude of the introduced OCVs between different sink pairs will be large, requiring excessive safety margins. Consequently, it is advantageous to limit the size of the subtrees. There are several features that determine the size of the subtree constructed from the clock trees within a region of the die. We observe that the most prevalent features are: i) the number of clock sinks, ii) the total sink capacitance, and iii) the region

die area. Therefore, we define the cost metric, as follows:

$$cost = \alpha \cdot N_{sink} + \beta \cdot C_{total} + \gamma \cdot A_{die} \quad (5.1)$$

where N_{sink} is the total number of clock sinks in a region; C_{total} is the total downstream capacitance at the inputs of clock sinks in a region; A_{die} is the area of a region. α , β and γ are the parameters to balance different terms in the cost function.

5.3.4 Construction of virtual topology

In this section, it is explained how the virtual clock tree topology is formed. The root node of each region is located at the mid-point of a region and corresponds to the root node of a bottom-level subtree. To obtain the virtual clock tree topology, first the clock sinks that are located at the same region are clustered. Next, the virtual clock tree topology is formed by inserting a virtual connection from each clock sink to its corresponding root node. The virtual clock tree topology that is obtained after inserting the virtual connections is shown in Figure 5.3(c).

5.3.5 Insertion of non-uniform safety margins

In this section, we detail how the non-uniform safety margins are inserted in the timing constraints. The magnitude of the OCVs in each timing constraint depends on the timing and topology of the clock tree. In our framework, we utilize the virtual clock tree to predict the magnitude of the OCVs. Next, we insert the appropriate non-uniform safety margins.

The timing of the clock tree and the latencies to the clock sinks are first estimated using the virtual clock tree, which is shown in Figure 5.3(c). Let T_h and T_k be the leaf node of the H-tree that is

located at the mid-point of region h and k , respectively. The latency on the path from $CCA(h, k)$ to the clock sinks in region h and k are respectively estimated as follows:

$$\begin{aligned} l_{CCA(h,k),h} &= l_{CCA(T_h,T_k),T_h} + l_h \\ l_{CCA(h,k),k} &= l_{CCA(T_h,T_k),T_k} + l_k \end{aligned} \quad (5.2)$$

where $l_{CCA(T_h,T_k),T_h}$ is the propagation delay on the path from $CCA(T_h, T_k)$ to T_h ; $l_{CCA(T_h,T_k),T_k}$ is the propagation delay from $CCA(T_h, T_k)$ to T_k . l_h and l_k are the average latency of the bottom-level subtrees in the virtual clock tree that are constructed in region h and k , respectively. The average latency is used because the bottom-most subtrees are constructed without any internal restrictions on the topology. Next, the safety margin $M_{(h,k)}$ is inserted in the timing constraints between clock sinks in region h and k , as follows:

$$M_{(h,k)} = c_{ocv} \cdot (l_{CCA(h,k),h} + l_{CCA(h,k),k}) \cdot c_s \quad (5.3)$$

where c_s is a user specified parameter that can be used to scale the inserted safety margins. The default value for the parameter c_s is 1. Finally, the safety margins are inserted into the skew constraints by replacing the δ_i and δ_j terms in Eq (2.4) with $M_{(h,k)}$ in Eq (5.3) as follows:

$$\begin{aligned} lb_{ij} + M_{(h,k)} &\leq t_i - t_j \leq ub_{ij} - M_{(h,k)} \\ i &\in h, j \in k \end{aligned} \quad (5.4)$$

The latencies $l_{CCA(h,k),h}$ and $l_{CCA(h,k),k}$ are obtained using SPICE simulations of the top-level tree. The average subtree latencies l_h and l_k are estimated to be zero in the first iteration. In subsequent iterations, the average latency of the subtree constructed in the previous iteration is used. The latencies of the subtrees are also obtained using SPICE simulations.

5.3.6 Specification of latency ranges

In this section, we explain how to specify latency constraints that ensure that the skew constraints are satisfied. A set of latency constraints consist of a latency range for each clock sinks. A latency range specifies a lower and upper bound on the arrival time of the clock signal to a clock sink. Skew constraints specify a restriction on the relative arrival time of the clock signal in between a pair of clock sinks. The motivation for converting the skew constraints into latency constraints is that it is easier to construct clock trees with respect to latency constraints [57]. The specification of the latency constraints is performed by capturing the skew constraints using a skew constraint graph (SCG). Next, a latency range is specified for each clock sink by formulating and solving an LP problem.

An SCG captures the skew constraints with non-uniform safety margins in Eq (5.4). In an SCG $G = (V, E)$, V is the set of clock sinks and E is the set of skew constraints. Let the flip-flops FF_i and FF_j be represented by vertices i and j , respectively. For each constraint in Eq (5.4), edges e_{ij} and e_{ji} are inserted to the SCG. Let $w_{ij} = ub_{ij} - M_{(h,k)}$ be the weight of edge e_{ij} and $w_{ji} = -lb_{ij} - M_{(h,k)}$ be the weight of edge e_{ji} . Next, we formulate the LP formulation in [57] to specify latency ranges, as follows:

$$\begin{aligned}
& \min c_{tns} \cdot P_{tns} + c_{wns} \cdot P_{wns} + \sum_{i \in V} c_{rf}(x_i^{ub}, x_i^{lb}) \\
& x_i^{lb} \leq x_i^{ub}, \forall i \in V \\
& x_i^{ub} - x_j^{lb} - P_{ij} \leq w_{ij}, \quad \forall (i, j) \in E \\
& P_{tns} = \sum_{(i,j) \in E} P_{ij} \\
& P_{wns} \geq P_{ij},
\end{aligned} \tag{5.5}$$

where x_i^{lb} and x_i^{ub} are the lower bound and upper bound of latency range for sink i , respectively. $f(\cdot)$ is a piece-wise linear function that aims to maximize the length of the latency ranges. More details about the piece-wise linear function is in [57]. P_{ij} is the term that captures the timing violation (negative slack) in the skew constraint between FF_i and FF_j . P_{tns} and P_{wns} denote the predicted total negative slack and predicted worst negative slack, respectively. c_r , c_{tns} and c_{wns} are respectively the parameters to balance the weights of latency range, total negative slack and worst negative slack in the objective. In our framework, c_{tns} and c_{wns} are specified to be significantly larger than c_r to minimize timing violations.

5.3.7 Construction of USTs

In this section, it is explained how subtrees are constructed bottom-up to connect the clock sinks to the top-level tree. A separate subtree construction is performed for the clock sinks assigned to each leaf node in the top-level tree. The subtrees are constructed using the useful skew tree construction algorithm in [57] that is based on the BST construction in [6]. Both algorithms are based on the DME paradigm, where smaller subtrees are iteratively merged to create larger subtrees (see Section 3.2.2).

The BST construction in [6] is based on keeping track of the minimum and maximum delay to a clock sink within a subtree. This makes it easy to merge (or join) two subtrees while meeting a skew bound B . The generalization to UST construction in [57] is based on converting the latency ranges into a virtual minimum and maximum delay offset for each clock sink. Next, USTs can be constructed while only checking that each subtree satisfies a skew bound B . Please refer to [57] for the technical details of specifying the virtual delay offsets. The BST construction is based on the DME paradigm and is performed exactly as in [6].

After all the clock sinks have been connected to the top-level tree, each subtree is simulated using

SPICE simulations. If the latency constraints are satisfied, the flow converges. Otherwise, the framework returns to the insertion of non-uniform safety margins in Section 5.3.5. The latency constraints are required to be checked because the UST construction only guarantees that the clock signal is delivered within latency ranges with an arbitrary offset [57].

5.4 Experimental Evaluations

The proposed framework is implemented in C++ and evaluations are performed using a quad-core 3.4 GHz Linux machine with 32GB of memory. The properties of buffers and wires are obtained from a 45nm technology library in [20]. The clock tree methodology is evaluated using the benchmarks in [7]. The details of each benchmark circuit is shown in Table 4.1. The timing of the constructed clock trees are evaluated with circuit simulations using NGSPICE. The c_{ocv} parameter is set to be 0.085 in our framework.

The structures are evaluated in terms of capacitance and timing quality. The timing quality is evaluated using TNS and WNS. The performance of the proposed framework is evaluated by constructing three different clock tree structures. The UST-U-CTS structure is a useful skew tree constructed with uniform safety margins inserted in the skew constraints [22, 31]. The structure is obtained by performing multiple tree constructions with uniform safety margins of different magnitude. The safety margin configuration that achieves the best timing quality is selected as the UST-U-CTS structure. The UST-U-CTO structure is obtained by applying CTO [30] to the UST-U-CTS structure. The UST-N structures are clock trees obtained using the proposed OCV-aware methodology.

In Section 5.4.1, the configurations within the framework are evaluated. In Section 5.4.2, the proposed methodology is compared with a state-of-art synthesis flow. The evaluation is performed

using the aforementioned tree structures.

5.4.1 Evaluation of framework configurations

In this section, we evaluate the algorithm design and parameter configurations within the proposed methodology.

The iterative subtree construction is evaluated using the *dma* circuit in Figure 5.5. The evaluations are performed by comparing the performance of each constructed clock tree within the iterative construction process.

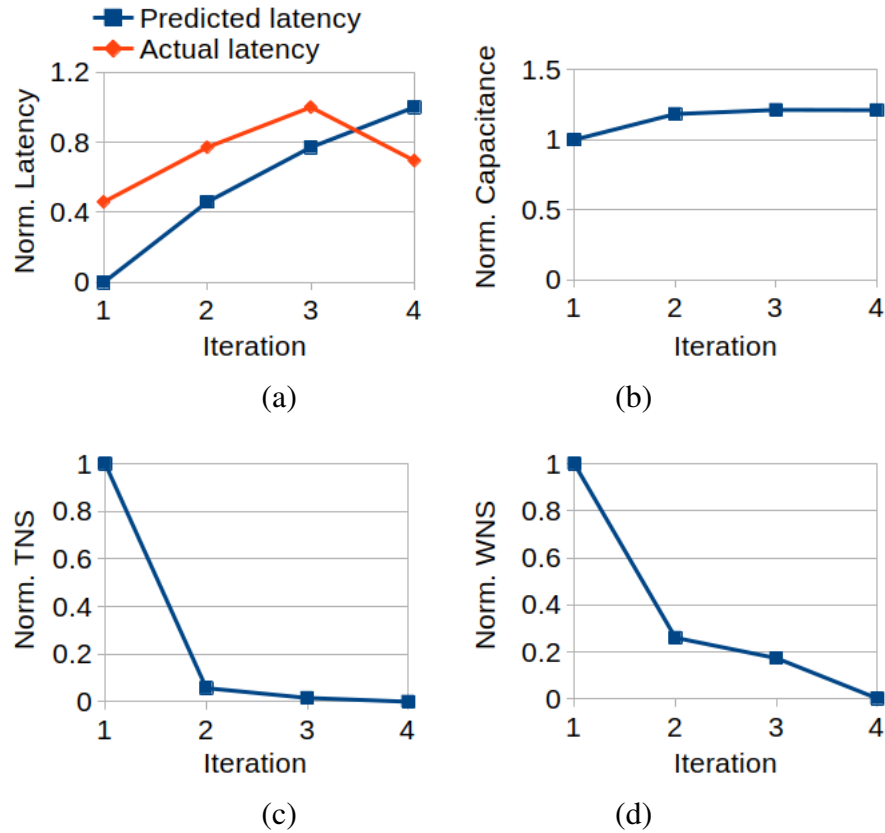


Figure 5.5: Evaluation of the framework in terms of (a) Latency. (b) Capacitance. (c) TNS. (d) WNS.

The predicted latency and the actual latency of a bottom-level subtree with respect to the iteration are shown in Figure 5.5(a). It can be observed that the gap between the predicted latency and the actual latency is gradually reduced until the predicted latency is smaller than the actual latency, which results in that the flow converges. In Figure 5.5(b), we evaluate the capacitance of the constructed clock tree. It can be observed that the capacitance of the clock tree increases with each iteration. This is easy to understand because the framework predicts larger and larger latencies, which results in that gradually larger non-uniform safety margins are inserted in the skew constraints. Intuitively, larger safety margins constrain the tree construction more, which translates into the synthesis of clock trees with higher capacitive overheads. Next, we evaluate the timing quality of the constructed clock trees using TNS and WNS in (c) and (d) of Figure 5.5, respectively. The figures show that the timing quality of the constructed clock tree is improved with each iteration. This stems from that better latency predictions are performed, which results in that safety margins with adequate magnitude are specified. The figure shows that the framework converges to a clock tree structure without timing violations after four iterations, which validates the effectiveness of the proposed framework.

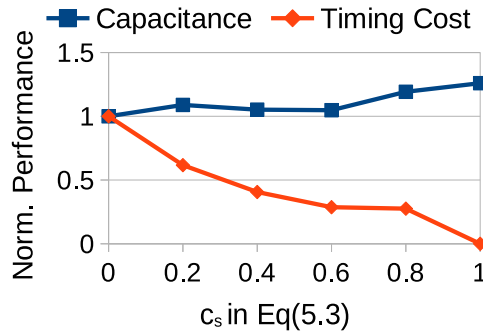


Figure 5.6: Sensitivity to scaling parameter c_s on *ecg*.

We analyze the sensitivity of the framework to the c_s parameter from Eq (5.3) in Figure 5.6. The performance of the framework is evaluated in terms of total clock tree capacitance and the timing

cost on the *ecg* benchmark circuit. The timing cost is the combined cost that consists of both TNS and WNS. The costs are weighted using c_{tns} and c_{wns} in Eq (5.5). The figure shows that the timing cost of the clock tree reduces with the increase of the c_s parameter. On the other hand, the capacitance increases when the c_s increases. This stems from that larger safety margins are specified when c_s is set to be larger, which constrains the clock tree construction.

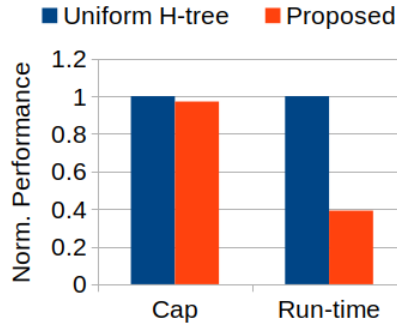


Figure 5.7: Evaluation of top-level tree construction using uniform H-tree vs. non-uniform H-tree.

We evaluate the top-level tree construction based on uniform and non-uniform H-trees in Figure 5.7. The top-level tree construction technique using only uniform H-tree is based on constructing 1-level, 2-level, 3-level H-trees and selecting the structure that provides the best performance in terms of timing quality. The proposed top-level tree construction is based on enumerating and pruning candidate topologies as explained in Section 5.3.2 and Section 5.3.3. The comparison is performed in terms of run-time and total capacitance on the *s15850* benchmark circuit. As it can be observed in Figure 5.7, the proposed technique results in 61% shorter run-time without any capacitive overhead. The improvement in the run-time stems from that the clock tree can be constructed by performing a single synthesis process using the proposed technique, while multiple clock trees are required to be synthesized to select the uniform H-tree that performs best.

Table 5.2: Evaluation of the clock tree structures in terms of performance and synthesis time.

| Benchmark | Structure | Safety margin | | | Capacitance | | | Latency (ps) | TNS (ps) | WNS (ps) | Run-time (min) |
|-----------|--------------------|---------------|-------------|-------------|----------------|--------------|---------------|-----------------|-------------|-------------|-------------------|
| | | min (ps) | avg (ps) | max (ps) | Buffer (pF) | Wire (pF) | Total (pF) | | | | |
| s15850 | UST-U-CTS [22, 31] | 40 | 40 | 40 | 5.4 | 14.8 | 23.1 | 374.0 | 66.1 | 14.8 | 1.7 |
| | UST-U-CTO [30] | ‘-’ | ‘-’ | ‘-’ | 6.6 | 15.0 | 24.6 | 378.5 | 31.8 | 6.1 | 36.3 |
| | UST-N | 11.9 | 17.2 | 29.0 | 4.7 | 9.5 | 17.3 | 311.8 | 0.0 | 0.0 | 6.9 |
| usbf | UST-U-CTS [22, 31] | 30 | 30 | 30 | 1.9 | 5.1 | 8.0 | 214.1 | 37.0 | 11.4 | 2.9 |
| | UST-U-CTO [30] | ‘-’ | ‘-’ | ‘-’ | 2.4 | 5.1 | 8.5 | 230.3 | 0.0 | 0.0 | 21.9 |
| | UST-N | 9.7 | 17.7 | 24.4 | 3.2 | 4.4 | 8.7 | 228.8 | 0.0 | 0.0 | 11.8 |
| ecg | UST-U-CTS [22, 31] | 35 | 35 | 35 | 7.7 | 22.4 | 34.8 | 267.1 | 685.4 | 18.9 | 44.4 |
| | UST-U-CTO [30] | ‘-’ | ‘-’ | ‘-’ | 11.2 | 23.1 | 39.0 | 283.7 | 31.7 | 3.6 | 416.7 |
| | UST-N | 20.5 | 26.6 | 35.5 | 13.3 | 24.1 | 42.1 | 333.4 | 25.4 | 3.0 | 165.1 |
| dma | UST-U-CTS [22, 31] | 20 | 20 | 20 | 1.8 | 4.5 | 7.6 | 180.2 | 144.9 | 8.1 | 5.3 |
| | UST-U-CTO [30] | ‘-’ | ‘-’ | ‘-’ | 2.6 | 4.6 | 8.5 | 180.2 | 25.9 | 4.4 | 29.8 |
| | UST-N | 8.8 | 18.1 | 20.7 | 3.2 | 4.4 | 8.9 | 173.5 | 1.0 | 0.3 | 32.2 |
| pci | UST-U-CTS [22, 31] | 30 | 30 | 30 | 2.5 | 6.5 | 11.2 | 267.1 | 38.9 | 7.3 | 7.3 |
| | UST-U-CTO [30] | ‘-’ | ‘-’ | ‘-’ | 3.0 | 6.5 | 11.8 | 276.1 | 2.8 | 0.7 | 67.9 |
| | UST-N | 9.4 | 18.0 | 24.1 | 3.8 | 6.2 | 12.2 | 218.6 | 1.1 | 0.6 | 40.2 |
| des | UST-U-CTS [22, 31] | 30 | 30 | 30 | 7.4 | 21.3 | 34.2 | 206.4 | 614.8 | 15.0 | 44.8 |
| | UST-U-CTO [30] | ‘-’ | ‘-’ | ‘-’ | 10.2 | 21.8 | 37.5 | 241.0 | 6.9 | 0.7 | 259.8 |
| | UST-N | 15.8 | 17.6 | 28.7 | 10.4 | 19.0 | 34.9 | 236.3 | 1.7 | 0.5 | 91.1 |
| Norm | UST-U-CTS [22, 31] | | | | | | 0.92 | 0.95 | 1.00 | 1.00 | 0.12 |
| | UST-U-CTO [30] | | | | | | 1.00 | 1.00 | 0.13 | 0.21 | 1.00 |
| | UST-N | | | | | | 0.97 | 0.95 | 0.01 | 0.05 | 0.53 |

5.4.2 Comparisons with state-of-the-art

In Table 5.2, we evaluate the performance of the three clock tree structures in terms of total capacitance, timing quality and run-time. We evaluate the total capacitance of the structures because it is well known to be highly correlated with the power consumption. The capacitance breakdown of the constructed clock trees are reported in the ‘Capacitance’ column. The timing quality is evaluated using latency, TNS and WNS. The latency, TNS and WNS of the structures are respectively reported in the columns of ‘Latency’, ‘TNS’ and ‘WNS’. The minimum, average and maximum safety margins that are used to construct the UST-U-CTS and UST-N structures are reported in the ‘Safety margin’ column. The run-time of the synthesized clock trees are reported in the ‘Run-time’ column. The run-times of the UST-U-CTS structure and the UST-N structure in the table

are the run-times of individual synthesis steps. The run-time of the UST-U-CTO structure is the cumulative run-time, i.e., the sum of the run-time of both the UST-U-CTS structure and the CTO process.

The normalized performance in terms of total capacitance, latency and run-time are normalized with respect to the UST-U-CTO structure. The TNS and WNS performance of the structures are normalized with respect to the UST-U-CTS structure. Note that all the synthesized clock trees in Table 5.2 meet the transition time constraint.

Compared with the UST-U-CTS structure, the UST-U-CTO structure have 87% and 79% lower TNS and WNS, respectively. The improvements in the timing quality stem from that CTO is applied to remove the timing violations that are obtained from the UST-U-CTS structure. The improvements in timing quality come at the expense of increasing the capacitance and the latency respectively by 8% and 5%. This is expected because CTO inserts buffers and wires into the clock tree to realize the delay adjustments, which introduces overheads in terms of total capacitance. The run-time of the UST-U-CTO is 88% longer because CTO requires performing a vast amount of timing analysis using SPICE simulations.

Next, we compare the performance of the UST-U-CTO structure with the UST-N structure. The table shows that the UST-N structure has 90% and 75% lower TNS and WNS, respectively. The improvements in the timing quality stem from synthesizing the clock tree by inserting non-uniform safety margins that are tailored to the topology. It is promising that the capacitance and the latency of the UST-N structure are respectively 3% and 5% lower when compared to the UST-U-CTO structure. This stems from that the UST-U-CTO structure is constructed by inserting safety margins that are larger than required for certain skew constraints, which results in constructing larger clock trees. On the other hand, the UST-N structure inserts only the required amount of safety margins that are tailored to the topology, which are reported in Table 5.2. The run-time of the

UST-N structure is 48% shorter when compared to the UST-U-CTO structure. The run-time improvements stem from that the UST-N structure is obtained by synthesizing a single clock tree while the UST-U-CTO structure is obtained by constructing multiple clock trees using different safety margins and selecting the clock tree structure that performs best in terms of TNS and WNS.

Based on the evaluations above, the UST-N structure demonstrates that the proposed methodology in Figure 5.1(b) outperforms the traditional clock tree construction methodology in Figure 5.1(a) in terms of average timing quality, capacitance and run-time.

5.5 Summary and conclusion

In this chapter, we proposed an OCV-Aware clock tree synthesis methodology that is capable of accounting the impact of OCV during the synthesis process. Moreover, the convergence of the proposed synthesis flow is completely automated unlike the state-of-the-art synthesis flow that often requires costly manual intervention in the form of ECOs to close timing. Compared to the state-of-the-art flow, the proposed flow demonstrates higher timing quality with shorter run-time and lower capacitance.

CHAPTER 6: SYNTHESIS OF CLOCK NETWORKS WITH A MODE RECONFIGURABLE TOPOLOGY

6.1 Introduction

This chapter addresses the problem of constructing clock networks for circuits that are required to operate in a low and high performance mode under PVT variations, which has been discussed in Section 3.6. To solve this problem, a clock network with a mode reconfigurable topology (MRT) is proposed for circuits that particularly have only positive-edge triggered sequential elements. The main contributions are summarized, as follows:

- **Reconfigurable Topology:** The proposed MRT structure can reconfigure the topology of the clock network based on the operational mode. In high performance mode, the MRT structure delivers the clock signal using a near-tree topology to provide robustness to variations. In low performance mode, power is saved by reconfiguring the clock network to deliver the clock signal using a tree topology.
- **Circuit Innovations:** The reconfiguration of the topology of an MRT structure is facilitated by OR-gates and a single clock gate. The OR-gates are used to construct alternative paths to each clock sink, which improves the robustness to variations. The advantage of using OR-gates to construct redundant paths is that no short circuit current is introduced, as there is only a single gate driving each net of wires.
- **Automated Synthesis Methodology:** We propose an automated synthesis methodology to construct MRT structures. The methodology balances hardware overhead and robustness to

This chapter is partially based on the paper that is published at 2020 International Symposium on Physical Design Conference [71] © 2020 ACM.

variations.

- **Experimental Validation:** The experimental results show that MRT structures provide higher robustness to variations and up to 25% lower power consumption than state-of-the-art near-tree structures in the high performance mode. In the low performance mode, the power consumption is 35% lower due to the ability of reconfiguring the topology into a tree.

The remainder of the chapter is organized, as follows: preliminaries are given in Section 6.2. Problem formulation is provided in Section 6.3. Previous works are reviewed in Section 6.4. The proposed MRT structure is introduced in Section 6.5 and the methodology is explained in Section 6.6. The experimental results are presented in Section 6.7.

6.2 Preliminaries

Timing constraints for multiple modes: For circuits with multiple operational modes, the timing constraints in Eq (2.1) and Eq (2.2) are required to be satisfied in all modes. The terms in Eq (2.1) and Eq (2.2) have different values at each operational mode due to different clock frequencies and supply voltages are used.

Let B be a lower bound on $|lb_{ij}|$ and $|ub_{ij}|$. Note that lb_{ij} is negative and ub_{ij} is positive for the circuits that are focused on this work. Similarly, let D be an upper bound of the timing deteriorates $(\delta_i + \delta_j)$ for a clock network with a tree topology. D is determined after the clock network is constructed and the timing deteriorations $(\delta_i + \delta_j)$ are known. This can be achieved by performing timing analysis for each operational mode while applying various forms of variations and recording the maximum $(\delta_i + \delta_j)$. Consequently, the skew constraints in Eq (2.4) can be reformulated, as

follows:

$$D - B \leq skew_{ij} \leq B - D. \quad (6.1)$$

Meeting the explicit skew constraints under the nominal conditions is easy ($D = 0$). When $D < B$, it is typically possible to construct a clock tree that satisfies the timing constraints under variations. When $D > B$, it is in general necessary to construct a clock network with a near-tree topology¹. Near-tree clock networks have a tighter upper bound on $(\delta_i + \delta_j)$. The bound is tighter because all pairs of clock sinks with timing constraints are located closer in the near-tree topology [35]. The near-tree structures are commonly synthesized with zero skew to maximize the robustness to variations [20, 21].

Dynamic Voltage and Frequency Scaling (DVFS) : Circuits with multiple modes utilize a lower clock frequency in the low performance modes. The scaled down clock frequency allows the supply voltage to be scaled down to save power without introducing any timing violations. The dynamic power consumption P is calculated, as follows [26]:

$$P = \alpha_{sw} \cdot C_L \cdot V_{DD}^2 \cdot f, \quad (6.2)$$

where α_{sw} is the activity factor, C_L is the switching capacitance, V_{DD} is the supply voltage, and f is the clock frequency.

When the clock frequency is reduced in the low performance modes, timing margins are introduced in the limiting setup and hold time constraints. These timing margins allow the supply voltage to be scaled down until the maximum propagation delay through the combinational logic becomes too long. The relationship between the supply voltage and the propagation delay can be formulated

¹Useful skew can be leveraged to enable clock trees to be constructed when D is only slightly larger than B [32].

as follows [49]:

$$t_{pd} \propto \frac{V_{dd}}{(V_{dd} - V_t)^2}, \quad (6.3)$$

where t_{pd} is the propagation delay of the combinational logic, V_t is the threshold voltage. It can also be understood from the Eq (6.3) that scaling up the supply voltage reduces the propagation delay, which introduces timing margins in the setup and hold time constraints. Consequently, supply voltage can be scaled up to meet the timing constraints in Eq (2.1) and Eq (2.2), which results in increasing the power consumption. Furthermore, several recent studies attempt to squeeze out additional power savings by performing clock tree optimization, where multiple modes are simultaneously optimized to improve both timing and power [8, 39].

6.3 Problem formulation

This chapter is focused on constructing clock networks for circuits that only have positive-edge triggered sequential elements and are required to operate either in a low performance mode or in a high performance mode. The clock frequency and the timing constraints are different for each operational mode. Consequently, clock networks must meet the setup and hold time constraints in Eq (2.1) and Eq (2.2) under PVT variations for each mode of operation. Moreover, the transition time constraints are required to be satisfied. The objective is to minimize the weighted clock power (P_T) using the cost metric as follows:

$$P_T = \beta P_H + \gamma P_L, \quad (6.4)$$

where P_H and P_L are the power consumption in high performance mode and low performance mode, respectively. β and γ are user defined parameters, which are used to balance P_H and P_L in

Eq (6.4).

6.4 The limitations of the previous studies

In Table 6.1, we compare the properties of clock networks with different topologies.

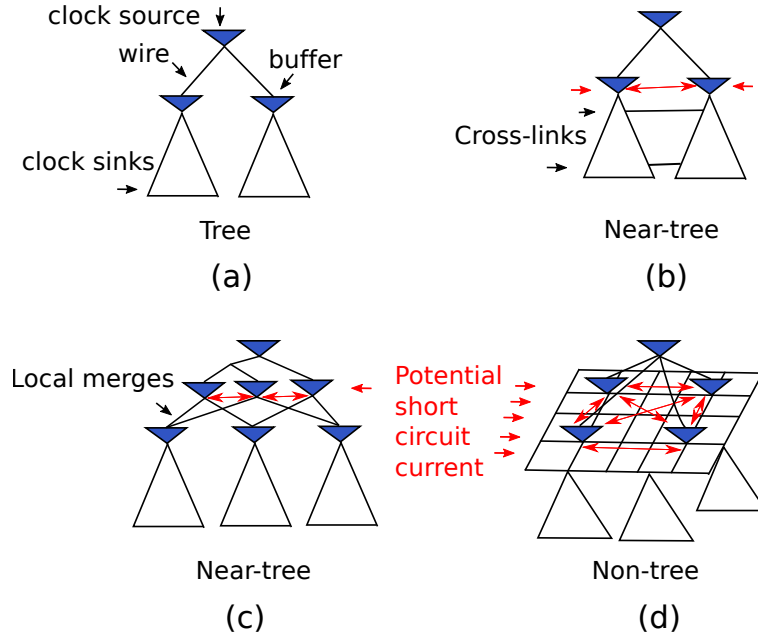


Figure 6.1: Clock networks with different topologies and the illustration of short circuit current on each topology.

As discussed in Section 2.3 and Section 3.6, there is a trade-off between robustness and power consumption for tree and non-tree (or near-tree) structures [9, 13, 15, 19, 27, 35]. The non-tree (or near tree) structures have multiple alternative paths from the clock source to each clock sink, which neutralizes the negative impact of variations. Naturally, the multiple paths introduce hardware overheads with respect to a tree topology. The current trend is to insert the redundancy at internal nodes of a tree structure to balance the robustness improvements with the hardware overheads [9, 35]. Clearly, near-tree structures seem advantageous to non-tree structures as clock

meshes. Nevertheless, near-tree structures are not widely (or at all) adopted in the industry. This is due to that i) near-tree structures introduce short circuit current and ii) near-tree structures can not be constructed using the tree construction algorithms that are integrated into the EDA tools. These challenges stem from that near-tree structures have multiple gates driving the same net of wires, as shown in Figure 6.1. One of the reasons why clock meshes have been successfully adopted in the industry is that symmetric clock trees are traditionally used to drive the mesh grid, which is easy to construct. Moreover, there have already been substantial investments in specialized EDA tools for the synthesis and analysis of such mesh structures. Nevertheless, the concept of sink-splitting was recently proposed to allow near-tree structures to be constructed using tree construction algorithms [35]. The technique involves splitting sinks (or the input pins to buffers) into multiple virtual sinks (or pins). Next, after a tree structure has been constructed to deliver the clock signal to each of the virtual sinks, the virtual sinks are fused to form a near-tree structure. An intuitive idea to reduce the power consumption in the low performance modes is to turn-off part of the non-tree (or near-tree) structure using clock gates while still guaranteeing that the clock signal is delivered using at least one path from the clock source to each clock sink. However, this is impossible to perform using state-of-the-art non-tree (or near-tree) structures because it would create static short circuit current.

Table 6.1: Comparison between clock networks with tree, near-tree, and non-tree topologies.

| Topology | Work | Robustness to variations | Power | Based on tree structure | Compatible with | |
|-----------------|--------------|--------------------------|--------------|-------------------------|-----------------|-----------------|
| | | | | | Voltage scaling | Reconfiguration |
| Tree | [1, 3, 23] | low | small | Yes | Yes | No |
| Near-tree | [9, 15, 35] | high | medium | No | Yes | No |
| Non-tree | [19, 25, 27] | high | very large | No | Yes | No |
| MRT (near-tree) | This work | high | medium | Yes | Yes | Yes |

In this chapter, we propose a clock network with a mode reconfigurable topology (MRT) with

properties as labeled in Table 6.1. The MRT structure has similar performance as near-tree structures in earlier studies but multiple paths are joined using OR-gates instead of wires. Consequently, there is only one gate driving each net of wires. Therefore, it is easy to understand that the MRT structure can be reconfigured into a tree topology by gating the clock signal in part of the structure. Moreover, the MRT structures can be constructed using the tree construction algorithms that are adopted in the EDA tools. Consequently, the MRT structure is based on tree structures.

6.5 Proposed MRT Structure

In this section, we introduce the properties of the proposed MRT structure. The overview of the MRT structure is given in Section 6.5.1. In Section 6.5.2, it is explained how MRT structure in the form of near-tree topology improves robustness to variations. In Section 6.5.3, we explained how MRT structures save power in the low performance mode.

6.5.1 Overview of the MRT structure

In this section, the proposed MRT structure is explained using the Figure 6.2. The MRT structure is constructed to operate in two modes using buffers, wires, OR-gates and a clock gate. The structure consists of two top-level trees that are joined using OR-gates to drive the bottom most subtrees. The input to the MRT structure is a mode control signal and the clock signal from the clock source. The mode control signal specifies the mode that the circuit should operate. In the high performance mode 1, the clock gate is transparent and the entire MRT structure is used to deliver the clock signal to the clock sinks, i.e., the topology is in the form of a near-tree. In the low performance mode 2, the clock signal is gated and the part of the MRT structure that is used to deliver the clock signal to the clock sinks is in the form of a tree topology. In Figure 6.2, a few gates and subtrees are labeled

'1' and '1 2' to indicate the modes where the respective components are used to deliver the clock signal.

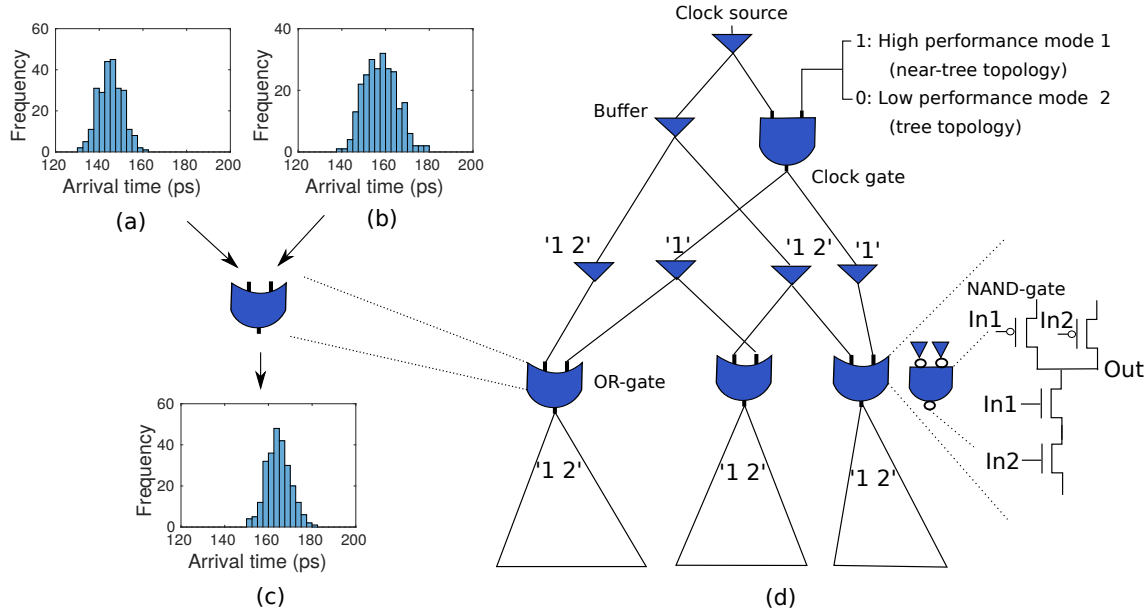


Figure 6.2: The arrival time distributions (for the positive-edge of the clock signal) at different points with respect to an OR-gate are shown in (a), (b) and (c). The variance of the arrival time distribution at the output of the OR-gate is tighter than that of the variance at the inputs, which demonstrates that OR-gates improve the robustness to variations. (d) The proposed MRT structure is constructed using OR-gates to provide robustness to variations.

6.5.2 Improving the robustness in high performance modes

In the high performance mode, the robustness to variations is improved by the two top-level trees that are joined together using the OR-gates. However, OR-gates only improve the robustness of the positive-edge of the clock signal. Nevertheless, positive-edge triggered sequential elements have loose timing constraints on the negative-edge of the clock signal. Therefore, it can be concluded that the MRT structure improves the overall robustness to variations.

The robustness of the positive-edge is improved because the OR-gates are implemented by NAND-gates with one INV-gate connected to each their inputs, as illustrated in the bottom right part of Figure 6.2. The two input NAND-gate has two parallel PMOS transistors that may charge the net attached to the output of the NAND-gate. Under nominal conditions, the MRT structure is designed such that the clock signal arrives at the same time to the separate input pins of each OR-gate. Hence, both PMOS transistors in the NAND-gate will equally contribute to charging the downstream net. Under variations, it is expected that the arrival time of the clock signal to the separate input pins will be different. The end-result is that the PMOS transistor that is turned-on first (last) will charge the net more (less) compared within the nominal case. A simple but effective model to estimate the robustness improvement of an OR-gate is to approximate the effective arrival time of the clock signal to the OR-gate with the mean of the two separate arrival times, i.e., an OR-gate will improve the robustness to variations by averaging out the negative effect of the variations. Of course, the model only holds when the last PMOS transistor to be turned-on is activated before the net downstream of the OR-gate is completely charged by the other PMOS transistor. Therefore, the proposed OR-gate is preferred to the traditional OR-gate consisting of a NOR-gate connected in series with an INV-gate. The explanation is that the capacitance of the internal node in the traditional OR-gate is smaller than the capacitance of the downstream subtree, which means that the internal node will be charged faster, and fewer variations can be neutralized.

To confirm that OR-gates improve the robustness of the delivery of the positive-edge of the clock signal, the structure in the figure is simulated using NGSPICE [14] while applying various forms of correlated variations such as wire width, channel length, supply voltage, and temperature variations using the Monte Carlo framework in [9]. More details about the variations are provided in Section 6.7. The arrival time distributions at the inputs of an OR-gate are shown in (a) and (b) of Figure 6.2. It is concluded that the OR-gates improve the robustness to variations because the variance of the arrival time distribution at the output of the OR-gate is close to half of the variance

of the distribution above the OR-gate, which is shown in Figure 6.2(c).

In terms of the negative-edge of the clock signal, it can be understood that the OR-gates act as a max operator using the same simple model. Although we are targeting designs with only positive-edge triggered sequential elements in this work, we experimentally observed that the MRT structures also deliver the negative-edge of the clock signal with higher robustness than a clock tree (see Section 6.7.1.4). The explanation is that the max operator effectively slows down the clock signal on paths where the propagation delay has been sped-up by variations more than it further slows down the propagation delay of paths that have become slower due to variations. Nevertheless, we intend to perform further analyses of the impact on the setup and hold times for circuits with negative-edge and dual-edge triggered sequential elements in our future work.

6.5.3 Reducing power in low performance modes

To save power in the low performance mode, the MRT structure can be reconfigured from near-tree topology to a tree topology based on the active mode. The reconfiguration can be performed without introducing any short circuit current because each net of wires is connected to only a single driving gate. In fact, this is the first near-tree (or non-tree) structure that can be configured based on the active mode. However, the reconfiguration introduces a trade-off between supply voltage scaling and clock network switching capacitance, which is shown in Table 6.2.

Table 6.2: Comparison between DVFS and topology reconfiguration combined with DVFS.

| Technique | V_{dd} | C_{clk} | Power |
|------------------------|-------------------|----------------|-------------------|
| DVFS | very small | full | small |
| DVFS + Reconfiguration | small | reduced | very small |

The dynamic power consumption of a VLSI circuit is calculated, as follows:

$$P = C_{comb} \cdot V_{DD}^2 \cdot f \cdot \alpha_{comb} + C_{clk} \cdot V_{DD}^2 \cdot f \cdot \alpha_{clk}, \quad (6.5)$$

where f is the clock frequency; C_{comb} and C_{clk} are the total capacitance of the combinational logic and the clock network, respectively. α_{comb} and α_{clk} are the activity factors of the combinational logic and clock network, respectively. V_{DD} is the supply voltage.

Traditional DVFS technique is performed by scaling down the supply voltage to the minimum possible value that the timing constraints are guaranteed to be satisfied (as explained in Section 6.2). For MRT structures, we propose to combine topology reconfiguration with DVFS in the low performance mode. First, the MRT structure is reconfigured into a tree topology. Second, DVFS is applied to the MRT structure.

The reconfiguration of the topology into a clock tree reduces the clock tree switching capacitance (C_{clk}) by turning-off a large portion of the clock network. However, the reconfiguration may introduce small nominal skews between pairs of clock sinks because the MRT structure has OR-gates with a different number of inputs, which have different propagation delays. Therefore, the timing margins available for voltage scaling are reduced by the reconfiguration. Consequently, the supply voltage will be slightly higher compared with only applying voltage scaling. Nevertheless, the overall power is expected to be significantly lower. The explanation is that the propagation delay of any gate is proportional to Eq (6.3). Therefore, large timing margins are required to only slightly reduce the supply voltage when the supply voltage V_{DD} is close to the V_t . Consequently, the reduction in the C_{clk} is expected to translate into large power savings than reducing V_{DD} slightly more.

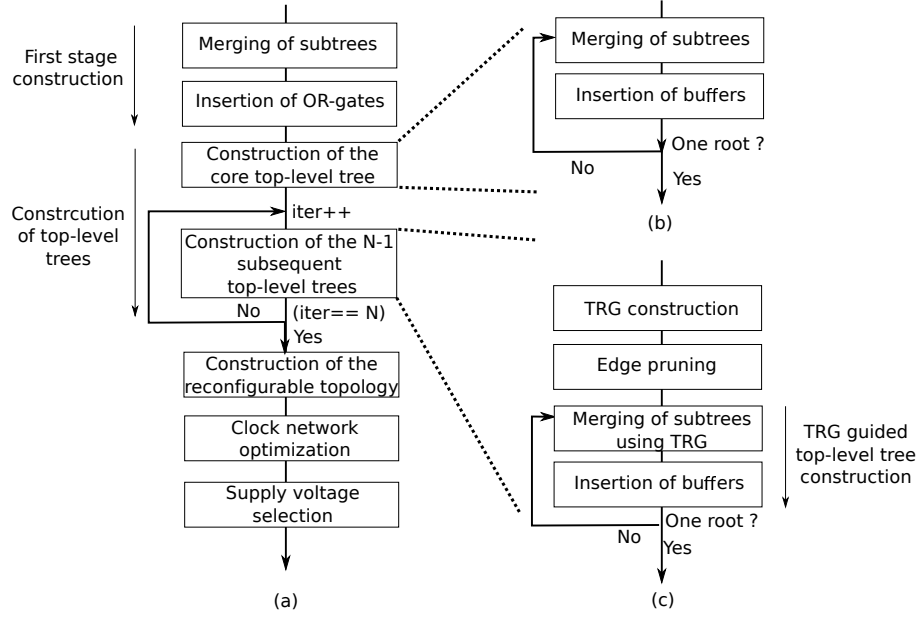


Figure 6.3: (a) Flow for constructing MRT structures. (b) Flow for construction of the core top-level tree. (c) Flow for construction of the $N - 1$ subsequent top-level trees.

6.6 Methodology

The flow for constructing the proposed MRT structures is shown in Figure 6.3. The flow is illustrated with an example in Figure 6.4. MRT structures are constructed bottom-up stage-by-stage using a modification of the classic zero-skew DME paradigm [2, 4, 5]. The details of the DME paradigm is previously explained in Section 3.2.2. A stage consists of a device (a buffer or an OR-gate) driving a subtree of wires connected to the stage-sinks. The stage-sinks of the first stage are the clock sinks. The stage-sinks of any other stage are the input pins of the devices driving the previous stage. The bottom-up construction phase is followed by a top-down embedding of internal nodes using the DME paradigm.

MRT structures consist of first stage subtrees that are driven by N top-level trees, where $N \geq 2$ is a user specified parameter. The first top-level tree is constructed with the objective of minimizing

wirelength while connecting each clock sink to the clock source. The first top-level tree is referred to as the core top-level tree. The remaining $N-1$ top-level trees are constructed to improve the robustness (see details in Section 6.6.2). The N top-level trees are joined to drive the first stage subtrees using OR-gates. We choose to insert OR-gates to drive the subtrees of the first stage because such procedure is known to balance robustness and capacitive cost [9]. The robustness provided by inserting redundancy is higher when placed closer to the clock sinks. At the same time, inserting redundancy closer to the clock sinks introduces larger hardware and performance overheads. The roots of the N top-level trees are connected together using a clock gate in order to enable the topology reconfiguration.

The flow in Figure 6.3 shows that the first stage of an MRT is constructed by merging subtrees (see Section 3.2.2) and inserting OR-gates to drive the subtrees (see Section 6.6.1). Next, the construction of the core top-level tree step is performed. This is followed by performing the construction of $N-1$ subsequent top-level trees step (see Section 6.6.2). The top-level trees are subsequently joined, gated, and routed to the clock source (see Section 6.6.3). After constructing the reconfigurable topology, the nominal skew of the MRT is tuned close to zero using a clock network optimization step (see Section 6.6.4). Lastly, DVFS is applied to determine the supply voltage for both modes of operation (see Section 6.6.5).

6.6.1 Insertion of OR-gates

In this section, we explain the insertion of OR-gates step. The input to this step are the first stage subtrees. For each subtree, the minimum sized OR-gate (with N input pins) that can drive the subtree without violating the transition time constraint is inserted at the root. The parameter N is limited to 4 in order to limit the size of the NMOS transistors within each NAND-gate (inside each OR-gate). A set of first stage subtrees are shown in Figure 6.4(a). After the OR-gate insertion step,

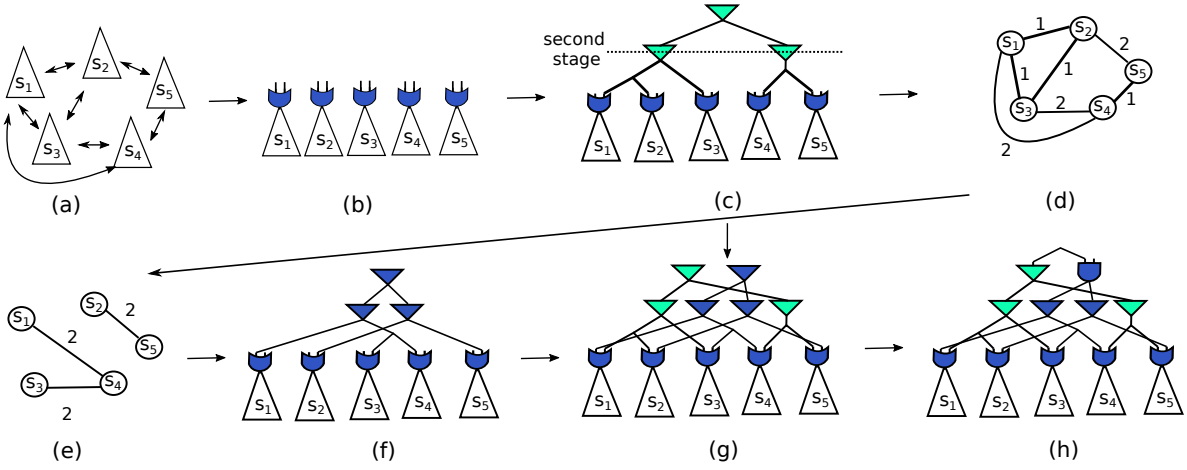


Figure 6.4: Example flow for the construction of an MRT structure when $N = 2$. (a) First stage subtrees. (b) Driver devices are inserted. (c) Core top-level clock tree is constructed. The driver devices used to construct the core-top level tree are colored in green. (d) The Topology Relation Graph (TRG) is constructed with respect to the topology. (e) The TRG after edge pruning is applied. (f) The second top-level clock tree is constructed using TRG. (g) The clock network after the construction of top-level trees shown in (c) and (f). (h) The clock network after constructing the reconfigurable topology.

the resulting subtrees are shown in Figure 6.4(b).

6.6.2 Construction of top-level clock trees

In this section, we explained how the N top-level trees in an MRT structure are constructed. The core top-level tree is constructed using the traditional zero-skew clock tree construction algorithm, which involves iteratively merging subtrees with the objective of minimizing wirelength. Next, the remaining $N-1$ top-level trees are constructed with the goal of improving the robustness to variations. This is achieved by constructing the subsequent $N-1$ top-level trees with the following two objectives:

- i) Clock sinks (with skew constraints between them) that are distant in the base (first) top-level

tree should be placed close in at least one of the subsequent top-level trees.

- ii) Subtrees that are close (topology-wise) in the core top-level tree should not be close in any subsequent top-level tree.

The motivation for the first objective is that larger variations are introduced in skew constraints between clock sinks that are distant in the topology. Therefore, the robustness can be significantly improved by placing such clock sinks (or subtrees) close in a subsequent top-level tree. The motivation for the second objective is that clock sinks that are close in the initial tree topology have no need of improving the robustness to variations. Therefore, the construction of such top-level trees should be avoided in order to minimize hardware overheads. Moreover, the construction of clock networks that are larger than necessary may actually degrade the overall robustness, as larger clock networks are more vulnerable to variations [9]. We propose to achieve the two aforementioned objectives by introducing a topology relation graph (TRG). Next, the TRG is used to guide the construction of the last $N-1$ top-level trees. The TRG is used to capture both skew constraints and the distance between pairs of subtrees in the tree topology. The first objective is achieved by encouraging subtree pairs with large edge weights in the TRG to be joined. The second objective is achieved by pruning edges in the TRG.

The details of the TRG construction are explained in Section 6.6.2.1. The edge pruning step is given in Section 6.6.2.2. The TRG guided clock tree construction is provided in Section 6.6.2.3. These steps are performed iteratively to construct the last $N-1$ top-level trees.

6.6.2.1 Construction of Topology Relation Graph

A TRG $G = (V, E)$ is constructed with respect to the first stage subtrees. The vertices V represent the first stage subtrees. The weighted edges E represent the distance in the topology between

pairs of subtrees that are sequentially related. A pair of clock sinks are sequentially related if there is a skew constraint between them. A pair of subtrees are called sequentially related if they correspondingly contain a pair of sequentially related clock sinks. The edge weight between sequentially related subtrees is defined to be equal to the minimum number of buffers that must be traversed to find the closest common ancestor (CCA) of the subtree pair. For example, in Figure 6.4(d) the edge weight between s_2 and s_3 is one because only one CCA buffer is required to be traversed. On the other hand, the edge weight between s_2 and s_5 is 2 because two buffers are required to be traversed. The arrows in the Figure 6.4(a) illustrate that the corresponding first stage subtree pairs are sequentially related. The TRG with respect to the topology of the core top-level tree in Figure 6.4(c) is illustrated in Figure 6.4(d).

After multiple top-level trees have been constructed, it can be understood that each subtree pair may have multiple CCAs. When forming the TRG for the construction of the third or fourth top-level tree, a candidate edge weight value is computed with respect to each of the previous top-level trees. Next, the weight for each edge in the TRG is set to the minimum of the candidate values.

6.6.2.2 *Edge pruning*

In this section, the edge pruning technique is explained. The edge pruning is performed by removing all edges in a TRG with a weight less or equal to one. This corresponds to avoiding to join subtrees that are joined in the second stage of the initial tree topology. Figure 6.4(e) shows the resulting TRG after performing edge pruning in the TRG of Figure 6.4(d). After pruning the edges, there may exist vertices without any incident edges, which means that the clock signal is already delivered with sufficient robustness. Consequently, there is no need for the corresponding subtree to participate in the subsequent top-level tree construction. This is facilitated by reducing the number of input pins of the corresponding OR-gate by one.

6.6.2.3 TRG guided Tree construction

In this section, we explained how the TRG is used to guide the construction of the top-level trees. The input to the TRG guided clock tree construction step is one input pin from each OR-gate that has been inserted to drive the first stage subtrees. The output is a top-level tree driving the input pins. The top-level tree is constructed by iteratively performing the merging of subtrees and the insertion of buffers steps, which is shown in Figure 6.3(c). However, when selecting subtrees to be merged in the NNG, both the required wirelength and topology distance are considered (see details below). Moreover, if no edge exists between a pair of vertices in the TRG, the corresponding subtrees in the NNG cannot be merged. The top-level tree constructed by the TRG guided tree construction is shown in Figure 6.4(f). The clock network structure that is obtained after the construction of two top-level trees are shown in Figure 6.4(g).

The conventional zero-skew tree construction methodology is based on joining the subtrees that require the minimum wirelength to be merged. In the TRG guided top-level tree construction, the merging of the subtrees is based on a cost function that accounts for both wirelength and distance in the topology. Let the cost of merging a subtree i and j be denoted c_{ij} and defined, as follows:

$$c_{ij} = \alpha d_{ij} - (1 - \alpha) p_{ij} \quad (6.6)$$

where d_{ij} is the wirelength required to join the subtrees i and j using a zero-skew merge. p_{ij} is the corresponding edge weight in the TRG. α is a user defined parameter to balance the importance of the two objectives. It can be understood that setting α to 1 would result in constructing a top-level tree using the traditional zero-skew metric. In our framework, we set α to be small (0.01) such that the cost metric always prioritizes topological distance over the wirelength. The wirelength component is used as a tie-breaker when there exist multiple edges with the same weight (p_{ij}).

Note that when two subtrees are merged, the corresponding vertices in the TRG must be merged into a single vertex. The incident edges of the two vertices are inserted between the newly formed vertex and the corresponding adjacent vertices (without modifying the edge weights). This may result in two parallel edges with different edge weights that are introduced between the newly formed vertex and a single adjacent vertex. For such cases, the parallel edges are replaced with a single edge with a weight equal to the minimum of the two edge weights.

6.6.3 Construction of the reconfigurable topology

In this section, we explained how the N constructed top-level trees are joined into an MRT structure. The top-level trees must be joined such that the MRT structure can be reconfigured between a tree and a near-tree topology. The reconfiguration is performed by gating the clock signal in part of the structure using a clock gate. In contrast with traditional clock gating that aims to completely turn-off a portion of the clock network, the reconfiguration based clock gating is performed such that there still exists at least one path from the clock source to each clock sink. Let the constructed top-level trees be denoted t_1 to t_N . To facilitate the reconfiguration between the tree and near-tree topology, we proposed to connect the top-level trees as illustrated in Figure 6.5. In high performance mode, the MRT structure has a near-tree topology and all the top-level trees t_1 to t_N deliver the clock signal to clock sinks. When the clock signal is gated in the low performance mode, the top-level trees t_2 to t_N are turned-off and only the core top-level tree t_1 delivers the clock signal to all clock sinks. Note that t_1 is the only top-level tree that is guaranteed to have a path to all the first stage subtrees.

The structure in the figure is constructed by joining t_2 to t_N using zero-skew merges. Next, we insert a clock gate to drive the resulting tree before merging the clock gate with the core top-level tree t_1 using a zero skew merge. Finally, we connect the resulting structure to the clock source. Of

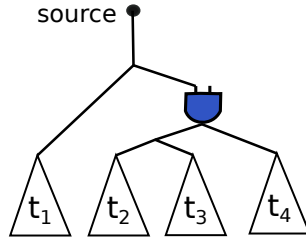


Figure 6.5: The reconfigurable topology

course, the buffers are inserted to balance delay if a zero-skew merge cannot be performed without violating the transition time constraints. This results in the MRT structure in Figure 6.4(h) which is formed from the two top-level trees in Figure 6.4(g). Next, a clock network optimization step in Section 6.6.4 is performed to balance the skew in the structure.

6.6.4 Clock network optimization

Clock network optimization (CNO) is performed to minimize the nominal skew in a clock network. It is typically very difficult to construct large clock networks with small nominal skew without utilizing clock network optimization. For MRT structures, clock network optimization is performed with the objectives of: i) minimizing the nominal skew close to zero and to ii) minimizing the skew between input pins of the same OR-gate, which improves the robustness to variations.

State-of-the-art clock tree optimization techniques are based on specifying and realizing delay adjustments to remove the timing violations, which is also essential to meet the useful skew constraints [12]. The specified delay adjustments are realized by inserting delay buffers and detour wires as explained in Section 3.3. For MRT structures, clock network optimization is performed using a two-phase approach. In the first phase, the input pins of the OR-gates are viewed as the clock sinks and the skew between pins of the same OR-gate is minimized by specifying and real-

izing delay adjustments, which improves the robustness of the MRT structures to variations. In the second phase, the nominal skew is minimized by specifying and realizing delay adjustment below the OR-gates. Note that it is easy to adapt clock tree optimization algorithms to the MRT structure because there is only a single gate driving each net, which enables conventional EDA algorithms and tools to be used.

6.6.5 *Supply voltage selection*

In this section, it is explained how the supply voltage is determined at each operational mode. The objective is to select the supply voltage that meets the timing constraints with the minimum possible power consumption while operating with the specified clock frequency for the operational mode.

After applying CNO to the MRT structure, 250 Monte Carlo simulations are performed to obtain the upper bound on the timing deteriorations (D) that the MRT structure achieves. Next, we tighten the timing margins by the amount of D to provide guardbands to variations. Finally, we perform DVFS by applying a binary search to determine the minimum supply voltage that meets the timing constraints. In our framework, the supply voltage is allowed to be specified in the range of V_t to 1.02V. Note that the supply voltage can never be scaled down too close to the threshold voltage because there are no excessive timing margins available in the evaluated designs.

6.7 Experimental Results

The proposed methodology was implemented in C++ and experimental evaluation was performed on a quad-core 3.4 GHz Linux machine with 32GB of memory. The properties of the buffers, the OR-gates and the wires were obtained from a 45 nm technology library [20]. The benchmarks

Table 6.3: Benchmarks in [7].

| Circuit (name) | Sinks (num) | Skew constraints (num) | Clock period (ps) | | Clock cycle for guardbands (%) |
|-------------------|----------------|---------------------------|-------------------|-------|-----------------------------------|
| | | | T_H | T_L | |
| s1423 | 74 | 78 | 200 | 1000 | 10 |
| usbf | 1765 | 33438 | 200 | 1000 | 10 |
| dma | 2092 | 132834 | 200 | 1000 | 10 |
| pci bridge32 | 3582 | 141074 | 200 | 1000 | 10 |
| ecg | 7674 | 63440 | 200 | 1000 | 10 |
| des peft | 8808 | 17152 | 200 | 1000 | 10 |
| aes | 13216 | 53382 | 200 | 1000 | 10 |

in [7] were used to synthesize our clock networks. The details about benchmark circuits (number of clock sinks, number of skew constraints, and clock period) are shown in Table 6.3. The transition time constraint for all the structures was set to 100 ps. The clock periods correspond to a clock frequency of $5GHz$ and $1GHz$ in the high and low performance modes, respectively. In each mode of operation, a portion of the clock cycle was reserved for guardbands to variations, which is shown in the ‘Clock cycle for guardbands’ column in Table 6.3.

Table 6.4: The properties of the structures.

| Structure | Redundancy insertion using | Synthesis is guided by | Topology |
|------------------|-------------------------------|---------------------------|-----------|
| Tree [1, 4, 23] | ‘-’ | NNG | Tree |
| Near-Tree-SS [9] | Sink splitting | NNG + Local merges | Near-tree |
| Near-Tree-OR | OR-gates | NNG + Local merges | Near-tree |
| MRT | OR-gates | NNG + TRG | Near-tree |

To demonstrate the effectiveness of the proposed framework, four different structures are constructed and evaluated. The properties of each structure are summarized in Table 6.4. For each structure, the techniques used to insert redundancy and guide the synthesis process are reported along with the topology. Specifically, we construct one tree structure (labeled ‘Tree’), two near-tree structures (labeled ‘Near-tree-SS’ and ‘Near-Tree-OR’), and the proposed MRT structure (la-

beled ‘MRT’). The tree structures were constructed using the methodology in Section 3.2.2, which is similar to that for the zero-skew clock trees in [1, 4, 23]. The Near-Tree-SS structures were constructed to mimic the locally-merged structures in [9], which are illustrated in Figure 6.1(c). The locally-merged structures were selected because they were reported to outperform clock trees with cross-links and clock meshes in [9]. The Near-Tree-OR structures were constructed to mimic the Near-Tree-SS structures. However, alternative redundant paths were joined using OR-gates instead of using sink-splitting. The MRT structures were constructed using the proposed flow in Figure 6.3(a). Based on the experiments performed in Section 6.7.1, the MRT structures were constructed using 2 top-level trees, i.e., the MRT structures were obtained using $N = 2$.

The structures are evaluated in terms of power consumption, supply voltage and timing quality that are obtained in high and low performance modes. The power consumption is evaluated using circuit simulations. The timing quality of the structures are evaluated by performing Monte Carlo simulations using NGSPICE. Each structure is simulated with 250 Monte Carlo simulations while applying wire width variations ($\pm 5\%$), voltage variations ($\pm 7.5\%$), channel length variations ($\pm 5\%$) and temperature variations ($\pm 15\%$) around the nominal values. The variations are spatially correlated and generated using a five-level quad-tree where equal magnitudes of variations are specified for each level of the constructed quad-tree [9, 67]. In each simulation, skew and transient time constraints are evaluated. If the chip meets timing constraints, the chip is classified as good. Otherwise, the chip is classified as defective. For all structures, no yield loss is obtained from the transition time constraints. The proposed setup overcomes the well-known deficiencies of the ISPD 2010 contest, which have been discussed at length in [2, 9, 35]. Therefore, we have not reported extensive comparisons with the contest results.

The MRT design configurations are evaluated in Section 6.7.1. The performance of the MRT structures are evaluated in Section 6.7.2.

6.7.1 Evaluation of MRT design configurations

In this section, we evaluate the impact of the MRT design configurations. The selection of the number of top-level trees is evaluated in Section 6.7.1.1. The TRG guided the top-level tree construction is evaluated in Section 6.7.1.2. The use of topology reconfiguration is evaluated in Section 6.7.1.3. The delivery of the negative-edge of the clock signal is evaluated in Section 6.7.1.4.

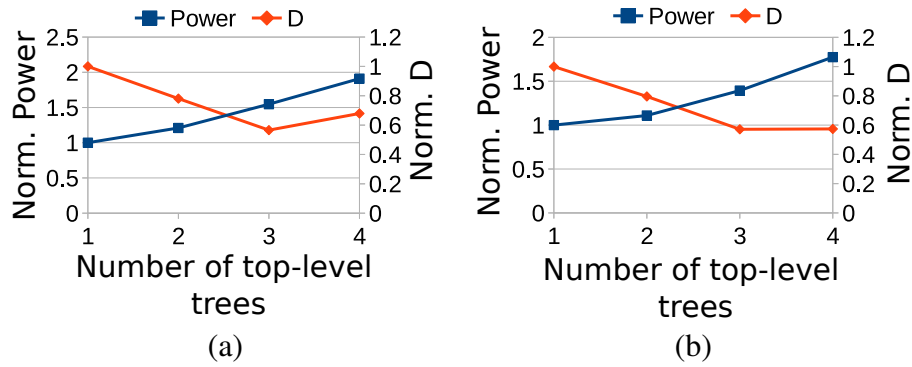


Figure 6.6: Evaluation of MRT structures with different number of top-level trees on the circuits (a) *dma* and (b) *aes*. The performance is evaluated in terms of power and robustness to variations. The clock network structure with 1 top-level tree is equivalent to the traditional clock tree.

6.7.1.1 Selection of the number of top-level trees

The selection of the number of top-level trees (N) is evaluated in Figure 6.6. The figure shows the performance of the MRT structure for different numbers of top-level trees on the circuits *dma* and *aes*. The performance is evaluated in terms of power and robustness to variations. The values presented in Figure 6.6 are normalized with respect to an MRT structure with $N = 1$, i.e., a clock network in the form of a clock tree. The robustness to variations is evaluated using D . The figure shows that D is improved when N is increased until a turning point from where the robustness is degraded. This is because each additional top-level tree improves the robustness to variations by

placing clock sinks that are distant in the initial tree topology close in at least one of the subsequent top-level tree topology. However, with each top-level tree that is constructed, the clock network becomes larger and more vulnerable to variations. Therefore, the robustness to variations is typically saturated when N is equal to three. Intuitively, the power consumption is increased with the number of top-level clock trees that are constructed, which can easily be observed in the figure.

Clearly, a fundamental problem for MRT structures becomes that of selecting N such that the minimum constraint on D is satisfied using the minimum amount of power consumption. This could potentially be solved using static timing analysis [35]. However, we consider this problem out of the scope of the chapter, as the chapter is focused on the concept and synthesis of MRT structures. Instead, we choose to evaluate the MRT structure with 2 top-level trees on all circuits because it provides the best trade-off between robustness and power consumption. Moreover, the MRT structures with more than 2 top-level trees introduce undesirable overheads in terms of both power consumption and runtime. Hence, the MRT structure with 2 top-level trees is selected to be the baseline for the proposed framework. In the remaining of this chapter, the MRT structure refers to the MRT structure that is obtained by constructing 2 top-level trees.

6.7.1.2 *Evaluation of the TRG guided top-level tree construction*

The TRG guided top-level tree construction is evaluated in Figure 6.7. In the figure, we compare constructing the top-level trees with the objective of minimizing wirelength and using the proposed metric in Eq (6.6). The comparison is performed in terms of average power consumption and average robustness to variations (D) over all the benchmarks in Table 6.3.

Compared with minimizing wirelength, the figure shows that the proposed metric improves D with 4% for the MRT structures. The robustness improvements stem from the proposed cost metric that encourages clock sinks (with skew constraints) that are distant in the initial tree topology to be

placed close in the subsequent tree topology. The robustness improvements come at the expense of a 2% overhead in power consumption. This is expected because subtrees that are distant in the initial tree topology are typically also distant spatially. Consequently, more wirelength is required to merge such subtrees, which intuitively increases power consumption.

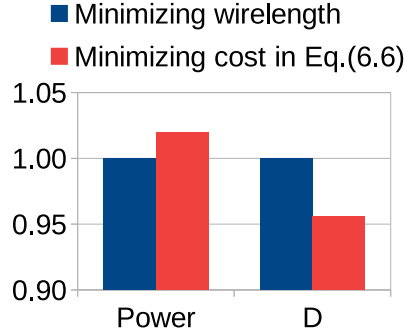


Figure 6.7: Evaluation of guiding the top-level tree construction using different cost metrics. The evaluation is performed in terms of normalized power and normalized robustness (D).

Clearly, the proposed cost metric introduces a trade-off between robustness and power consumption. We consider the small overheads in power consumption to be acceptable for the slight improvements in robustness. Consequently, we choose to guide the construction of the top-level trees using the proposed cost metric. Note that the trade-off between robustness (D) and power consumption is regulated by the parameter α . In our implementation, we set α to prioritize topology distance over wirelength, i.e., α is set close to zero ($\alpha = 0.01$).

6.7.1.3 Evaluation of topology reconfiguration

In this section, we evaluate the effectiveness of saving power using topology reconfiguration combined with DVFS. In Figure 6.8(a), we compare the proposed technique with only applying DVFS in terms of supply voltage and clock network switching capacitance. The figure shows that the

proposed technique results in 1% higher supply voltage and 12% lower clock network switching capacitance. This is because the topology reconfiguration introduces small nominal skews, which reduces the available timing margins by 2%. The smaller timing margins translate to requiring a 1% higher supply voltage to be used. Compared with only applying DVFS, it is not surprising that the proposed technique results in only marginally higher supply voltage, as large timing margins are required to slightly reduce V_{DD} , when V_{DD} is close to V_t , as discussed in Section 6.5.3.

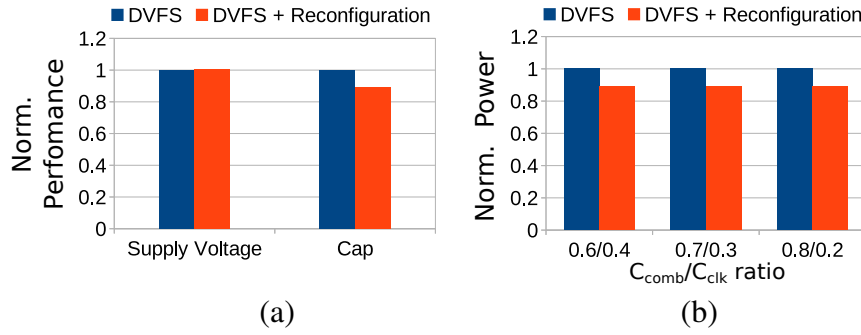


Figure 6.8: Evaluation of DVFS vs reconfiguration combined with DVFS in terms of average (a) supply voltage and switching capacitance and (b) total circuit power for different ratios of C_{comb} and C_{clk} . The experimental results shown in the figure are the average values for the benchmarks in Table 6.3.

In Figure 6.8(b), we compare the proposed technique with DVFS in terms of total circuit power. The total circuit power is computed using Eq (6.5) for different ratios of switching capacitance between the combinational logic (C_{comb}) and the clock network (C_{clk}). The figure shows that the proposed technique reduces the total circuit power by about 11%. The power saving is relatively independent of the ratio between the capacitance of the combinational logic and the clock network. The power saving is obtained due to the lower clock network switching capacitance. Consequently, we conclude that topology reconfiguration combined with DVFS is advantageous to only applying DVFS. The only drawback of the reconfiguration is that a mode control signal is required to be delivered to the clock gates. Nevertheless, the overhead is limited because there is only one clock gate in the MRT structures. Moreover, the wirelength used to deliver the mode control signal does

not contribute significantly to the dynamic power consumption. The results in the figure were obtained using α_{comb} and α_{clk} respectively set to 0.1 and 1. V_t for the technology is 0.4V [20].

6.7.1.4 Evaluation of the negative-edge of the clock signal

In this section, we evaluate the robustness in the delivery of the positive-edge and negative-edge of the clock signal. The comparison is performed in terms of D in Figure 6.9. The results shown in the figure are the average values obtained from the MRT structures for the benchmarks in Table 6.3.

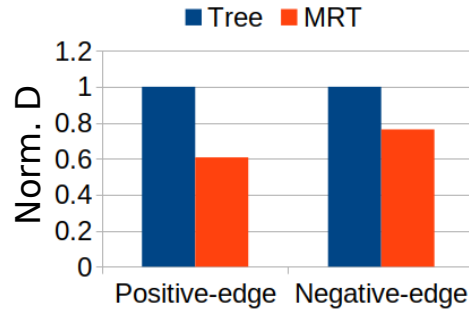


Figure 6.9: Evaluation of the MRT structure for the negative-edge and the positive-edge of the clock signal.

The figure shows that MRT structure delivers both the positive-edge and negative-edge of the clock signal with higher robustness to variations than a Tree structure. However, the robustness improvements are higher for the positive-edge of the clock signal than for the negative-edge of the clock signal. As it is shown in Figure 6.9, the MRT structure delivers the positive-edge of the clock signal with 39% lower D while the negative-edge is only delivered with 24% lower D . The difference in the robustness improvements stem from that the OR-gates act as average operators and max operators for the positive-edge and negative-edge of the clock signal, respectively. This was discussed and analyzed in Section 6.5.2. Consequently, the MRT structures are ideal for circuits with only positive-edge triggered sequential elements. However, they can also be used for

circuits with both positive-edge and negative-edge triggered sequential elements. Nevertheless, a slightly lower clock frequency will be required to be used.

Table 6.5: Evaluation of structures in terms of power consumption, supply voltage and timing quality.

| Benchmark (name) | Structure (name) | High Performance Mode | | Low Performance Mode | | Timing Constraints Satisfied? (Yes/No) | P_T in Eq (6.4) (mW) | Runtime (mins) |
|---------------------|---------------------|--------------------------|---------------|--------------------------|---------------|---|------------------------------|-------------------|
| | | Supply Voltage (V) | Power (mW) | Supply Voltage (V) | Power (mW) | | | |
| s1423 | Tree [1, 4, 23] | 1.02 | 35.97 | 0.61 | 2.25 | No | 29.23 | 2.1 |
| | Near-tree-SS [9] | 1.02 | 48.18 | 0.61 | 2.99 | No | 39.14 | 1.4 |
| | Near-tree-OR | 0.99 | 38.06 | 0.61 | 2.40 | Yes | 30.93 | 0.9 |
| | MRT | 0.99 | 36.27 | 0.61 | 2.06 | Yes | 29.43 | 0.9 |
| usbf | Tree [1, 4, 23] | 1.01 | 38.00 | 0.61 | 2.53 | Yes | 30.91 | 6.6 |
| | Near-tree-SS [9] | 0.98 | 55.66 | 0.61 | 3.65 | Yes | 45.25 | 5.4 |
| | Near-tree-OR | 0.98 | 48.06 | 0.62 | 3.24 | Yes | 39.10 | 3.6 |
| | MRT | 0.98 | 42.24 | 0.61 | 2.57 | Yes | 34.31 | 3.2 |
| dma | Tree [1, 4, 23] | 1.00 | 48.44 | 0.61 | 3.20 | Yes | 39.39 | 10.0 |
| | Near-tree-SS [9] | 0.98 | 71.02 | 0.61 | 4.80 | Yes | 57.78 | 7.3 |
| | Near-tree-OR | 0.99 | 70.68 | 0.64 | 4.76 | Yes | 57.49 | 7.5 |
| | MRT | 0.98 | 54.47 | 0.62 | 3.33 | Yes | 44.24 | 5.6 |
| pci_bridge32 | Tree [1, 4, 23] | 1.01 | 72.97 | 0.61 | 4.84 | Yes | 59.35 | 13.2 |
| | Near-tree-SS [9] | 0.98 | 98.53 | 0.61 | 6.65 | Yes | 80.15 | 16.8 |
| | Near-tree-OR | 0.99 | 120.16 | 0.63 | 7.92 | Yes | 97.71 | 18.3 |
| | MRT | 0.99 | 83.54 | 0.62 | 5.04 | Yes | 67.84 | 10.1 |
| ecg | Tree [1, 4, 23] | 1.02 | 175.10 | 0.61 | 11.35 | No | 142.35 | 62.0 |
| | Near-tree-SS [9] | 0.99 | 227.66 | 0.61 | 15.06 | Yes | 185.14 | 86.5 |
| | Near-tree-OR | 1.02 | 257.65 | 0.63 | 15.65 | No | 209.25 | 85.7 |
| | MRT | 1.02 | 220.18 | 0.62 | 12.07 | Yes | 178.56 | 49.1 |
| des | Tree [1, 4, 23] | 1.02 | 176.36 | 0.61 | 11.43 | No | 143.37 | 133.0 |
| | Near-tree-SS [9] | 1.01 | 228.75 | 0.61 | 14.32 | Yes | 185.86 | 89.4 |
| | Near-tree-OR | 0.99 | 196.65 | 0.62 | 12.91 | Yes | 159.90 | 53.8 |
| | MRT | 0.99 | 177.76 | 0.61 | 11.08 | Yes | 144.42 | 53.5 |
| aes | Tree [1, 4, 23] | 1.02 | 334.18 | 0.62 | 21.62 | No | 271.67 | 300.0 |
| | Near-tree-SS [9] | 1.01 | 466.89 | 0.61 | 29.72 | Yes | 379.45 | 293.2 |
| | Near-tree-OR | 1.02 | 527.13 | 0.64 | 31.43 | No | 427.99 | 283.5 |
| | MRT | 1.01 | 386.55 | 0.62 | 21.01 | Yes | 313.44 | 213.0 |
| Norm | Tree [1, 4, 23] | 1.00 | 1.00 | 1.00 | 1.00 | 3 | 1.00 | 1.00 |
| | Near-tree-SS [9] | 0.98 | 1.37 | 1.00 | 1.37 | 6 | 1.37 | 0.90 |
| | Near-tree-OR | 0.99 | 1.37 | 1.02 | 1.35 | 5 | 1.37 | 0.80 |
| | MRT | 0.98 | 1.12 | 1.00 | 1.00 | 7 | 1.11 | 0.60 |

6.7.2 Evaluation of MRT structures

In Table 6.5, we evaluate the performance of MRT structures in both high performance mode and low performance mode. The structures are evaluated in terms of supply voltage, power con-

sumption and the timing quality under variations. Recall that the MRT structures are obtained by performing the flow in Figure 6.3(a) using $N = 2$.

The supply voltage and power consumption of the structures are listed in the columns as ‘Supply Voltage’ and ‘Power’, respectively. The timing quality of the structures are evaluated based on the timing constraints in Eq (2.1) and Eq (2.2) in both modes of operation. The timing quality is reported as ‘Yes’ or ‘No’ in the column labeled ‘Timing Constraints Satisfied?’. The weighted clock power in Eq (6.4) is reported in the column labeled as ‘ P_T ’. The runtime of each structure is reported in minutes in the column labeled as ‘Runtime’. In the experimental setup, β and γ are set to be 0.8 and 0.2, respectively. The normalized results are shown in the bottom of the table and the best results in each column are shown in bold. The normalized results for the timing quality are reported as the total number of benchmarks that each structure meets the timing constraints.

Compared with the Tree structures, the Near-Tree-SS structures meet the timing constraints on 3 more benchmarks in both modes of operation. This is easy to understand because the Near-Tree-SS structures have alternative paths from clock source to clock sinks, which improves the robustness to variations. The Near-Tree-SS structures operate in 2% lower supply voltage in the high performance mode, when compared to the Tree structures. The reason is that the Near-Tree-SS structures have more timing margins available due to the alternative paths from clock source to clock sinks, which enables the supply voltage to be scaled down slightly more when compared to the Tree structures. However, the supply voltage for the Tree structures and the Near-Tree-SS structures are similar in the low performance mode because the timing constraints are significantly looser in the low performance mode due to the lower operating frequency. The power consumption of the Near-Tree-SS structures is 37% higher than the Tree structures in both modes of operation, which results in a 37% higher P_T . The explanation for this is that the Near-Tree-SS structures have alternative paths from the clock source to the clock sinks, which introduce significant capacitive overheads. The runtime of the Near-Tree-SS structures are 10% shorter. The explanation is that

clock tree optimization using SPICE simulations is more time-consuming for Tree structures than for Near-Tree-SS structures.

The Near-Tree-SS structures meet the timing constraints on 6 benchmarks while the Near-Tree-OR structures meet the timing constraints on 5 benchmarks. This stems mainly from that OR-gates are slightly more vulnerable to variations than clock buffers. Recall that the only difference between the two structures is that OR-gates are used to join alternative redundant paths instead of sink-splitting. The Near-Tree-SS structures operate in 1% and 2% lower supply voltage than the Near-Tree-OR structures in the high performance mode and the low performance mode, respectively. This is because the Near-Tree-SS structures have more timing margins available than that of the Near-Tree-OR structures, which enables the supply voltage to be scaled down slightly more. The power consumption of the Near-Tree-OR structure is similar to the Near-Tree-SS structures in the high performance mode. In the low performance mode, the Near-Tree-OR structures have 2% lower power consumption than the Near-Tree-SS structures. The lower power consumption in the low performance mode is a result of the Near-Tree-OR structures consume no short circuit power as there is only one gate driving each net. P_T is similar for both structures. The runtime of the Near-Tree-OR structures is 10% shorter than the Near-Tree-SS structures.

The MRT structures meet timing constraints on all benchmarks while the Near-Tree-OR structures fail to meet timing constraints on *aes* and *ecg*. The improvements in timing quality stem from that redundancy are inserted using top-level trees instead of using local-merges. The local-merges technique only considers if pairs of subtrees are sequentially related while the proposed TRG guided top-level tree construction attempts to place every pair of clock sinks with timing constraints close in at least one tree topology. The supply voltage for the MRT structures is respectively 1% and 2% lower, as compared to the supply voltage for the Near-Tree-OR structures in the high performance mode and the low performance mode. The MRT structures have 25% and 35% lower power consumption than the Near-Tree-OR structures in the high performance mode and the low perfor-

mance mode, respectively. Thus, the MRT structures have 26% lower P_T than the Near-Tree-OR structures. The improvement in P_T is a result of that the Near-Tree-OR structures have an excessive amount of redundant paths, which translates into larger clock networks with higher power consumption. Moreover, the MRT structures reconfigure the topology into a tree by turning-off the redundant paths, which results in significant power savings in the low performance mode. The MRT structures have similar power consumption with the Tree structures in the low performance mode, which validates the effectiveness of the reconfiguration of the topology. The runtime of the MRT structures is 20% shorter than the runtime of the Near-Tree-OR structures.

To illustrate the robustness improvements of the MRT structures over the Tree structures, we show a histogram of the skew introduced by variations for both the Tree structure and the MRT structure on the circuit *usbf* in Figure 6.10. As it can be observed, the MRT structure has a tighter skew distribution than the Tree structure, i.e., higher robustness to variations.

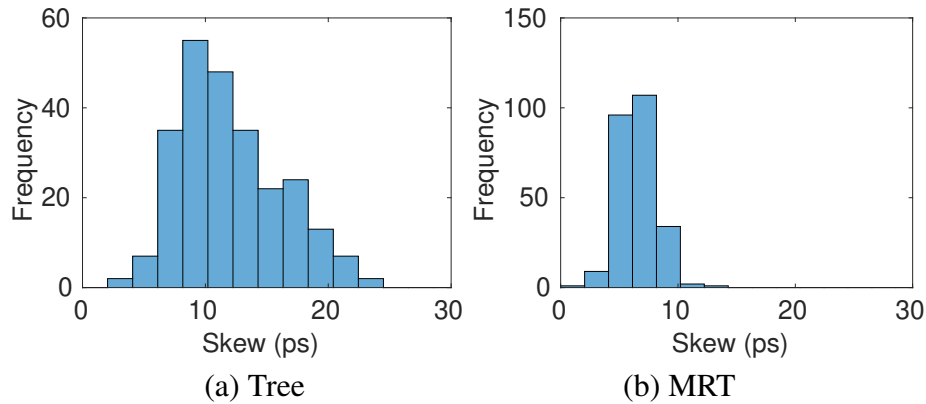


Figure 6.10: Histogram of skews from Monte Carlo simulations of (a) Tree and (b) MRT structures on *usbf*.

Intuitively, a clock tree is always the best clock network solution if it can satisfy the timing constraints without excessive guardbands. In this chapter, we focus on circuits where a higher robustness to variations is required. For such circuits, we have observed that the MRT structures

are capable of providing higher robustness to variations when compared to the state-of-the-art Near-Tree-SS structures. Moreover, the MRT structures can provide significantly lower power consumption in both modes of operation. Consequently, we conclude that the proposed MRT structures are advantageous to the Near-Tree structures [9], which have been demonstrated to outperform clock trees with cross-links. Recall that clock meshes have 3X-5X high power consumption than a clock tree. Therefore, the MRT structures are also advantageous to non-tree structures as the power overhead is as low as 12% in the high performance mode. Furthermore, the topology of the MRT structure can be reconfigured to save additional power in the low performance mode.

6.8 Summary and conclusion

In this chapter, we proposed the construction of clock networks with a mode reconfigurable topology for positive-edge triggered sequential elements. In high performance modes, the MRT structure is reconfigured into a near-tree to provide the required robustness to variations. In low performance modes, power is saved by reconfiguring the MRT structure into a tree topology using a single clock gate. To the best of the authors' knowledge, this is the first clock network that can be reconfigured from a near-tree into a tree topology. Compared to the state-of-the-art near-tree structures, the experimental results demonstrate improvements in power consumption and robustness to variations.

CHAPTER 7: FUTURE WORKS

In this chapter, the direction of the future research is outlined. The future research direction is mainly focused on the development of the OCV-Aware clock tree construction methodologies.

In Chapter 5, an OCV-aware CTS methodology is proposed. Based on our experimental evaluations, we speculate that there exists a great opportunity to further extend the proposed methodology. Specifically, we observed that the selection of the top-level topology have a huge impact on the final timing quality. In addition, the latency prediction for the bottom-level subtrees can be improved to more accurately account for the impacts of OCVs within a shorter amount of time using machine learning models.

Top-level topology exploration: In our preliminary study, the top-level of the clock trees are constructed by enumerating and pruning various candidate non-uniform H-trees, which provided promising improvements in terms of timing quality and runtime. We believe that the cost metric that is used to prune the candidate topologies can be further improved by integrating the timing slacks. In addition, modifying the top-level topology based on the feedback from the clock tree that is obtained in the previous iteration may provide a great opportunity to improve both the timing quality and the cost of the clock trees. Moreover, various clock tree structures and techniques can be explored to construct the top-level topology.

Latency predictions based on machine learning models: In the proposed methodology, the OCV predictions at each iteration are based on the actual latencies of the clock tree that is obtained in the previous iteration. Therefore, each bottom-level subtree is required to be constructed multiple times until the flow converges to a solution, which introduces an overhead in terms of runtime. To overcome this problem, various machine learning models can be used to accelerate the convergence of the proposed framework while accurately predicting the latency (OCVs) [61–63].

LIST OF REFERENCES

- [1] K.D. Boese and A.B. Kahng. 1992. Zero-skew clock routing trees with minimum wirelength. In *Proceedings of the IEEE International ASIC Conference and Exhibit*. 17–21.
- [2] Shashank Bujimalla and Cheng-Kok Koh. 2011. Synthesis of Low Power Clock Trees for Handling Power-Supply Variations. In *Proceedings of the International Symposium on Physical Design (ISPD '11)*. 37–44.
- [3] T.-H. Chao, Y.-C. Hsu, and J.-M. Ho. 1992. Zero skew clock net routing. In *Proceedings of the Design Automation Conference*. 518–523.
- [4] Y. P. Chen and D. F. Wong. 1996. An Algorithm for Zero-Skew Clock Tree Routing with Buffer Insertion. In *Proceedings of the European Conference on Design and Test (EDTC '96)*. 230.
- [5] M. Edahiro. 1993. A Clustering-Based Optimization Algorithm in Zero-Skew Routings. In *Proceedings of the Design Automation Conference (DAC)*. 612–616.
- [6] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao. Bounded-skew clock and steiner routing. *ACM Transactions on Design Automation of Electronic Systems*, 3(3):341–388, July 1998.
- [7] Rickard Ewetz, Shankarshana Janarthanan, Cheng-Kok Koh, and Chuan Yean Tan. 2015. *Benchmark circuits for clock scheduling and synthesis*. ([Available Online] <https://purr.purdue.edu/publications/1759>).
- [8] Rickard Ewetz, Shankarshana Janarthanan, and Cheng-Kok Koh. 2015. Construction of re-configurable clock trees for MCMM designs. In *Proceedings of the Design Automation Conference (DAC)*. 1–6.

- [9] Rickard Ewetz and Cheng-Kok Koh. 2015. Cost-Effective Robustness in Clock Networks Using Near-Tree Structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 4 (2015), 515–528.
- [10] Chau-Chin Huang et al. 2016. Timing-driven Cell Placement Optimization for Early Slack Histogram Compression In *Proceedings of the Design Automation Conference, 2016. (DAC'16)*. 81:1–81:6.
- [11] Myung-Chul Kim et al. 2015. ICCAD-2015 CAD Contest in Incremental Timing-driven Placement and Benchmark Suite In *Proceedings of the International Conference On Computer Aided Design, 2015. (ICCAD'15)*. 921–926.
- [12] Jianchao Lu and Baris Taskin. 2009. Post-CTS clock skew scheduling with limited delay buffering. In *IEEE International Midwest Symposium on Circuits and Systems*. 224–227.
- [13] Tarun Mittal and Cheng-Kok Koh. 2011. Cross Link Insertion for Improving Tolerance to Variations in Clock Network Synthesis. In *Proceedings of the International Symposium on Physical Design (ISPD '11)*. 29–36.
- [14] NGSPICE. 2012. [Available Online] <http://ngspice.sourceforge.net/>. (2012).
- [15] A. Rajaram, Jiang Hu, and R. Mahapatra. 2004. Reducing clock skew variability via cross links. In *Proceedings of the Design Automation Conference, 2004. (DAC)*. 18–23.
- [16] Anand Rajaram, David Z. Pan, and Jiang Hu. 2005. Improved Algorithms for Link-Based Non-Tree Clock Networks for Skew Variability Reduction. In *Proceedings of the International Symposium on Physical Design (ISPD '05)*. 55–62.
- [17] Subhendu Roy, Pavlos M. Mattheakis, Laurent Masse-Navette, and David Z. Pan. 2014. Clock Tree Resynthesis for Multi-Corner Multi-Mode Timing Closure. In *Proceedings of the International Symposium on Physical Design (ISPD '14)*. 69–76.

- [18] Hyungjung Seo, Juyeon Kim, Minseok Kang, and Taewhan Kim. 2015. Synthesis for power-aware clock spines. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'15)*. 126–131.
- [19] Xin-Wei Shih, Hsu-Chieh Lee, Kuan-Hsien Ho, and Yao-Wen Chang. 2010. High variation-tolerant obstacle-avoiding clock mesh synthesis with symmetrical driving trees. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'10)*. 452–457.
- [20] C. N. Sze. 2010. ISPD 2010 High Performance Clock Network Synthesis Contest: Benchmark Suite and Results. In *Proceedings of the International Symposium on Physical Design (ISPD '10)*. 143.
- [21] C. N. Sze, Phillip Restle, Gi-Joon Nam, and Charles Alpert. 2009. Ispd2009 Clock Network Synthesis Contest. In *Proceedings of the International Symposium on Physical Design (ISPD '09)*. 149–150.
- [22] Chung-wen Albert Tsao and Cheng-kok Koh. 2002. UST/DME: A Clock Tree Router for General Skew Constraints. *ACM Transactions on Design Automation of Electronic Systems* 7, 3 (2002), 359–379.
- [23] R.-S. Tsay. 1991. Exact zero skew. In *Proceedings of the International Conference on Computer-Aided Design Digest of Technical Papers*. 336–339.
- [24] G. Venkataraman, N. Jayakumar, J. Hu, P. Li, Sunil Khatri, Anand Rajaram, P. McGuinness, and C. Alpert. 2005. Practical techniques to reduce skew and its variations in buffered clock networks. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. 592–596.
- [25] Ganesh Venkataraman, Zhuo Feng, Jiang Hu, and Peng Li. 2006. Combinatorial Algorithms

- for Fast Clock Mesh Optimization. In *Proceedings of the International Conference on Computer Aided Design. (ICCAD)* 563–567.
- [26] Laung-Terng Wang, Yao-Wen Chang, and Kwang-Ting (Tim) Cheng. 2009. *Electronic Design Automation: Synthesis, Verification, and Test*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [27] Linfu Xiao, Zigang Xiao, Zaichen Qian, Yan Jiang, Tao Huang, Haitong Tian, and Evangeline F. Y. Young. 2010. Local clock skew minimization using blockage-aware mixed tree-mesh clock network. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. 458–462.
- [28] Christoph Albrecht et al. 2002. Maximum Mean Weight Cycle in a Digraph and Minimizing Cycle Time of a Logic Chip. *Discrete Applied Mathematics* 123, 1-3 (2002), 103–127.
- [29] Krit Athikulwongse, Xin Zhao, and Sung Kyu Lim. 2010. Buffered Clock Tree Sizing for Skew Minimization Under Power and Thermal Budgets. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC’10)*. 474–479.
- [30] Rickard Ewetz. 2017. A Clock Tree Optimization Framework with Predictable Timing Quality In *Proceedings of the Design Automation Conference (DAC’17)*. 13–18.
- [31] Rickard Ewetz and Cheng-Kok Koh. 2018. Scalable Construction of Clock Trees with Useful Skew and High Timing Quality. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018).
- [32] J.P. Fishburn. 1990. Clock skew optimization. *IEEE Transaction on Computers* 39, 7 (1990), 945–951.
- [33] Matthew R. Guthaus, Dennis Sylvester, and Richard B. Brown. 2006. Clock Buffer and Wire

- Sizing Using Sequential Programming In *Proceedings of the Design Automation Conference (DAC'06)*. 1041–1046.
- [34] S. Held, B. Korte, J. Massberg, M. Ringe, and J. Vygen. 2003. Clock scheduling and clock-tree construction for high performance ASICs. In *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*. 232–239.
 - [35] Dong-Jin Lee and Igor L. Markov. 2011. Multilevel tree fusion for robust clock networks. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. 632–639.
 - [36] J. Lillis, Chung-Kuan Cheng, and T. T. Y. Lin. 1996. Optimal wire sizing and buffer insertion for low power and a generalized delay model. *IEEE Journal of Solid-State Circuits* 31, 3 (1996).
 - [37] Logan Rakai et al. 2013. Buffer Sizing for Clock Networks Using Robust Geometric Programming Considering Variations in Buffer Sizes. In *Proceedings of the International Symposium on Physical Design, (ISPD'13)*. 154–161.
 - [38] Venky Ramachandran. 2012. Construction of Minimal Functional Skew Clock Trees In *Proceedings of the International Symposium on Physical Design, (ISPD'12)*. 119–120.
 - [39] Subhendu Roy, Pavlos M. Mattheakis, Laurent Masse-Navette, and David Z. Pan. 2015. Clock Tree Resynthesis for Multi-Corner Multi-Mode Timing Closure. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 4 (2015), 589–602.
 - [40] King Ho Tam et al. 2008. Dual- V_{dd} Buffer Insertion for Power Reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27 (2008), 1498–1502.
 - [41] Jeng-Liang Tsai, Tsung-Hao Chen, and C. C. P. Chen. 2004. Zero skew clock-tree optimization with buffer insertion/sizing and wire sizing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23, 4 (2004), 565–572.

- [42] Necati Uysal and Rickard Ewetz. 2018. OCV Guided Clock Tree Topology Reconstruction. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'18)*, 1–6.
- [43] L. P. P. van Ginneken. 1990. Buffer placement in distributed RC-tree network for minimal Elmore delay. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 865–868.
- [44] Du Zhong, Wen Zhiping and Yu lixin. A delay metric for VLSI interconnect. In *Proceedings of the International Conference on ASIC*, 2005, pp. 1042-1046.
- [45] Tao Lin, E. Acar and L. Pileggi. h-gamma: an RC delay metric based on a gamma distribution approximation of the homogeneous response. In *Proceedings of the International Conference on Computer-Aided Design. Digest of Technical Papers (IEEE Cat. No.98CB36287)*, 1998, pp. 19-25.
- [46] Alpert, Charles J. and Devgan, Anirudh and Kashyap, Chandramouli. A Two Moment RC Delay Metric for Performance Optimization. In *Proceedings of the International Symposium on Physical Design*, 2000, pp. 69-74.
- [47] B. Tutuianu, F. Dartu and L. Pileggi. An explicit RC-circuit delay approximation based on the first three moments of the impulse response. In *Proceedings of the Design Automation Conference*, 1996, pp. 611-616.
- [48] Kai Wang and Malgorzata Marek-Sadowska. 2004. Buffer Sizing for Clock Power Minimization Subject to General Skew Constraints In *Proceedings of the Design Automation Conference. (DAC'04)*. 159–164.
- [49] V. Stojanovic, D. Markovic, B. Nikolic, M. A. Horowitz, and R. W. Brodersen. 2002. Energy-

- delay tradeoffs in combinational logic using gate sizing and supply voltage optimization. In *Proceedings of the European Solid-State Circuits Conference*. 211–214.
- [50] W-C. Elmore. 1948. The Transient Analysis of Damped Linear Networks with Particular Regard to Wideband Amplifiers. In *Applied Physics*. vol. 196, 1.
- [51] Dennis J.-H. Huang and Andrew B. Kahng and Chung-Wen Albert Tsao. On the Bounded-Skew Clock and Steiner Routing Problems. In *Proceedings of the Design Automation Conference, 1995. (DAC'95)*, 508–513, 1995.
- [52] Cong, Jason and Kahng, Andrew B. and Koh, Cheng-Kok and Tsao, C.-W. Albert. Bounded-skew clock and Steiner routing under Elmore delay. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, 66–71, 1995.
- [53] Ewetz, Rickard and Koh, Cheng-Kok. A Useful Skew Tree Framework for Inserting Large Safety Margins. In *Proceedings of the International Symposium on Physical Design (ISPD'15)*, 85–92, 2015.
- [54] H. Bakoglu. Circuits, interconnects, and packaging for VLSI. *Reading, MA: Addison-Wesley*, 1990.
- [55] T.-B. Chan, K. Han, A. B. Kahng, J.-G. Lee, and S. Nath. OCV-aware top-level clock tree optimization. In *Proceedings of the Great Lakes Symposium on VLSI. (GLSVLSI'14)*, pages 33–38, 2014.
- [56] M. Edahiro. Minimum skew and minimum path length routing in VLSI layout design. In *NEC Research and Development*, pages 569–575, 1991.
- [57] R. Ewetz and C.-K. Koh. Clock tree construction based on arrival time constraints. In *Proceedings of the International Symposium on Physical Design (ISPD'17)*, pages 67–74, 2017.

- [58] W.-C. D. Lam and C.-K. Koh. Process variation robust clock tree routing. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'05)*, pages 606–611, 2005.
- [59] N. Uysal, W.-H. Liu, and R. Ewetz. Latency constraint guided buffer sizing and layer assignment for clock trees with useful skew. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'19)*, pages 761–766, 2019. <https://doi.org/10.1145/3287624.3287681>
- [60] L. W. Nagel. SPICE2: A computer program to simulate semiconductor circuits. *Technical Report ERL-M520*, University of California, Berkeley, CA, May 1975.
- [61] A. B. Kahng, U. Mallappa, L. Saul, and S. Tong. Unobserved Corner' Prediction: Reducing Timing Analysis Effort for Faster Design Convergence in Advanced-Node Design. In *Proceedings of the 2019 Design, Automation Test in Europe Conference Exhibition (DATE'19)*, pp. 168–173, 2019.
- [62] A. B. Kahng, U. Mallappa, and L. Saul. Using Machine Learning to Predict Path-Based Slack from Graph-Based Timing Analysis. In *Proceedings of the 2018 IEEE 36th International Conference on Computer Design (ICCD)*, pp. 603–612, 2018.
- [63] S. Bian, M. Hiromoto, M. Shintani, and T. Sato. LSTA: Learning-based static timing analysis for high-dimensional correlated on-chip variations. In *Proceedings of the Design Automation Conference (DAC)*, pp. 1–6, 2018.
- [64] D. Mangiras, P. Mattheakis, P.-O. Ribet, and G. Dimitrakopoulos. Soft-Clustering Driven Flip-flop Placement Targeting Clock-induced OCV. In *Proceedings of the 2020 International Symposium on Physical Design (ISPD'20)*, pp. 25–32, 2020.
- [65] A. Farshidi, L. Rakai, L. Behjat, and D. Westwick. Variation-aware clock network buffer

- sizing using robust multi-objective optimization. In *Optim Eng* , vol. 17, no. 2, pp. 473–500, 2016.
- [66] IBM. ILOG CPLEX Optimization studio. 2016.
- [67] A. Agarwal, D. Blaauw, and V. Zolotov. 2003. Statistical timing analysis for intra-die process variations with spatial correlations. In *ICCAD-2003. Proceedings of the International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*. 900–907.
- [68] D. Lee, M. Kim, and I. L. Markov. 2006. Low-power clock trees for CPUs. In *Proceedings of the International Conference on Computer Aided Design (ICCAD '10)*. pp. 444–451.
- [69] A. Rajaram and D. Z. Pan. 2006. Variation tolerant buffered clock network synthesis with cross links. In *Proceedings of the International Symposium on Physical Design (ISPD '06)*. pp. 157–164.
- [70] R. Ewetz, C. Y. Tan, and C.-K. Koh. 2016. Construction of Latency-Bounded Clock Trees. In *Proceedings of the International Symposium on Physical Design (ISPD '16)*. pp. 81–88.
- [71] Necati Uysal, Juan Ariel Cabrera, and Rickard Ewetz. 2020. Synthesis of Clock Networks with a Mode Reconfigurable Topology and No Short Circuit Current. In *Proceedings of the International Symposium on Physical Design (ISPD '20)*. 103–110.
- [72] Necati Uysal and Rickard Ewetz. 2021. An OCV-Aware Clock Tree Synthesis Methodology. In *Proceedings of the International Conference on Computer Aided Design [accepted] (ICCAD '21)* .