Fakultät für Elektrotechnik und Informationstechnik
Technische Universität München

Design-Dependent Monitors to Track the Timing of Digital CMOS Circuits

Jahnavi Kasturi Rangan

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr.-Ing. Ralf Brederlow

Prüfende der Dissertation:
1. apl. Prof. Dr.-Ing. habil. Helmut Gräb
2. Prof. Dr.-Ing. Steffen Paul

Die Dissertation wurde am 28.05.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 23.09.2019 angenommen.

Abstract

Decrease in technology nodes has resulted in increased process and environmental variations (PVT) of semiconductor devices. These increased PVT variations in turn have a higher impact on digital circuit timing. To avoid timing errors in worst case scenarios, substantial guardbands are added to the operating frequency and supply voltage of circuits. These guardbands make sure that the required timing performance is met, however, this results in increased power consumption, area, cost and design time. Additionally, verifying if the timing criteria are met after manufacturing of chips is of utmost importance. For this reason, multiple test-structure circuits are required to be implemented on different parts of a chip such that the timing behavior of these test-structures are highly correlated to the behavior of a chip. The measurement data from these monitors should be able to predict the timing of the complete chip with high accuracy.

This thesis presents synthesis of design-dependent ring oscillators which aim to track the delays of setup-time critical paths of a chip. In the design-dependent ring oscillator (DDRO) approach, at first a set of setup-time critical paths are identified and grouped based on their similarities of timing behavior across PVT space. Subsequently, a DDRO is constructed for each group of critical paths. Frequency measurements from DDROs then directly represent the timing of their respective critical path groups.

In this work, in addition to the state-of-the-art method various DDRO synthesis formulations are developed and their accuracy to track critical path timing is evaluated. Moreover, the computational complexity of DDRO synthesis method is significantly reduced by using static timing analysis (STA) instead of analog simulation to characterize delays. Additionally, usage of developed heuristic optimization algorithms instead of direct solvers further reduces the computational complexity to synthesize DDROs. Furthermore, suitable clustering algorithms are selected such that a single DDRO can be constructed for a group of critical paths. The DDRO synthesis and clustering methods are evaluated on a sub-40nm technology. Based on the evaluation of DDROs, the synthesis methods proposed in this thesis tracks the timing of a digital design much better than other state-of-the-art timing monitors. Amongst the monitors investigated, DDROs are found to be the best suited timing monitors for large-scale industrial designs, in particular due to their reduced synthesis complexity.

Acknowledgment

This work was supported by the German Federal Ministry of Education and Research (BMBF) under the project number 16ES0303 within the funding project RESIST-RESilient transport InfraSTructure to extreme events. This work was concluded during my work as a PhD candidate at Infineon Technologies A.G., Neubiberg, Germany.

First and foremost, I would like to thank my professor Apl. Prof. Dr.-Ing. habil. Helmut Graeb for generously being my adviser for my PhD thesis. He constantly encouraged and guided me through all the stages of my thesis. His advice on my research work and career has been invaluable. His deep knowledge and experience in the field of circuit optimization and electronic design automation has been greatly favorable for my research work.

I am extremely thankful to Infineon Technologies A.G. for giving me an opportunity to pursue an industrial PhD in collaboration with the Technical University of Munich, Germany. I am especially thankful to all my fantastic team members for not only sharing their extensive technical knowledge but also for their guidance and moral support during the past three years. Not only have I had technical knowledge growth, but I have also grown as an individual under their guidance. Thank you also for the delightful and interesting discussions.

I am thankful to Dr.-Ing. Nasim Pour Aryan who was my supervisor at Infineon Technologies and a colleague over the past three years and now a very good friend. I would like to thank her for all the discussions we had technical and otherwise. I am also grateful to her for reviewing my dissertation and for all the words of encouragement. Many Thanks to Mr. Jens Bargfrede who closely guided me throughout the research work. His words of wisdom helped me during the tough days of my research. Moreover, his deep technical knowledge in integrated circuits, mathematics and programming was a guiding light for my research. Furthermore, I am extremely thankful to Mr. Christian Funke whose tremendous knowledge in the field of timing of digital circuits was a constant source of novel ideas. He constantly challenged my knowledge on the subject and kept me on my toes which helped me understand the global picture and the practical usage of my work. Moreover, his advice on presenting my research to various audiences has helped me develop and improve my presentation skills.

Special thanks to Dr. Rainer Kress, Dr.-Ing Klaus Oettinger, Mr. Steffen Rost, Mr. Werner May who gave me the wonderful opportunity to be part of Infineon family and pursue my PhD. Many thanks to Dr.-Ing Petra Nordholz and Mr. Thomas Moehring who are my other team members at Infineon who made the work environment spirited, interesting and enjoyable. Furthermore, I would like to thank Lantao Wang who did his master thesis and internship under my supervision at Infineon. His work supported my research which helped the completion of the my research.

This PhD thesis would have never been a success without the support of my parents- Mr. Kasturi Rangan, Ms. Kamakshi and my extended family. Throughout my life, they have been a constant source of encouragement and extend their support to pursue my dreams and achieve my goals. Their moral support is invaluable and I would like to express my deepest gratitude for being such a wonderful family.

I would like to thank my friend Faisal Caeiro who inspired me to pursue my dreams and supported me through all the stages of my PhD thesis.

Finally, last but not the least my special thanks to all my friends who are a part of my life and believed that I could achieve any goal that I set for myself.

Contents

1 Introduction

1.1 Motivation

Electronic systems are used extensively in safety critical applications involving public health, property and environment. Hence it is essential to ensure the correct working and reliability of the electronic systems in use. Some examples of these safety critical systems could be anti-lock braking systems in automobiles, motor and gear controllers, airbag controllers, fly-by-wire aircraft, shut-down systems at nuclear power plants [1], and electronics in the field of medicine and recreation. Due to the nature of these applications, safety critical electronic systems are required to operate across a wide range of environmental conditions and workload. Therefore, to ensure safe operation, reliability and robustness of safety critical systems are of prime importance.

Compared to former larger technology nodes, the performance of electronic circuits such as timing are further influenced by variations in manufacturing, environment and aging [2] [3] [4] [5] [6]. Variations in timing of digital circuits due to manufacturing and environmental variations might lead to timing errors which in turn could lead to failure of electronic systems. To prevent such failures, analysis and implementation of a chip is performed with worst case scenarios. Thus, additional design guardbands are added based on worst case timing analysis and is verified by a sign-off method. This pessimistic approach ensures the requested circuit performance which is the operating frequency, but at the cost of larger area, higher power consumption and increased design time.

Furthermore, manufactured chips could be subjected to more variations with deteriorated timing attributes due to process or aging which are not modeled accurately and resulting in mismatches between hardware and the model. Thus, the timing of certain chips might not meet the required timing criteria. Therefore, speed binning [7] [8] of all the produced chips is performed where, as the name suggests, the chips are categorized based on their frequency. Additionally, post-silicon chip validation [9] [10] is done to verify if the required timing criteria are met. Thus, based on these steps, the yield of manufactured chips, i.e. the percentage of chips amongst the complete set which meet the required timing criteria, is obtained.

During the post-silicon chip validation, additional guardbands known as test guardbands are added to check if chips meet the timing criteria due to test inaccuracies and because the chips could be subjected to additional variations and aging. Moreover, after manufacturing, no two chips are identical and also, multiple tests on the same chip show different results. An example of addition of test guardbands is done by overcompensating the frequency of the given chip. This means that if a chip is set to work at frequency

f_C Hz, then the chip is verified for $f_C + \Delta f_C$ Hz. The added guardband Δf_C depends on the accuracy of timing prediction of the entire chip during post-silicon chip validation.

For accurate prediction of chip timing, performing an exhaustive test, i.e. to test each path of a design to ensure zero defects is not feasible either with scan test methods [11] or with functional test patterns [12]. Moreover, increase in test time results in increased test costs and thus the test time should be minimized. Therefore, this thesis proposes to use timing monitors as test structures that can be placed on multiple parts of a chip and are used to evaluate the delay of a chip at the given fabrication condition and at different voltage and temperature conditions. Formerly, generic ring oscillator which are single cell type test structures were used to evaluate the chip timing behavior. However, these monitors are not adequate since they are not design-specific and therefore do not reflect the timing behavior of a chip.

Figures 1.1 and 1.2 show examples of timing monitors which have poor and good correlation to the manufactured chips, respectively. Figures show f_{TM}, the minimum frequency of the timing monitor which corresponds to the chip frequency f_C, that meets the required timing criteria. Since the measurement frequency of timing monitors cannot have 100% correlation with the chip frequency, classifying only chips below f_{TM} could result in chips which are classified as a good chip but might not meet the required timing criteria.

Therefore, a certain test guardband by overcompensating frequency Δf is added which is based on the quality of timing monitors. The timing monitor in Figure 1.1 does not accurately reflect the timing behavior of the chip and therefore, has larger test guardband. On the other hand, the timing monitor in Figure 1.2 is highly correlated to the chip frequency and thus has reduced test guardband. Therefore, the yield of the produced chips could be higher by using a good timing monitor which has a sharper screening in comparison to a poorly correlated timing monitor.

Thus, a set of timing monitors is required to be designed and implemented which can precisely represent the timing behavior of chip design across process, temperature and voltage conditions. Furthermore, it should be ensured that these timing monitors collectively represent the timing behavior of all the blocks on the chip such that they can fully reflect the timing behavior of the chip. Additionally, correlating timing monitors whose timing behavior represent the timing behavior of an industrial design could also aid in the improvement of timing models. Thus, the design of design-dependent timing monitors are also considered as an integral part towards the improvement of timing models.

In summary, design-dependent timing monitors which reflect the delay behavior of a digital design help in the substantial reduction of test guardbands and increase the yield of produced chips. This thesis dives into this design specific approach and endeavors to provide different design methods for the design-dependent timing monitors which are highly correlated to a given digital design. Moreover, different methods are discussed which can reduce the required number of timing monitors making them feasible to evaluate of timing of large scale industrial designs.

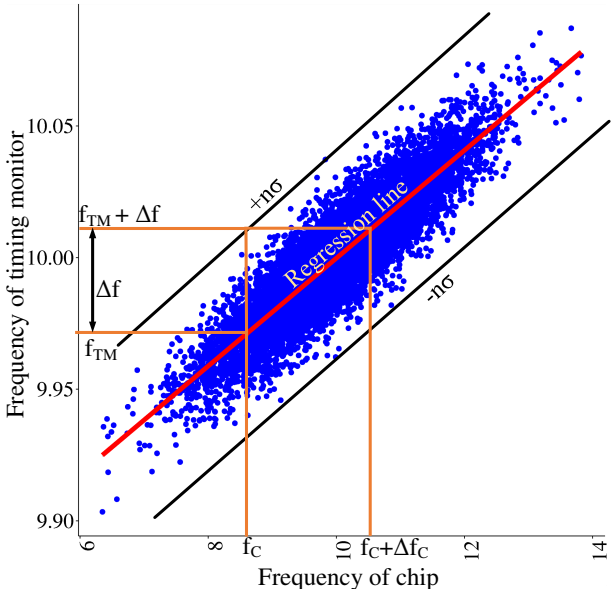


Figure 1.1: Example of a timing monitor with poor correlation to manufactured chips thus resulting in increased test guardbands and resulting in low chip manufacturing yield

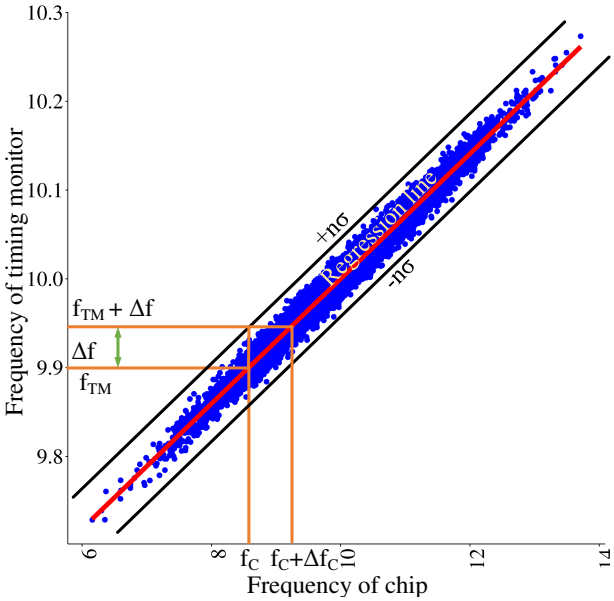


Figure 1.2: Example of a timing monitor with good correlation to manufactured chips thus resulting in reduced test guardbands and resulting in high chip manufacturing yield

1.2 State of the Art

The state-of-the-art performance monitors can be broadly classified into two categories, namely, generic monitors and design-dependent monitors. The aim of this thesis is to build monitors which can mimic the delay behavior of a given design. Therefore, the existing state-of-the-art monitors are analyzed with respect to the quality of representing the timing behavior of a given digital design.

Traditional timing monitors are generic ring oscillator (GRO) monitors and process-specific ring oscillators (PSROs) [13] [14] [3] [15]. These monitors are constructed with a single cell type. A ring oscillator is built by connecting the output of a chain of inverting CMOS standard cells or circuit blocks to its input, in form of a closed loop. The frequency of oscillation of a ring oscillator is measured to obtain the signal propagation delay through the chain of inverting CMOS circuit which are built to analyze the technology node at an early stage.

GROs are simple structures with low design and area cost. PSROs are also made of single cell type ROs, but are built in order to extract the sensitivity of delay of circuits w.r.t. specific process parameters. This means, not only frequency but current and voltage drop of these test structures are measured. However, both GROs and PSROs are neither based on the analysis of all standard cell types nor on the timing behavior of critical paths. This makes correlating the design-specific and path-specific behavior to these ring oscillators very complex [9] [16] [17] and thus PSROs and GROs are not suitable candidates to represent the timing of a chip design.

To represent the delay behavior of a chip, design-dependent monitors were developed. One of the design-dependent monitors is the in-situ approach. The concept of in-situ monitors is to evaluate the timing of a chip by monitoring its setup-time critical paths. Hence, an additional flip-flop, so called *shadow* flip-flop, with a delayed clock is placed alongside the capture flip-flop of a critical path [18] [18] [19]. Due to the addition of shadow flip-flops in critical paths of the design, in-situ monitors are intrusive to the design. Moreover, the additional flip-flops added to thousands of critical paths result in a tremendous increase in area and power overhead.

The drawbacks of GROs and in-situ monitors are overcome by representative critical path (RCP) [20] monitors. Hence, a single monitor is used to represent all the setup critical paths in a design. Although RCPs are design-specific, having only one monitor to represent all the critical paths makes the RCP inaccurate. The drawback of RCPs is overcome by tunable replica circuits (TRCs) [21], which are built adjacent to the critical paths and tuned to track their delays. However, TRCs are intrusive and highly complex to design.

The drawbacks of the mentioned approaches are overcome by employing design dependent ring oscillator (DDRO) monitors [22] [23] [24]. In the DDRO approach, setup critical paths are grouped based on their similarities w.r.t. delay sensitivities to PVT. For each path group, a DDRO consisting of a chain of tiles is constructed, where a tile is a chain of standard cells of equal type. The objective of the DDRO is to match the timing behavior of the target critical path group. The state-of-the-art DDROs however, do not consider the impact of input transition time and output load of a tile during the

DDRO synthesis. Moreover, due to the concept of sensitivity matching, these DDROs are forced to use SPICE-based delay characterization which makes them unsuitable for large-scale industrial designs. Additionally, process parameters affecting delay sensitivities differ for each technology node. Thus, majorly contributing process parameters impacting the delay sensitivities have to be evaluated for every technology node. Furthermore, given that it is highly challenging to compute delay sensitivities with high accuracy, sensitivity-based DDROs might result in tracking delay with poor quality to their respective critical paths.

In summary, there is a lack of highly accurate monitoring systems whose timing behavior match the timing behavior of a chip across various conditions of PVT. Thereby, evaluating the timing of a chip such that the voltage and timing overestimation of a chip can be reduced effectively by the reduction of guardbands.

1.3 Contributions of this Work

This thesis focuses on the methods to synthesize DDROs to represent the timing of critical paths in a digital design. Thereby, a new method of DDRO synthesis is proposed which uses delay-tracking for construction of DDROs in contrast to the sensitivity-tracking used in the existing state-of-the-art method [22]. Furthermore, high accuracy settings for the characterization of building blocks of DDROs are proposed which involves delay characterization with realistic environment. Moreover, quadratic programming with higher accuracy for the DDRO synthesis is proposed instead of the linear programming used in the existing state-of-the-art method [22]. With the high accuracy settings for delay characterization and DDRO objective, direct solvers for the DDRO synthesis result in infeasible CPU run-times. Therefore, three new heuristic algorithms are developed which have reduced and feasible CPU run-times.

To further reduce the computational complexity, delay characterization for DDRO synthesis is obtained by static timing analysis (STA) instead of SPICE simulation, which is used in the state-of-the-art method. Also, to facilitate DDRO synthesis for large scale industrial designs, critical paths are identified by STA and three different clustering methods for the grouping of critical paths are investigated; namely, KMeans++, Hierarchical clustering and model based clustering. For each group of critical paths one DDRO is synthesized. Additionally, using STA makes it feasible to obtain the delays of thousands of critical paths on a chip and thus makes the DDRO synthesis feasible for large scale designs. DDROs are synthesized for groups of critical paths and evaluated for their timing tracking ability of their respective group. To assess the quality of the developed DDRO synthesis method, DDROs are built and evaluated for an industrial technology below 40nm. The proposed DDRO synthesis methods are evaluated by computing the delay tracking error with respect to the given target critical path. Parts of this work have been published in [25] and submitted to TCAS-1 journal to be published in January 2020 [26].

1.4 Structure of this Dissertation

The remainder of this dissertation is structured as follows. In Chapter 2, the literature on the causes of process, voltage and temperature variations, along with their impact on the timing of digital circuits is reviewed. Furthermore, the existing state-of-the-art timing monitors used for evaluating the timing of technology nodes and chip design are explained. In Chapter 3, the design goal of DDROs to evaluate timing of critical paths on a chip is explained. Moreover, mathematical formulations of the sensitivity-based DDROs are introduced. Additionally, enhanced methods of characterizing the delay of building blocks of DDROs are presented. Furthermore, the concept and formulation of delay-tracking-based DDRO synthesis is presented. This Chapter also describes three new heuristic methods developed to solve the DDRO optimization problem. Chapter 4 presents results of DDRO synthesis for various scenarios: (1) various DDRO formulations and synthesis methods, (2) various accuracies of delay characterization, (3) synthesis of DDROs using direct solver versus heuristic method, (5) synthesis of DDROs using extracted data from STA, (6) synthesis of DDROs with different parameter settings of the solver and (7) quality of delay matching of DDROs versus generic ring oscillators (GROs). Chapter 5 explains the three different clustering methods investigated to group critical paths namely Hierarchical, KMeans++ and Expectation-Maximization algorithms. Furthermore, in order to evaluate the best suited method to cluster critical paths for DDRO synthesis, the results of the three algorithms are compared. In Chapter 6 DDROs are synthesized for each critical path cluster and are evaluated w.r.t their respective cluster. Chapter 7 summarizes the various DDRO synthesis results and concludes on the DDRO matching quality.

2 Background

The scaling of technology nodes has resulted in scaling of the process, temperature and voltage (PVT) variations. This scaling of PVT variations has a significant impact on the timing of digital circuits. This impact on timing performance in worst case can result in failure of manufactured chips. In order to establish error free timing performance of a chip, the different sources and impact of PVT variations have to be analyzed and modeled [27] [28]. This Chapter explains the different sources of manufacturing variations along with their classification. Furthermore, the cause and impact of environmental variations on timing of digital circuits are explained. Moreover, to verify the timing of a digital design, methods used to analyze the timing for digital circuits are explained. Additionally, the existing state-of-the-art timing monitors to evaluate the timing of a chip post-silicon are explained.

2.1 Process Variations

Process variations are the unintentional physical variations observed in the transistor parameters of devices and interconnects from intended theoretical design values [29]. These variations are observed in the form of deflected device characteristics from its actual intended behavior [30].

The procedure to manufacture integrated circuit is done on a silicon wafer along with several chemical and mechanical processes. At first, a desired material is deposited on a wafer, after which photo-lithography is performed to transfer the required pattern on to the wafer using light. Finally, etching is done to remove the unwanted material from the wafer. These processes are repeated several times to manufacture a complete chip. However, these chemical and mechanical steps are subjected to process variations which impacts the performance of manufactured devices within die as well as between die-to-die, wafer-to-wafer, and lot-to-lot [31].

The uniformity of the fabricated transistor is highly dependent on the precision and control of the lithography process. The wavelength of light used in the lithography process is $\lambda = 193$ nanometers and has remained the same since the 130nm technology node [32]. Due to this, there is an increased amount of diffraction occurring during the lithography process which results in critical dimensions (CD) and the device structure gets increasingly blurred [33]. Advanced process methods are developed which are used to reduce the blurring effect of the fabricated transistors. However, it is still highly challenging to reduce the absolute deviation of the physical parameters of the transistors.

The physical parameters of transistors which result in performance variation of these devices is given as follows: The critical dimensions (CD) of a MOSFET transistor are determined by the channel length (L) and width (W). Therefore, varying these parameters

from their nominal values result in significant variation from the expected performance of MOSFETs. Moreover, dielectric layer which isolates the gate electrode from the substrate has a specific thickness (Tox) and is critical to the device behavior. Additionally, threshold voltage (Vth) which is the minimum voltage required to switch on a transistor is another important parameter which defines MOSFET's performance. The drive current of the transistor is directly proportional to W/L. Therefore, fluctuations of the transistors width (W) and length (L) directly influence its drive current. Additionally, channel length variation also alters the transistor threshold voltage (Vth) which directly influences the drive current and thereby the switching time of transistors. The following Subsections explain the variations of transistor parameters due to different process variations.

2.1.1 Line Edge Roughness

Line edge roughness (LER) is the result of random variations at the edges of gate pattern resulting in roughness of the printed gate pattern edge. Photo-resists are light sensitive materials used to create required geometric patterns of transistors along with light insensitive material known as photo-masks. Light sources of specific wavelengths are used for the removal of photo-resists during the transistor manufacturing. LER is caused by statistical variations of the photon count or imperfections occurring during the removal of photo-resist and therefore, is a source of intrinsic parameter fluctuations in MOSFETs. Moreover, with decreasing technology nodes, the wavelength of the light source used in photo-lithography is larger than the critical dimensions of transistors which results in larger transistor gate variations. Thus, LER is an major source of variation of channel length [34] [35].

2.1.2 Random Dopant Fluctuation

The channel region of a transistor is doped with atoms of a material different from the substrate. This determines and controls the transistor threshold voltage (Vth). These doped atoms are known as impurity atoms or dopants. Addition of dopants is performed using ion implantation where the dopant atoms are randomly placed into the transistor channel. However, the process of ion implantation leads to statistical variations in the actual number of implanted dopants. The change in number of dopants in the channel regions affects the carrier potential energy which in turn affects the threshold voltage and current. Thereby impacting the drive strength of a transistor. In older technology nodes, thousands of dopant atoms were implanted in the channel region. Therefore, a deviation of few atoms had negligible impact on the transistor behavior. However, with the shrinking technology nodes, only tens of dopant atoms are used which results in larger mismatch of transistor behavior [36] [37].

2.1.3 Optical Proxity Effect

Optical proximity effect (OPE) is the variation in the size and shape of a transistor feature due to the proximity of other neighboring devices. OPE occurs at the step of

photo-lithography due to the diffraction of light with the neighboring structures. Polysilicon material is used as the gate contact and is effected due to OPE. Thus, OPE results in variations of the effective transistor length as a function of local layout. This results in process induced systematic intra-die variations [4].

2.1.4 Gate Oxide Thickness Variations

The gate oxide can be grown by using oxygen and nitrogen with an absolute accuracy of 12 inter atomic layers. Incorrect gas flow during the oxide growth can result in gate oxide thickness variations. Non-uniform or ultra-thin gate oxide thickness causes variations in threshold voltage across the wafer and can also cause gate leakage current. Moreover, former technologies had gate oxide thickness (TOX) of tens of inter atomic layers and had larger gates which resulted in negligible impact of TOX variations. However, in technologies below 30 nm with oxide thicknesses between 13 nm (approx. 515 inter atomic spacings), TOX variations can result in substantial uncertainty in threshold voltage [38].

2.2 Classification of Process Variations

Timing of a digital design is an important performance parameter that varies due to process variations. Therefore, the classification and effect of different sources of process variations have to be taken into consideration while modeling the switching time of a transistor or a digital logic gate. The sources of different process variation are classified as follows [29]:

2.2.1 Systematic Variations

Systematic or deterministic variations are produced due to the steps involved in the manufacturing flow. In the manufacturing steps, optical proximity effects caused during lithography process is one of the main root causes for the process variation. Optical proximity effects (OPE) mainly result in variation in layout design and fall into the category of systematic variation. Thus, the amount of variation introduced across dies and wafers is approximately the same and this can be corrected using techniques such as optical proximity correction (OPC) methods.

2.2.2 Non Systematic Variations

Non systematic variations are random variations which are generated from various process steps in the manufacturing flow. Due to the randomness of these variations, they are statistically modeled and are further classified based on their behavior into inter-die (global variation) and intra-die (local variation).

2.2.2.1 Inter-die Variation

Inter-die variations result in process variations which affect the performance parameters between two manufactured dies. In this case, the parameters within one die is assumed to be affected in the same way and result in a uniform value within the die. Figure 2.1 shows different types of inter-die variation.

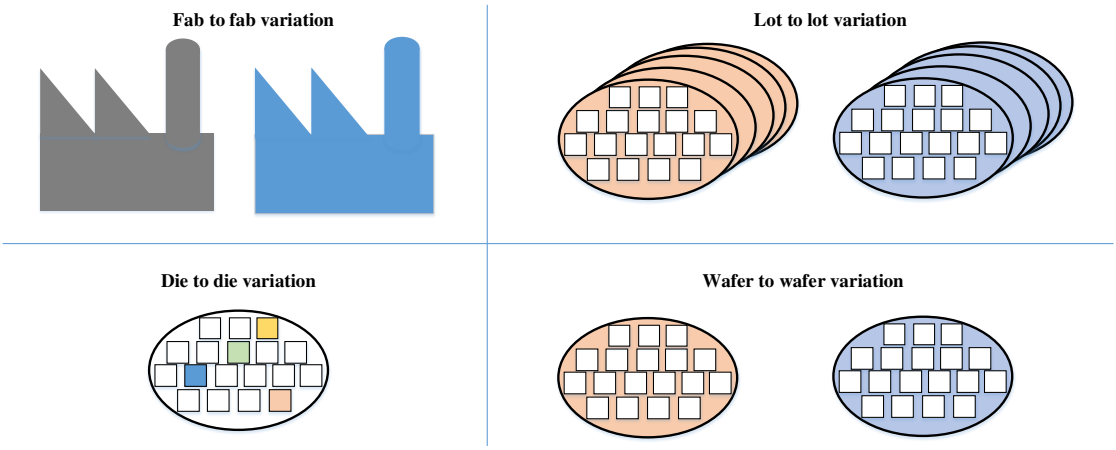


Figure 2.1: Different types of inter-die variation [29]

Die to die variations can occur due to the mis-alignment of masks exposed one after another in the lithography process. Wafer to wafer variations can occur due to inexact placement of the wafers in the equipment when processing different wafers. Lot to lot variations can occur due to different parameter settings during the working of the wafer equipment. Fab to fab variation can occur due to a specific condition set for the manufacturing of each of fab. An example of the impact of inter-die variations on chip frequency is shown in Figure 2.2. From Figure 2.2, it can be seen that the speed of the manufactured dies decreases with increase in distance from the center of the wafer. Thus, the dies lying in the inner concentric regions on the wafer on an average have higher speed than the outer concentric regions.

2.2.2.2 Intra-die Variation

Intra-die variations are classified as local variation and affect different parts of the same chip differently. Due to process variations, different devices on the same chip have different physical dimensions and result in difference in performance within the chip. With the decrease in technology node, the effect of inter-die variations have been increasing and contribute to a major part of the total process variation. Intra-die variations can be further classified as follows:

1. **Random Variations**: As the name suggests, these variations are random and not correlated to the variations in other parts of the chip. Two main causes of these

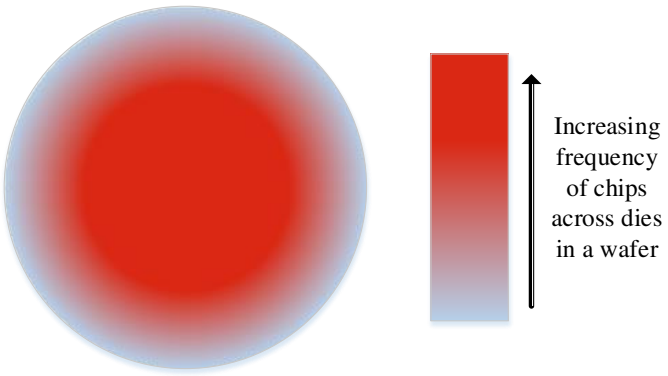


Figure 2.2: Variations across dies in a wafer [29]

variations are random dopant fluctuation (RDF) and line edge roughness (LER), as described in Subsections 2.1.2 and 2.1.1, respectively.

2. **Spatially Correlated Variations**: These include the process variations within the die where different parts of the die could have process variations correlated to each other due to the effect of certain proximity. Basic manufacturing effects such as etching and photo-lithography are the main causes for these variations. The physical dimensions of a transistor such as channel length (L), width (W) and oxide thickness is generally affected by spatially correlated variations.

2.3 Voltage Variations

With the decrease in technology nodes, environmental variations such as voltage and temperature also have larger impact on the performance of semiconductor devices. Moreover, for some power efficient designs, electronic circuits are operated in sub-threshold voltage regions. Small variations of voltage in sub-threshold regions can lead to erroneous performance of circuits. Voltage fluctuations in semiconductor devices are mainly caused due to the following reasons:

2.3.1 IR Drop

IR drop is also known as voltage drop, occurs due to the flow of current through the parasitic resistance of the power supply grid. The voltage fluctuation is given by Ohm's law

$$\Delta V_{IRdrop} = R_{grid} \cdot i(t) \tag{2.1}$$

where ΔV_{IRdrop} is the voltage fluctuation, R_{grid} is the parasitic capacitance of the power grid, i is the current which is dependent on time t.

2.3.2 Current Derivative Noise

Current derivative noise occurs due to parasitic inductance. Packaging of semiconductors does not scale as fast as scaling of the CMOS technology node which introduces larger parasitic inductance. This results in larger current noise with the decrease in technology node. Voltage variation due to current derivative noise is given by:

$$\Delta V_{\frac{di}{dt}} = L_{parasitic}\frac{di}{dt} \tag{2.2}$$

where $\Delta V_{\frac{di}{dt}}$ is the voltage variation due to current derivative noise and $L_{parasitic}$ is the parasitic inductance. Furthermore, the combination of voltage droop and current derivative noise could either lead to voltage drops or to voltage overshoots [2] [39]. The following section describes the impact of voltage variation on timing of digital circuits.

2.3.3 Impact of Voltage on Timing of Digital Circuits

The impact of voltage variation on circuit timing for a specific temperature is shown in Figure 2.3. It can be seen that the delay of circuit increases with decrease in voltage. The relationship between CMOS logic gate propagation delay and supply voltage can be written as [2]:

$$t_{gate} \propto \frac{V_{DD}}{(V_{DD} - V_{TH})^{\theta}} \tag{2.3}$$

where t_{gate} is the delay of a CMOS logic gate, V_{DD} is the supply voltage, V_{TH} is the threshold voltage and θ is the technology dependent parameter. It can be seen that small changes in V_{DD} close to the value of V_{TH} result in larger changes in gate delay t_{gate}. In order to evaluate the voltage changes on the semiconductor device and account for the timing variation, on chip test structures or sensors are required to monitor the voltage variations.

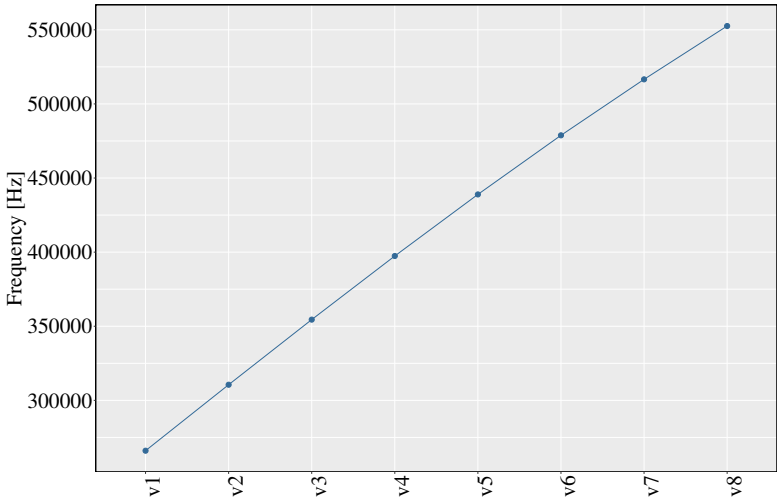


Figure 2.3: Frequency of a ring oscillator with increasing voltage at a constant temperature

2.4 Temperature Variations

Temperature variation is yet another major factor for performance variation on semiconductor devices. The main causes for the changes in temperature on device are (1) changes in the ambient temperature and (2) local temperature variations due to power dissipation in transistors at the region of high activity, also known as hot spots [2] [39]. Temperature variations also have time constants which can vary between milliseconds to seconds.

Figure 2.4 shows the impact of circuit delay with varying device temperature at different voltages. Due to the decrease in carrier mobility in the semiconductor and increase in parasitic resistance of the interconnect, an increase in the temperature increases the delay of a circuit. However, in smaller technology nodes a new effect namely the inverted temperature effect (ITE) has become prevalent where at lower voltages, the delay of a circuit decreases with increase in temperature.

In summary, it can be seen that the delay of a circuit is highly impacted by process, temperature and voltage variations. Therefore, digital circuits have to be extensively analyzed for their timing behavior. To ensure proper functioning of circuits, timing analysis should ensure that all timing constraints are met across all possible different conditions of PVT. The next section explains about the methods that are used for the analysis of circuit timing.

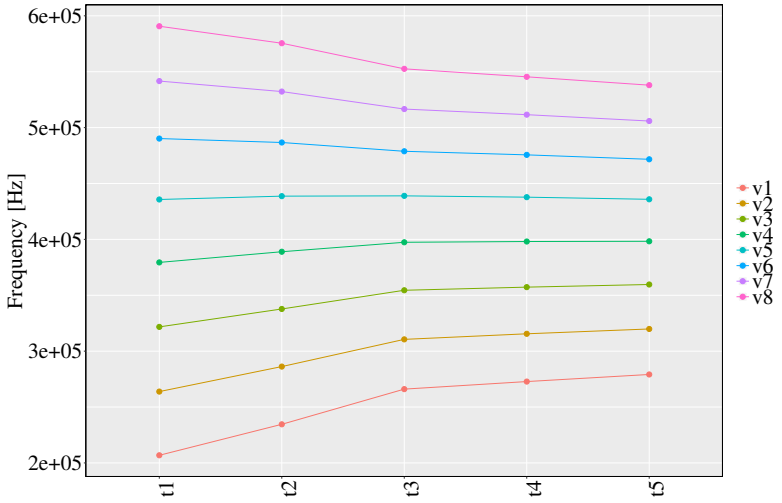


Figure 2.4: Frequency of a ring oscillator with increasing temperature at different voltages

2.5 Timing of Digital Circuits

Digital circuits consists of a combination of logical and synchronous elements. Moreover, these circuits can be classified into data path and clock path. Data paths are the ones where the data signal transmits through the circuit. Clock paths are the ones where the clock signal flows to different sequential cells. The logic portion of the digital data path consists purely of CMOS gates without memory and are built using NMOS and PMOS transistors. Edge-triggered flip-flops are used as sequential cells in synchronous digital circuits [40]. The clock path mainly connecting the flip-flops used in the synchronous circuits also consists of certain logic gates. In very large scale industrial (VLSI) designs, basic logic gates and flip flops are the building blocks of larger digital circuits and they are pre-designed in the form of standard cells. A collection of standard cells for a certain technology node compose a standard cell library.

In order to evaluate the performance of digital circuits, transistors which are the smallest building blocks of these digital circuits are modeled for each technology node using transistor parameters based on a standard known as BSIM parameters [41]. These transistor models are then used to pre-characterize the functionality and performance of the standard cells in terms of timing, power, etc.

Figure 2.5 shows the propagation delay or timing of an inverter cell. The time taken between the output transition (Z) reaching 50% of its final value and input transition (A) reaching 50% of its final value is termed as propagation delay of a cell. Based on the timing analysis of standard cells, the propagation delay of a logic gate depends on (1) process parameters (P), (2) voltage (V), (3) temperature (T), (4) input transition time or slope of input signal, (5) output RC load, (6) timing arc, and (7) rising or falling transition at the input pin as shown in Figure 2.5.

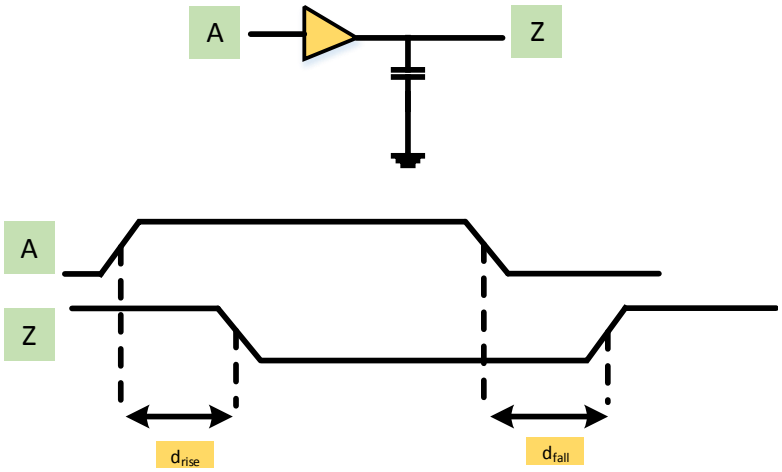


Figure 2.5: Propagation delay of a standard cell

Figure 2.6 illustrates an example of a digital circuit where a logical path is sandwiched between a launch and a capture flip-flop. The time taken between the clocking of data at the launch flip flop at $Q1$ in Figure 2.6 and the arrival of the data at the capture flip-flop $D2$ is known as the timing of data path in a combinatorial digital path. The clocking of data at the launch flip-flop also considers the time taken for the data to propagate between $CLK1$ to $Q1$. Thus, the clock-to-Q-delay of the launch flip-flop is also added to compute the propagation delay of the data path.

Additionally, there also exist clock paths which are timing paths that are completely traversed by clock signals. Generally only buffers and inverters gates are used in the clock paths. However, gated clock paths are sometimes used for power saving where a gated clock path could have other logic gates such as AND, NAND gates. Thus, there exists two different kinds of clock paths:

1. **Launch path**: This is the timing path of the clock signal between the clock source and the launch flip-flop.

2. **Capture path**: This is the timing path of the clock signal between the clock source and the capture flip-flop

Simple edge triggered flip-flops used for synchronous circuits have typically two inputs and one output pin. The input pins are the clock and data signal input pins. A clock signal with a specific time period is given into the clock pin. The data signal pin receives its input either from an input port or another logic circuit. The output pin of a flip-flop is the data which is captured by the edge of the clock input. The amount of time taken for the clock to propagate the data signal is known as clock-to-Q-delay of the flip-flop. Moreover, for the data to be clocked in correctly and maintain signal integrity, flip-flops define two inherent features which are explained as follows [42]:

1. **Setup time**: In an ideal flip-flop, the data signal (D) can arrive at the time when the clock edge is triggered. However, in practical scenarios for the clock edge to

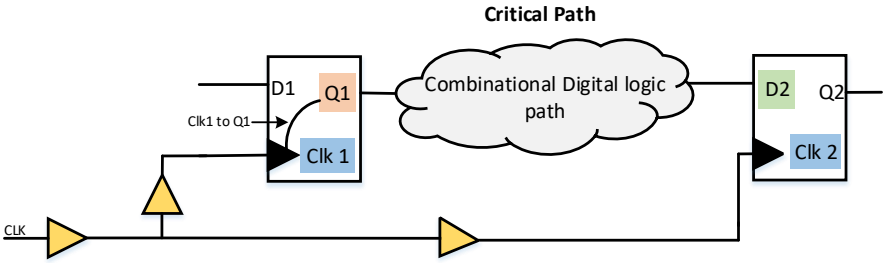


Figure 2.6: Example of a combinatorial digital path

capture the data signal correctly, the data signal must arrive earlier than the clock edge and remain stable at the time of arrival of the clock edge. The amount of time the data must arrive before the clock edge for the clock to safely capture the data signal is known as the setup time of the flip-flop. T_S in Figure 2.7 shows the setup time for a flip-flop.

2. **Hold time**: In order to ensure data signal integrity, the data signal (D) should remain stable for a certain amount of time after the transition of the active edge of the clock. This time is known as the hold time of the flip-flop. T_H in Figure 2.7 shows the hold time for a flip-flop.

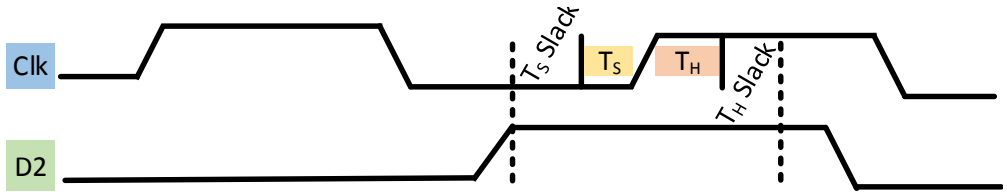


Figure 2.7: Example of timing checks at a capture flip-flop for a combinatorial digital path

An industrial design of digital circuits could consist of millions of digital paths consisting of synchronous and logic circuits. In order to ensure correct working of all these paths, the timing constraints such as setup and hold times explained earlier should be met, i.e. for setup time, the data signal (D) should arrive before the setup time of the flip-flop, and for hold time, the data signal should remain stable for a certain time. The set of digital paths in synchronous logic circuits which have the largest and smallest propagation delays are known as critical paths of a digital design. These critical paths tend to violate the setup and hold timing constraints. Critical paths whose data signal (D) comes after the required setup time T_S of a flip-flop result in a setup time violation. Critical paths whose data signal (D) changes before the required hold time T_H of a flip-flop results in a hold-time violation.

In order to check the timing violations of a digital design, extensive timing analysis of circuits are to be performed and all the timing conditions of a design should be met. Moreover, as explained earlier these paths also undergo timing variations due to PVT

variations. Thus, the timing analysis performed should also consider these variations to ensure correctness of timing. The following Subsections explain two different methods for analyzing timing of digital circuits.

2.6 Timing Analysis of Digital Circuits

2.6.1 SPICE

SPICE is a software which can simulate the different electrical performances of an electronic circuit. SPICE uses mathematical models of transistors and other common circuit components such as resistors, capacitors, etc. to simulate the given digital circuits. The input parameters for timing analysis using SPICE simulation are:

1. SPICE netlist of the given set of digital synchronous circuit.
2. Netlist of the standard cells used in the digital circuits.
3. SPICE models for different electrical devices for different conditions which includes the models for global and local variations.
4. SPICE deck which contains the description of the timing analysis to be performed.

SPICE based timing analysis is more accurate than other methods due to extensive characterization of each standard cell and parasitic RC and specific conditions of input parameters. However, due to this extensive analysis, the SPICE based timing analysis for thousands of paths is computationally very complex and results in non-feasible CPU run-times [43].

2.6.2 Static Timing Analysis

Static timing analysis (STA) is an exhaustive method to analyze and verify timing of digital circuits. The main motivation of using STA instead of SPICE is due to the reduced CPU run-times through which tens of thousands of critical paths of a design can be analyzed and timing verified in a feasible run-time. Moreover, with SPICE it would be extremely complex to create the environment for timing analysis and perform an exhaustive timing analysis on the entire digital design. Furthermore, it is also impossible to check local variations of all the critical paths with SPICE based simulations, thus proving that STA is a better tool to evaluate timing of a chip.

First, delays of the standard cells are pre-characterized using SPICE simulation for various PVT conditions, input transition times and output loads. These pre-characterized delays are stored and used for STA. To run STA, a digital design along with the clock definitions and specification regarding the external environment such as input-output ports and delay constraints are given. STA then validates whether the given digital design functions without any timing violations at given clock frequencies and supply voltage conditions by performing setup and hold timing checks. Moreover, with STA

the entire digital design with all the timing checks for all possible digital paths and design scenarios can be analyzed in one run [44] [45]. However, the worst case delay at each standard cell is assumed during STA which results in pessimistic timing analysis. This in turn influences the performance of circuits and also results in increased area, power dissipation and cost.

Therefore, to reduce the pessimism in guardbands, the timing of each chip should be evaluated accurately. Timing monitors are used to evaluate the timing of a design or a technology node after manufacturing on different chips and different parts of the chip. Based on the measured timing from these timing monitors across various conditions of PVT, the guardbands can be reduced by decreasing the operating supply voltage and/or increasing the speed of the chip.

2.7 Timing Monitors: Related Work

Timing monitors are test structures which are placed on a chip. These test structures are used to evaluate the influence of PVT variations on the timing of digital circuits. The state-of-the-art test structures are designed either to monitor the timing of a specific technology node or to monitor the timing of a specific chip design. The different state-of-the-art timing monitoring methods are described in the below Subsections.

2.7.1 Path Delay Tests

With the decrease in technology nodes, there is an increase in the mismatch between the results of timing analysis using timing models and silicon. This may have an impact on the circuit performance and correct functioning of the digital system. Therefore, there is a need to validate the timing on silicon. One of the methods of validating timing on silicon is by measuring the path delays on the chip [11]. Path delay tests aims to analyze the overall delay of path or the sum of the delays of each element on the path [46]. Scan chain testing is used as a method to analyze and detect timing errors on silicon.

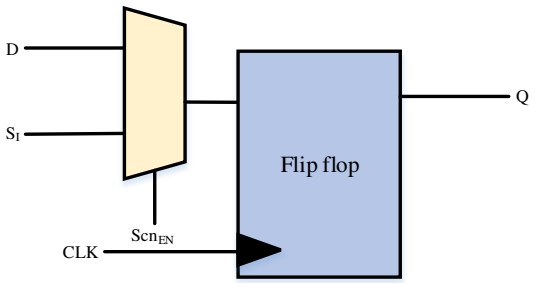


Figure 2.8: Scan flip-flop

The structure of a scan flip-flop is shown in Figure 2.8. The scan flip-flop uses a 2 input multiplexer where one input is the data input (D) for functional mode of the circuit. The second input is the scan input signal (S_I). The enable signal (Scn_{EN}) controls if

the circuit works in scan mode or functional mode. The output signal (Q) gives the captured data at the flip-flop. The flip-flop is also clocked by a clock signal (CLK). The scan flip-flop is the building block of a scan chain. The scan chain also requires input and output ports which are used to read-in and read-out the data through the scan chain. The first and the last flip-flops of the scan chain are connected to the scan in port and scan out port, respectively. Figure 2.9 illustrates a scan chain with input clock (CLK), scan input, scan enable signal and scan output. Delay path testing with scan chain is performed to detect variations in timing observed in the combinatorial logic block. The output of the scan chain is verified for correctness based on the input of the scan chain. This helps in understanding how much slower or faster the delay path is, in comparison to the prediction using timing models. Additionally, scan chains are used mainly for fault testing of functional circuits.

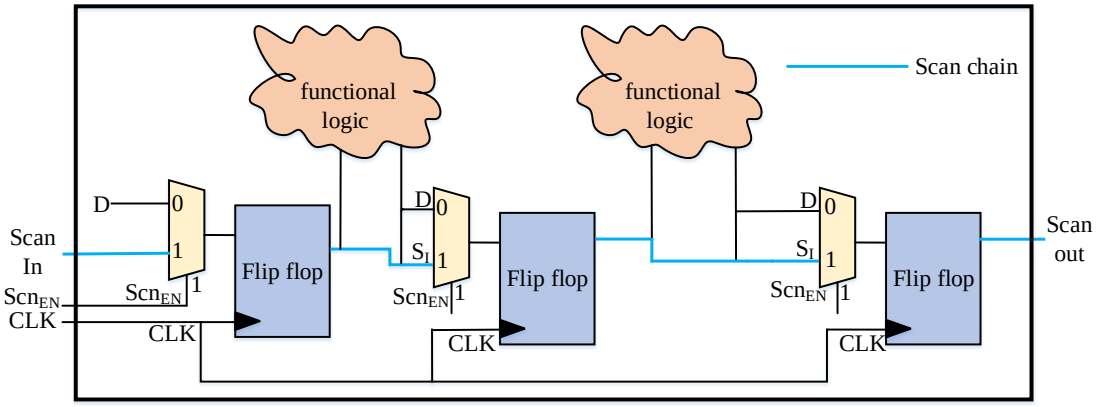


Figure 2.9: Scan chain

However, path delay tests have several disadvantages. By measuring only the path delays it would be difficult to extract the variations in PVT parameters and the model parameters. Furthermore, addition of a scan MUX, input and output scan ports are required for each scan chain. This results in higher area and power consumption. Moreover, a digital design consists of tens of thousands of paths and it would not be feasible to perform path delay testing on the entire design to verify timing.

2.7.2 Generic Ring Oscillators

Ring oscillators are standalone test structures which could be designed either based on a product or a technology node. These ring oscillators could be operated at frequency conditions similar to a product application. Additionally, the measurements from ring oscillators (ROs) are fundamentally closer to the behavior of a technology node where the random process variations are canceled due to averaging over the length of the ROs. ROs are used together with frequency divider circuits to enable ease of measurement of the frequency of ROs. Furthermore, ROs also aid in the analysis of power performance for a specific technology node. Ring oscillator test structures are also testable early in the development process of technology and characterization.

ROs can be designed to represent PVT variations such that the timing of a chip can be diagnosed during post silicon validation testing. The method proposed in [47] is to synthesize single cell type ring oscillators. Multiple ROs are generated from different cell types which show different effects of process variations. Since, these ring oscillators are not synthesized based on a digital design they are also known as generic ring oscillators (GROs). Also, the silicon measurement data from these ring oscillators are analyzed and either the different process parameters are modeled or the existing models of PVT variations are verified.

Figure 2.10 shows an example of a single-cell type ring oscillator. A single cell type is connected in chain with a feedback loop to oscillate. In order to oscillate, the connected cell types should be inverting. Moreover, an enable signal is used for enabling the oscillation. As seen in Figure 2.10, ROs are standalone structures and therefore complex methods of delay paths selection and activation need not be performed.

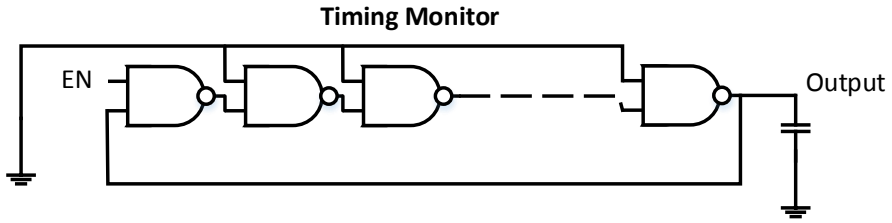


Figure 2.10: Example of a single-cell type ring oscillator

However, since GROs are not design specific they cannot predict the timing of a design. In order to predict the timing of a specific design, new methods of developing design-dependent timing monitor are explained in the next two sections.

2.7.3 Critical Path Replica Circuits

Critical path replica circuits are used to evaluate the timing a chip across PVT variations. The main advantage of the replica circuits is that they are designed to evaluate the timing behavior of a specific digital design. In [48] [49] [50], critical paths from a design are selected and replicated in the form of standalone test structures. The measurement results from these test structures directly correlate the timing behavior of the selected critical paths. However, a digital design could have thousands of critical paths and it is not feasible to generate a replica path for each critical path on a design.

In order to identify a feasible number of critical paths to generate replica circuits as given in [51], critical paths which have similar timing behavior are identified. After this, a set of minimum number of critical paths are sub-selected for which replica circuits are synthesized. In [52] and [53], the selection of critical paths are further improved by the usage of linear algebra and machine learning to dynamically select a set of critical paths monitors which can predict the timing behavior of a chip across PVT conditions.

Yet another method is proposed in [20] to synthesize test structures which replicate critical paths. An automated synthesis algorithm is used to build on-chip test structures

that can replicates the various effects of parameters variations of all the critical paths on a chip. The synthesis method identifies the worst case delay of all the critical paths due to intra-die and inter-die variations on a chip and synthesizes a test structure to represent the worst case delay. The measurement on this replica circuit could be used to predict the worst case circuit delay of the entire chip. However, this test structure cannot evaluate random intra-die variations. Moreover, critical paths on a chip have very different timing behavior across different PVT conditions. Therefore, a single test structure will not be able to accurately represent the timing behavior of all the critical paths on a chip.

In order to overcome the disadvantage of critical path replica test structures, tunable replica circuits (TRC) are introduced in [21]. TRCs are built with different logic gate stages such as NAND and NOR stages. These logic stages are placed next to each pipeline stage of the functional circuit and are tuned to track the critical path delays at the respective pipeline stages. The number of TRCs required are much lesser than replica circuits in [48] [49] [50]. Moreover, unlike the replica circuits and generic ring oscillator test structures they can also detect local variations. However, TRCs are intrusive and are extremely complex to tune to replicate the timing behavior of critical paths.

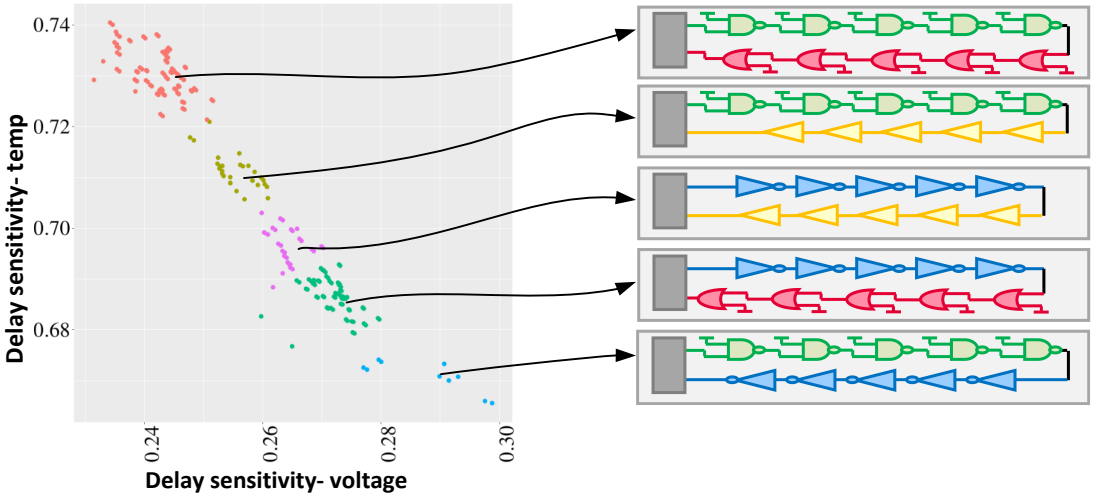


Figure 2.11: Clustering of similar behaving critical paths and DDRO synthesis for a cluster of critical paths

2.7.4 DDROs

Design-dependent ring oscillators (DDROs) are standalone ring oscillator test structures which are designed to track the timing behavior of critical paths of a digital design. The DDROs are synthesized to match the delay sensitivities of critical paths. A set of critical paths for a digital design are identified and grouped together based on the similarity in their delay sensitivity behavior across PVT conditions. As shown in Figure 2.11, a DDRO is then synthesized per group of critical paths [22].

Figure 2.12 shows a method used in [22] to synthesize DDROs. At first, critical paths delay sensitivities are obtained of critical paths and similar behaving critical paths are grouped together. The next step is to synthesize one DDRO per critical path group such that the delay sensitivities of a DDRO match the delay sensitivities of the critical path group. In the DDRO synthesis, tiles are the building blocks of DDROs where tiles are made of chains of standard cells. Figure 2.13 shows an inverter tile example where a tile is made of 5 standard cells [22]. A DDRO is then synthesized using integer linear programming for a target critical path group. The integer linear programmer builds a DDRO by concatenation of tiles by matching the delay sensitivities of target critical path group by the summation of linear sensitivities of the selected tiles. The frequencies of DDROs on the chip are measured on silicon and assessed by which the timing of the target critical path group is evaluated.

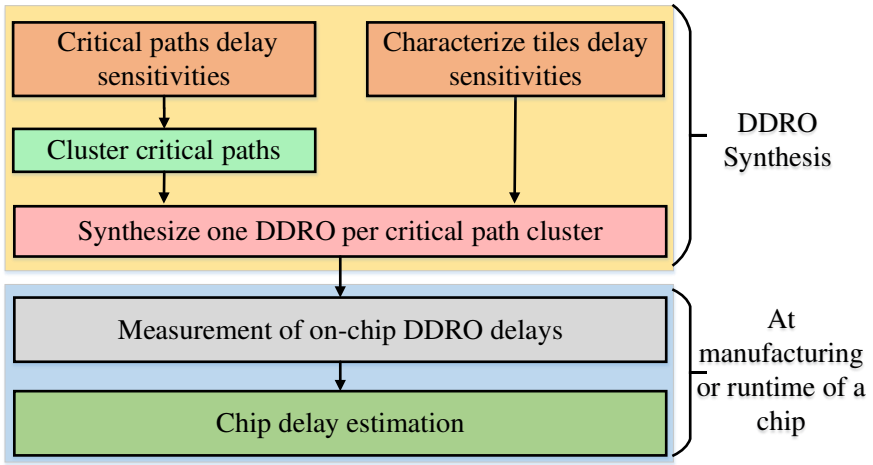


Figure 2.12: State-of-the-art DDRO synthesis method

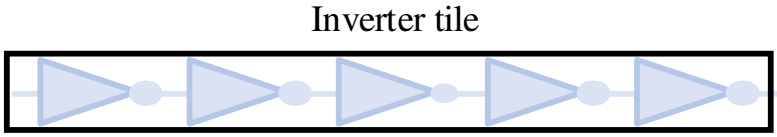


Figure 2.13: Example of a tile in a DDRO

There are several advantages of using DDROs in comparison to other timing monitors. DDROs are more accurate in comparison to having a single critical path monitor because multiple DDROs are synthesized for representing critical paths. Moreover, DDROs are non-intrusive test structures. The required number of DDROs to be placed on a chip is greatly reduced due to the synthesis of one DDRO per group of critical paths. Therefore, only few DDROs are required to accurately estimate the timing of all the critical paths on a chip. Furthermore, DDROs can be used in the early manufacturing stage for process tuning, guard-band reduction and also in the later stage for real-time timing prediction. The change of the monitor's purpose is done by redefining the target sources of variation.

However, these DDROs do not consider the impact of input transition time and output load during the synthesis. Moreover, due to the concept of sensitivity matching, these DDROs are forced to use SPICE-based delay characterization which makes it not feasible for large-scale industrial designs. Additionally, the majorly contributing process parameters impacting the delay sensitivities have to be evaluated for every technology node since the process parameters affecting the delay sensitivities differ for each technology node. Furthermore, computation of delay sensitivities with high accuracy is extremely challenging for the DDRO synthesis. Additionally, the state-of-art DDROs rely on the concept that matching delay sensitivities will create DDROs which can track the delays of critical path groups. Thus, given that the sensitivity based DDROs have several disadvantages, they might result in tracking delay with poor quality to their respective critical paths. Thus, in this work, further investigations are performed on the synthesis of DDROs to improve the quality of DDROs which enables them to accurately track the delays of critical paths on a chip across various conditions of PVT.

3 Design-Dependent Timing Monitors

The concept of the state-of-the-art design-dependent ring oscillators (DDROs) as an on chip timing monitor in [22] is introduced in Sec. 2.7 to improve performance of digital CMOS circuits and design closer to the edge. DDROs, as the name suggests are design specific monitors and are designed to represent critical paths on a chip design. In this chapter, in order to further improve the timing evaluation capabilities of DDROs and provide a more robust design, the following novel methods of DDRO synthesis are proposed: (1) in addition to the linear programming in the state-of-the-art DDROs in [22], quadratic objective is proposed in the DDRO formulation for matching delay sensitivities, (2) to improve DDROs matching quality, more accurate methods of 2-tile and 3-tile delay characterizations of building blocks of DDROs are described in addition to the state-of-the-art 1-tile delay characterization, (3) the implementation of DDRO synthesis formulation is explained for quadratic programming using a direct solver, (4) a new concept for the synthesis of DDROs is presented by tracking the delay of critical paths which can use STA data, thus, enabling their application in large scale industrial design, and (5) three new heuristic methods are developed as an alternative to direct solvers to solve the delay-tracking-based DDRO synthesis problem.

3.1 Design Goal of a DDRO

The fundamental goal of a DDRO as shown in Figure 3.1 is to mimic the delay behavior of a critical path or a group of critical paths across PVT conditions. The delay of a critical path varies non-linearly across PVT conditions and the aim of DDROs is to track the delay variations of the critical paths across the different PVT conditions. Frequency measurements of these DDROs on silicon represent the timing of critical paths and thereby monitors the timing of a chip. The main goal of this work is to further increase the DDRO delay tracking accuracy and to facilitate the DDRO synthesis for large-scale designs.

A DDRO is constructed by concatenating tiles of different types. The different tile types are combined in a specific manner to represent the timing behavior of the target critical path group. In this thesis, the DDROs are constructed by tiles which are chains of standard cells of equal type. Figure 3.2 gives an example of DDRO constructed from tile concatenations. The aim of using tiles instead of individual standard cells is to reduce the dependency of delay variations with respect to the output load and input slew. The timing of individual standard cells in a standard cell library are highly influenced by input conditions such as input slope and output load. In order to reduce this influence of input conditions, tiles are used in the synthesis of DDROs. A tile's inner most cells

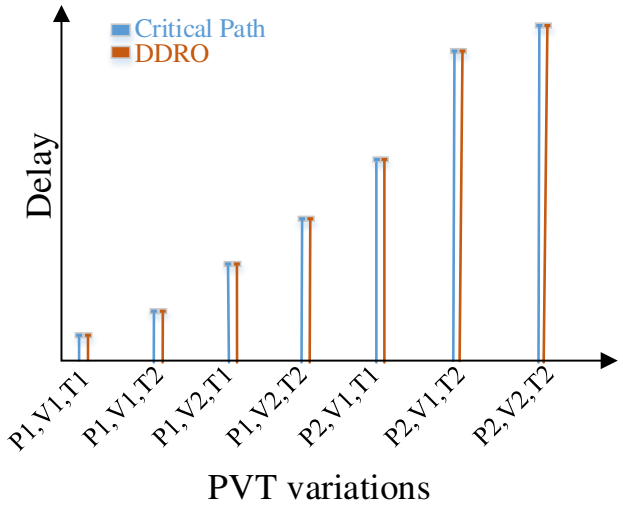


Figure 3.1: Goal of delay-tracking DDRO: The behavior of the DDRO mimics the behavior of the critical path over the PVT parameters

are shielded from the impact of input slew and output load by the first and last few cells respectively.

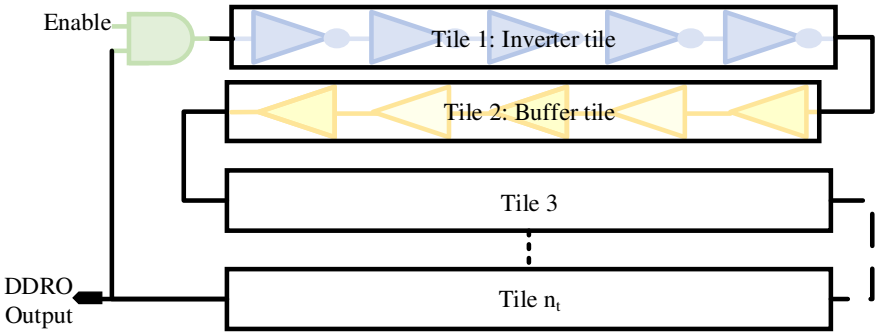


Figure 3.2: Structure of a DDRO: Concatenation of tiles

Figure 3.3 shows the structural outline of synthesis and evaluation of DDROs. A collection of types of standard cells are chosen and tiles are constructed for a selected tile length. The delay attributes of the constructed tiles are characterized across various PVT conditions. Furthermore, a set of critical paths on a design are identified and their timing attributes are characterized at various PVT conditions. In order to synthesize DDROs for a large scale industrial chip design, the number of synthesized DDROs should be limited to a feasible number. For this reason, critical paths are grouped based on their similarity in timing behavior and one DDRO is synthesized for each of the critical path groups using optimization methods. Synthesized DDROs are evaluated by comparing their delay-tracking behavior to their given respective critical path group.

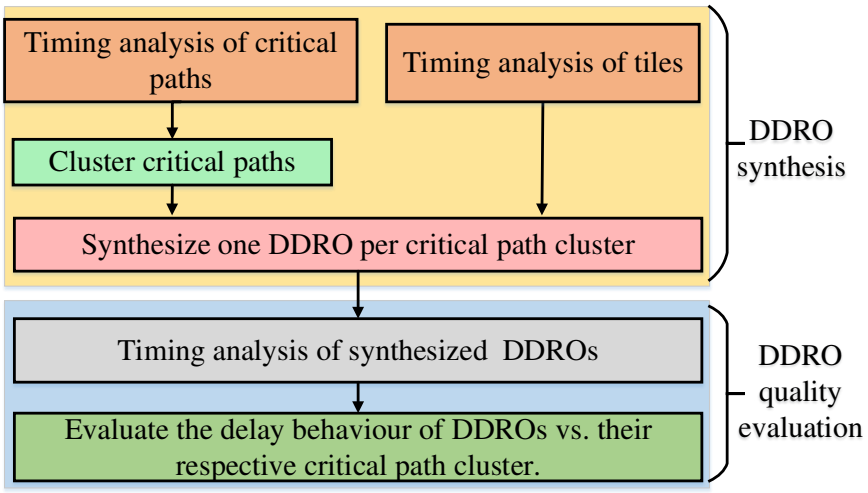


Figure 3.3: Flow diagram for DDRO synthesis and evaluation

In this section, the concept of synthesizing DDROs is explained in detail. Sec. 3.2 presents the concept and formulation of the existing state-of-the-art sensitivity based DDROs. In addition to integer linear programming used previously in [22], a new formulation is described to improve DDRO delay matching in Sec. 3.2. In order to further reduce the impact of input slew and load in tiles, new methods of characterizing tile delays are explained in Sec. 3.3. Sec. 3.4 presents the implementation method of DDRO algorithms using traditional optimization solvers. In order to overcome the drawbacks of sensitivity based DDROs, Sec. 3.5 proposes a novel idea for the synthesis of DDROs by matching delays instead of delay sensitivities to critical paths. In Sec. 3.6, the drawbacks of using traditional solvers for DDRO synthesis are presented. Moreover, new Heuristic algorithms to optimize the DDRO objectives are explained, which overcome the drawbacks of traditional solvers.

3.2 Formulation of Sensitivity-Based DDRO Design

This section first explains the formulation of state-of-the-art sensitivity-based DDROs. Later, a new optimization method is proposed to improve the DDRO's matching quality. Figure 3.4 illustrates a DDRO path comprising of n_t tiles where d_i is the delay of a tile at position i. In the DDRO synthesis, a specific tile type is selected for each tile position. $\varphi(i)$ refers to the cell type used in the construction of tile at position i. The set of selected types of tiles in the tile library are built from standard cells such as $AND, OR, NAND, etc.$ of various drive strengths and from different standard cell libraries. The number of cell types in the tile library is denoted as n_φ.

In order to match the delay sensitivities of DDROs to critical paths [22], delay sensitivities of tiles and critical paths are characterized, by the central difference method. Delay sensitivities by central difference method are calculated by varying each of the

PVT parameters $\mathbf{p}$ from the nominal value $\mathbf{p}_0$ by a small variation $+\Delta\mathbf{p}$ above the nominal value and $-\Delta\mathbf{p}$ below the nominal value [54] [25]:

$$\nabla d_Y(\mathbf{p}_0) \approx \frac{d_Y(\mathbf{p}_0 + \Delta\mathbf{p}) - d_Y(\mathbf{p}_0 - \Delta\mathbf{p})}{2 \cdot \Delta\mathbf{p}} \tag{3.1}$$

In Equation (3.1), the total number of parameters considered for the sensitivity analysis is n_P and $d_Y(\mathbf{p}_0)$ is the delay of a circuit such as the propagation delay of a critical path $d_{CP}(\mathbf{p}_0)$ or a tile delay $d_i(\mathbf{p}_0)$. SPICE simulation is used to perform the delay sensitivity analysis [55].

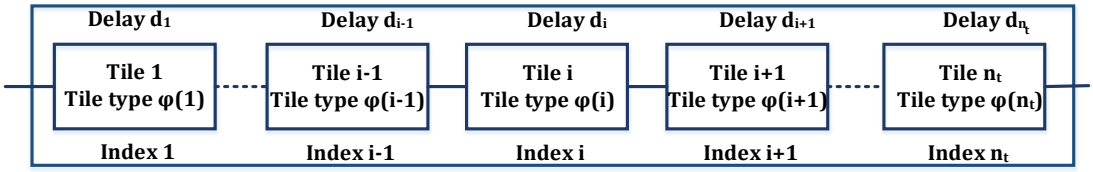


Figure 3.4: DDRO path structure

For the DDRO synthesis, the considered PVT parameters $\mathbf{p}$ are globally varying and statistically independent. From Figure 3.4, the DDRO path is constructed by concatenating n_t tiles. Therefore, the delay of a DDRO path is the summation of delays of n_t tiles.

$$d_{DDRO}^{(X)} = \sum_{i=1}^{n_t} d_i^{(X)}(\mathbf{p}) \tag{3.2}$$

The superscript (X) denotes the method of characterizing the tile delay sensitivities and is explained in Sec. 3.3. A linear model of tile delay $d_i^{(X)}(\mathbf{p})$ is assumed around $\mathbf{p}_0$ by a variation in the PVT parameters of $\Delta\mathbf{p}$ using the sensitivity from Equation (3.1):

$$\begin{aligned} d_i^{(X)}(\mathbf{p}_0 + \Delta\mathbf{p}) = & d_i^{(X)}(\mathbf{p}_0) + \\ & \nabla d_i^{(X)}(\mathbf{p}_0)^T \cdot \Delta\mathbf{p} \end{aligned} \tag{3.3}$$

Inserting Equation (3.3) into (3.2) leads to a linear model of the DDRO path delay $d_{DDRO}^{(X)}$ around $\mathbf{p}_0$ by varying the parameters by $\Delta\mathbf{p}$

$$\begin{aligned} d_{DDRO}^{(X)}(\mathbf{p}_0 + \Delta\mathbf{p}) = & \sum_{i=1}^{n_t} d_i^{(X)}(\mathbf{p}_0) + \\ & \sum_{i=1}^{n_t} \nabla d_i^{(X)}(\mathbf{p}_0)^T \cdot \Delta\mathbf{p} \end{aligned} \tag{3.4}$$

According to Equation (3.4), the DDRO delay sensitivity is the sum of all its tile delay sensitivities. The DDRO optimization goal is then devised to select the number of tiles n_t as well as the type $\varphi(i)$ of each tile position $\{i = 1, \ldots, n_t\}$, such that the critical path sensitivity matches the DDRO sensitivity:

$$\underset{n_t, \varphi(i), i=\{1,\ldots,n_t\}}{min} \left\| \nabla d'_{CP}(\mathbf{p}_0) \cdot n_t - \sum_{i=1}^{n_t} \nabla d_i^{'(X)}(\mathbf{p}_0) \right\|_{pn} \tag{3.5}$$

pn specifies the type of vector norm. When $pn = 1$, l_1 norm or linear (L) optimization is used similar to the state-of-the-art DDROs [22] and when $pn = 2$, l_2 norm or quadratic (Q) optimization is used. The characterized delay sensitivities are normalized for tiles and critical paths to remove the physical units of different PVT parameters and to improve the problem condition of the objective by equalizing the value ranges among the different PVT parameters:

$$\begin{aligned} \nabla d_i^{'(X)}(\mathbf{p}_0) &= \frac{\nabla d_i^{(X)}(\mathbf{p}_0)^T \cdot \Delta\mathbf{p}}{d_i^{(X)}(\mathbf{p}_0)} \\ \nabla d'_{CP}(\mathbf{p}_0) &= \frac{\nabla d_{CP}(\mathbf{p}_0)^T \cdot \Delta\mathbf{p}}{d_{CP}(\mathbf{p}_0)} \end{aligned} \tag{3.6}$$

For the synthesis of DDROs, delays are analyzed at various PVT corners to evaluate the worst case timing of digital circuits. Digital circuits mostly include process corners namely nominal (NOM), fast (FAST) and slow (SLOW). The synthesized DDROs should consider the sensitivities of the target critical paths at all the timing corners. Sec. 3.2.1 describes the method that is used to include multiple PVT corners.

3.2.1 Delay Sensitivity Matching at Multiple PVT Corners

The synthesized DDROs by the method of delay-sensitivity matching are to be tracked across various conditions of PVT. Therefore, the delay sensitivities of tiles and critical paths across PVT conditions are collected together in a vector for the delay-sensitivity matching:

$$\nabla d_{CP}(\mathbf{p}_{PVT}) = \begin{bmatrix} \nabla d_{CP}(\mathbf{p}_{PVT_1}) \\ \nabla d_{CP}(\mathbf{p}_{PVT_2}) \\ \cdots \\ \nabla d_{CP}(\mathbf{p}_{PVT_{n_{PVT}}}) \end{bmatrix} \tag{3.7}$$

$$\nabla d_i^{(X)}(\mathbf{p}_{PVT}) = \begin{bmatrix} \nabla d_i^{(X)}(\mathbf{p}_{PVT_1}) \\ \nabla d_i^{(X)}(\mathbf{p}_{PVT_1}) \\ \cdots \\ \nabla d_i^{(X)}(\mathbf{p}_{PVT_{n_{PVT}}}) \end{bmatrix} \tag{3.8}$$

where $\{PVT_1, PVT_2, \ldots, PVT_{n_{PVT}}\}$ are the various PVT conditions and n_{PVT} is the number of PVT corners. The conventional method to characterize delay sensitivities using analog SPICE simulation has high computational complexity because it uses transistor models. As mentioned in Sec. 3.2, for sensitivity-based DDRO synthesis, the delay-sensitivities w.r.t. PVT parameters are characterized using central difference method [55]. Thereby, the number of simulations for each timing arch is $3 \cdot n_p \cdot n_{PVT}$, where n_p is the number of PVT parameters. Thus, delay sensitivity characterization using SPICE based timing analysis for thousands of critical paths is not feasible. In addition, the transistor parameters which vary globally and are statistically independent and should be identified for each of the standard cell libraries for every new technology node. Moreover, the major contributing parameters which have high influence on delay are unique for each standard cell library. These parameters have to be ascertained through extensive data analysis of delay sensitivities of tiles or standard cells. In order to avoid extensive data analysis of delay sensitivities and the transistor parameters, a new method to synthesize DDROs is explained in Sec. 3.5 which uses delays instead of delay sensitivities, thus eliminating the computational effort to compute the delay sensitivities.

3.3 Characterization of Tile Delay Sensitivities

As mentioned before, tiles are used in the synthesis of DDROs where tiles are concatenated set of standard cells. The state-of-the-art method in [22] uses 1-tile delay characterization as explained in 3.3.1. In order to be able to linearly concatenate tile delays, the characterized tile delays need to be highly accurate. Moreover, tile delays and tile delay sensitivities using 1-tile delay characterization are affected by their input slope and output load conditions. Therefore, 1-tile delay characterization does not yield accurate values of delays. Thus, to increase the accuracies of tile delays, three additional methods of characterizing tiles are proposed in Sec. 3.3.2, and Sec. 3.3.3.

3.3.1 1-Tile Delay Characterization (T1)

A single combination of input transition time and output load values is used for the 1-tile delay characterization. The delay of the tile $d_i^{(X)}$ is dependent on PVT parameters $\mathbf{p}$ and the tile type $\varphi(i)$. Moreover, for n_φ different tile types, each of the n_φ tile types makes $n_T = n_\varphi$ possible tile delays and tile delay sensitivities for a certain tile position i. Thus, the number of tile delay characterizations increases linearly, $O(N)$. In Figure 3.5,

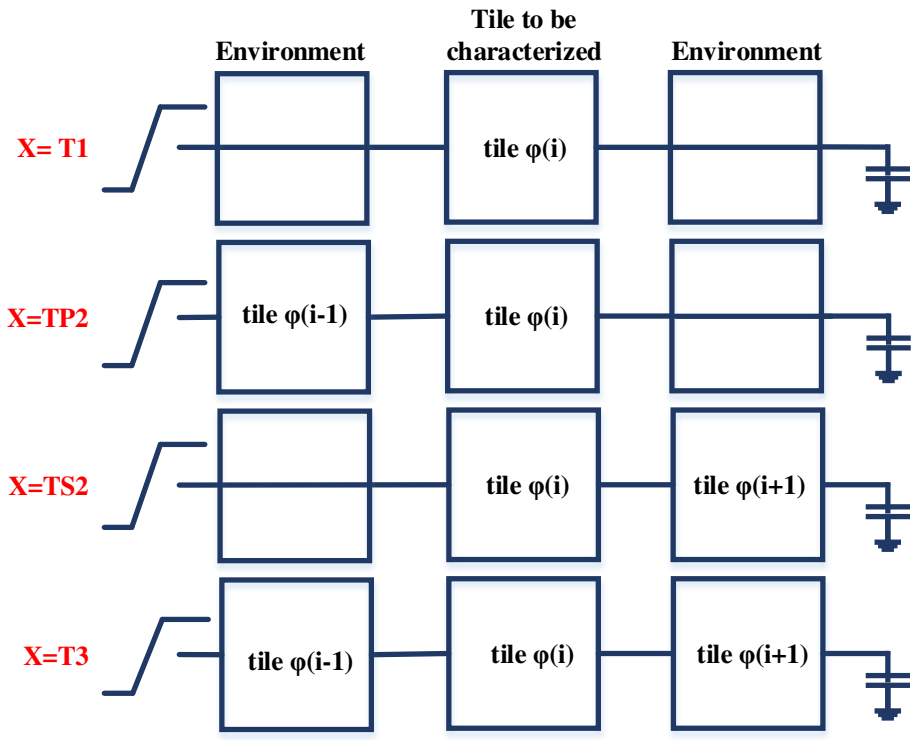


Figure 3.5: Tile delay characterization methods

$X = T1$ represents 1-tile delay characterization with constant values of input transition and output load.

$$d_i^{(T1)}(\mathbf{p}) = d_i^{(T1)}(\varphi(i), \mathbf{p}) \tag{3.9}$$

The combination of input transition time and output load are chosen based on analysis of tile delay behavior. Moreover, the base tiles selected for the synthesis of DDROs could have different drive strength and therefore different input pin capacitances. This could lead to unequal input transition and output load seen by various tiles in a DDRO. Additionally, using ideal constant input transition time and output load is not realistic. This implies that, the characterized tile delays will not reflect the real delays when concatenated to form DDROs. Therefore, more accurate methods of tile characterizations are necessary in order to have a better match of timing behavior of DDROs to their respective target critical paths.

3.3.2 2-Tile Delay Characterization (T2)

In a 2-tile characterization, either the influence of the input transition on the tile is considered by adding a preceding tile in the characterization (TP2), or the influence of

the output load on the the tile delay is considered by adding a succeeding tile to the characterization (TS2). In Figure 3.5, $X = TP2$ represents the 2-tile delay characterization with preceding tile and constant value of output load. The delay of the tile d_i^{TP2} is dependent on PVT parameters $\mathbf{p}$, preceding tile type $\varphi(i-1)$, and on the tile type $\varphi(i)$ in the DDRO path. In Figure 3.5, $X = TS2$ represents the 2-tile delay characterization with constant value of input transition time and a succeeding tile as output load. The delay of the tile d_i^{TS2} is dependent on PVT parameters $\mathbf{p}$, tile type $\varphi(i)$, and on the succeeding tile type $\varphi(i+1)$ in the DDRO path. Moreover, for n_φ different tile types, each of the n_φ tile types have n_φ successor types which makes $n_T = n_\varphi^2$ possible tile delays. Thus, the order of growth of tile delay characterizations increases quadratically, $O(N^2)$.

$$\begin{aligned} d_i^{(TP2)}(\mathbf{p}) &= d_i^{(TP2)}(\varphi(i-1), \varphi(i), \mathbf{p}) \\ d_i^{(TS2)}(\mathbf{p}) &= d_i^{(TS2)}(\varphi(i), \varphi(i+1), \mathbf{p}) \end{aligned} \tag{3.10}$$

In the cases where tile types seleced in the tile library have different drive strengths, the tiles in a DDRO would experience different output loads. This also leads to different input transition times at the beginning of each tile. Therefore, although the accuracy of 2-tile delay characterization is more than 1-tile delay characterization, 2-tile delay characterization has limitations due to idealistic output loads or input transition times.

3.3.3 3-Tile Delay Characterization (T3)

The 3-tile characterization considers the influence of both the input transition and the output load on the tile delay. In Figure 3.5, $(X = T3)$ represents the 3-tile delay characterization with preceding and succeeding tiles. The delay of the tile d_i^{T3} is dependent on PVT parameters $\mathbf{p}$, on the tile type $\varphi(i-1)$ that precedes the tile i in the DDRO path, tile type $\varphi(i)$, and on the tile type $\varphi(i+1)$ that succeeds tile i in the DDRO path. Therefore, for n_φ different tile types, the number of tile delay characterizations is $n_T = n_\varphi^3$.

$$d_i^{(T3)}(\mathbf{p}) = d_i^{(T3)}(\varphi(i-1), \varphi(i), \varphi(i+1), \mathbf{p}) \tag{3.11}$$

Due to formulation of more realistic input slopes and output loads from 1-tile to 3-tile delay characterization, there is an increase in accuracy of tile characterization and corresponding increase in the order of complexity. The order of complexity of tile characterization is linear, quadratic and cubic for 1-tile, 2-tile and 3-tile delay characterization

respectively. This results in not only massive increase in characterization time with 3-tile delay characterization, but also an increase in the size of optimization problem to synthesize DDROs. That is, the problem increases from linear to cubic from 1-tile to 3-tile delay characterizations. Therefore, Sec. 3.6 explains an iterative approach to solve large sized optimization problems.

3.4 Synthesis Using Optimization Solvers

The DDRO synthesis formulation explained in Sec. 3.2 is a combinatorial minimization problem with discrete variables. The traditional way to solve discrete variable problems is by integer programming [56]. In this thesis, the proposed methods to solve the objective of DDRO synthesis are by linear and by quadratic integer programming problems [56]. The traditional method is to use direct integer solvers such as (1) public domain solver GNU-LP [57] and (b) commercial solver CPLEX [58]. CPLEX can solve both quadratic and linear objective problems whereas GNU-LP can only solve linear objective problems. This section explains the matrix formulations to convert the DDRO objective in Equation (3.5) into matrix and vectors which can be fed into the quadratic and linear programming solvers. Sec. 3.4.1 demonstrates the matrix formulations for the objective of DDRO formulation. Sec. 3.4.2 formulates the DDRO objective into linear programming. Sec. 3.4.3 formulates the DDRO objective into quadratic programming.

3.4.1 Matrix Formulations for the Objective of Sensitivity-Based DDRO

In this section, the DDRO objective from Equation (3.5) is formulated into vector and matrix for critical path delay sensitivities and tile delay sensitivities, respectively. Moreover, the constraints used for the DDRO length, and the tile-tile relationship for 2-tile and 3-tile delay characterizations are also explained. The right hand side of Equation (3.5) consists of the delay sensitivities of critical paths which are written as a vector of delay sensitivities across PVT parameters multiplied by a factor of DDRO length n_t:

$$\mathbf{b} = \nabla d'_{CP}(\mathbf{p}_{PVT}) \cdot n_t \tag{3.12}$$

$\mathbf{b}$ is a column vector with dimensions $\mathbb{R}^{(n_p \cdot n_{PVT}) \times 1}$ where n_{PVT} is the number of PVT parameters for all corners and n_p is the number of PVT parameters considered for delay sensitivity matching at a certain corner. Tile delay sensitivities which are combined to match the sensitivities of the given critical path target are enumerated as a matrix for all positions of tile i and for all available tile types $\varphi(i)$ across n_{PVT} parameters.

The delay sensitivities of tile types $\varphi(i)$ can be computed either by 1-tile, 2-tile or 3-tile delay characterization. Thus, the enumeration of tile types for a certain position i can be defined as $\left\{\varphi_1^{(X)}(i), \varphi_2^{(X)}(i), \ldots, \varphi_{n_T}^{(X)}(i)\right\}$. Thereby, matrix $\mathbf{TM}_i^{(X)}$ is constructed to enumerate all tile types for a certain tile position i.

$$\mathbf{TM}_i^{(X)} = \left[\nabla d'^{(X)}_{\varphi_1^{(X)}(i)}(\mathbf{p}_{PVT}) \quad \nabla d'^{(X)}_{\varphi_2^{(X)}(i)}(\mathbf{p}_{PVT}) \quad \dots \quad \nabla d'^{(X)}_{\varphi_{n_T^{(X)}}(i)}(\mathbf{p}_{PVT}) \right] \tag{3.13}$$

Thus, $\mathbf{TM}_i^{(X)}$ gives the tile matrix for a single tile position i for all n_T possible tile delays, tile delay sensitivities and n_{PVT} parameters. The sensitivity-based DDRO synthesis consists of n_t tile positions. The matrix $\mathbf{TM}_i^{(X)}$ is further enumerated for all tile positions $i \in \{1 \cdots n_t\}$:

$$\mathbf{Z} = \left[\mathbf{TM}_1^{(X)} \quad \mathbf{TM}_2^{(X)} \quad \dots \quad \mathbf{TM}_i^{(X)} \quad \dots \quad \mathbf{TM}_{n_t}^{(X)} \right] \tag{3.14}$$

$\mathbf{Z}$ formulates the right hand side of the Equation (3.5) for the synthesis of sensitivity-based DDRO. The DDRO optimization with n_t tiles requires the consideration of the transition between subsequent tiles and depends on the four different methods of tile delay characterization. Figure 3.6 illustrates the dependency between two adjacent tiles $i-1$, i and $i+1$, for 2-tile and 3-tile delay characterization. In case of 2-tile delay characterization, the delay of tile i depends either on succeeding tile $i+1$ or preceding tile $i-1$. In case of a 3-tile delay characterization, the delay of tile i depends on the preceding tile $i-1$ and succeeding tile $i+1$.

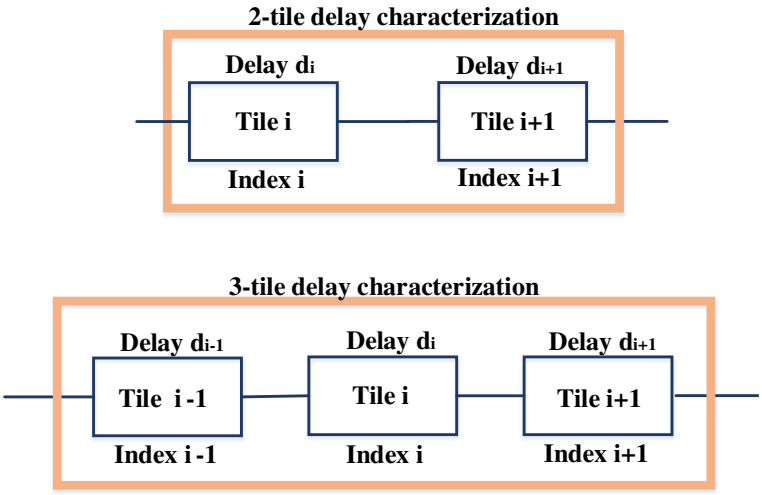


Figure 3.6: Dependency between adjacent tiles $i-1$, i and $i+1$ on delay characterization

In order to satisfy restriction based on tile-delay characterization methods, constraints are established which describe the relationship between two adjacent tiles for the cases of 2-tile and 3-tile delay characterizations.

$$C(i-1,i) = \{(\varphi_k, \varphi_l) \quad |\varphi_k \in \mathbf{TM}_{i-1}, \quad \varphi_l \in \mathbf{TM}_i\}, \\ |C(i-1,i)| = n_\varphi \tag{3.15}$$

3.4.2 Matrix Formulations for Linear Objective

The state-of-the-art method of the DDRO formulation from Equation (3.5) is $pn = 1$. This can be written in the form of integer linear optimization which is transformed to integer quadratic programming. In integer linear programming, the objective and the constraints are linear. Linear programming is computationally less intensive than quadratic programming [59]. Based on Equation (3.12) for the critical path delay sensitivities and Equation (3.14) for tile delay sensitivities across PVT parameters, the equations for linear programming can be formulated by the following derivation:

$$\min_{\mathbf{x}} \underbrace{\|\mathbf{b} - \mathbf{Z}\mathbf{x}\|_1}_{\mathbf{f}(\mathbf{x})=\mathbf{s}} \qquad s.t. \quad \sum_{i=1}^{n_T} x_i = n_t, \quad x_i \in \{0,1\} \tag{3.16}$$

The above minimization equation has a objective value vector $\mathbf{s}$ such as in [60]. Here, $\mathbf{x}$ is the unknown column vector and describes the tile type and position of each tile. For every tile position i, there are n_T possible choices of characterized tile delay sensitivities as shown in Equations (3.13) and (3.14) for sensitivity-based DDROs.

$$\|\mathbf{s}\|_1 = \sum_{i=1}^{n_T} |s_i| \tag{3.17}$$

Therefore, the objective function can be reformulated as:

$$\min \underbrace{\sum_{i=1}^{n_T} s_i}_{1^T \cdot \mathbf{s}} \quad s.t. \quad \underbrace{-s_i \leq f_i(\mathbf{x}) \leq s_i}_{\substack{\mathbf{b}-\mathbf{Z}\mathbf{x} \leq \mathbf{s} \\ \mathbf{b}-\mathbf{Z}\mathbf{x} \geq \mathbf{s}}}, \quad \sum_{i=1}^{n_T} x_i = n_t, \quad x_i \in \{0,1\} \tag{3.18}$$

3.4.3 Matrix Formulations for Quadratic Objective

The DDRO formulation from Equation (3.5) when $pn = 2$ can be written as integer least squares optimization, which is transformed to integer quadratic programming. In integer quadratic programming, the objective is quadratic and the constraints are linear. Based on Equation (3.12) for the critical path delay sensitivities and Equation (3.14) for tile delay sensitivities across PVT parameters, the quadratic optimization can be formulated as follows:

$$
\begin{aligned}
&\underset{\mathbf{x}}{min}\frac{1}{2}\left\|\mathbf{b}-\mathbf{Z}\mathbf{x}\right\|_2^2 \\
&min\left(\frac{1}{2}(\mathbf{b}-\mathbf{Z}\mathbf{x})^T(\mathbf{b}-\mathbf{Z}\mathbf{x})\right) \\
&min\left(\frac{1}{2}\mathbf{x}^T\mathbf{Z}^T\mathbf{Z}\mathbf{x}-\mathbf{Z}^T\mathbf{b}\mathbf{x}+\frac{1}{2}\mathbf{b}^T\mathbf{b}\right) \\
&min\left(\frac{1}{2}\mathbf{x}^T\mathbf{Z}^T\mathbf{Z}\mathbf{x}-\mathbf{Z}^T\mathbf{b}\mathbf{x}\right) \\
&s.t.\quad \sum_{i=1}^{n_t} x_i = n_t
\end{aligned}
\tag{3.19}
$$

Here, $\mathbf{x}$ is the unknown column vector and describes the tile type and position of the tile. In this thesis, DDROs are synthesized using both linear and quadratic programming. The synthesized DDROs are evaluated for the representation of the timing attributes of the target critical paths.

3.5 Delay-Tracking-Based DDRO

The state-of-the-art DDRO [22] explained in Sec. 3.2 is designed to match its delay sensitivities to a given target critical path according to Equation (3.5). In this thesis, a new synthesis method is proposed where the objective function matches the delays across PVT conditions, instead of delay sensitivities. This synthesis method is termed delay-tracking based DDRO. The measured frequency of the synthesized DDRO is then directly correlated to the delay of critical path. In this method, the delays of digital circuits can be simulated either by using SPICE [55] or by static timing analysis (STA) [61]. However, it would not be feasible to perform SPICE simulation for a complete design. Simulation of circuit delays with STA is computationally much less intensive than SPICE. Therefore, millions of critical paths on a design can only be characterized using STA at various PVT conditions in a feasible time. Furthermore, STA can also include effects of local variations without much increase in run-time for each cell type and therefore can easily incorporate worst case scenarios in contrast to SPICE simulation. To summarize, the goal of the delay-tracking based DDROs is to incorporate STA delays for tiles and critical paths such that DDRO synthesis is feasible for a large chip design.

The delays of tiles and critical paths are characterized at the n_{PVT} different PVT conditions $\mathbf{p}_{PVT_j}$ where $j = \{1, \ldots, n_{PVT}\}$.

$$
\mathbf{p}_{PVT} = \begin{bmatrix} \mathbf{p}_{PVT_1} \\ \mathbf{p}_{PVT_2} \\ \cdots \\ \mathbf{p}_{PVT_{n_{PVT}}} \end{bmatrix}
\tag{3.20}
$$

The resulting delays are collected in their respective vectors:

$$\begin{aligned}\mathbf{d}_{CP}(\mathbf{p}_{PVT}) &= \begin{bmatrix} d_{CP}(\mathbf{p}_{PVT_1}) \\ d_{CP}(\mathbf{p}_{PVT_2}) \\ \dots \\ d_{CP}(\mathbf{p}_{PVT_{n_{PVT}}}) \end{bmatrix} \\ \mathbf{d}_i^{(X)}(\mathbf{p}_{PVT}) &= \begin{bmatrix} d_i^{(X)}(\mathbf{p}_{PVT_1}) \\ d_i^{(X)}(\mathbf{p}_{PVT_2}) \\ \dots \\ d_i^{(X)}(\mathbf{p}_{PVT_{n_{PVT}}}) \end{bmatrix}\end{aligned} \tag{3.21}$$

From Equation (3.2), the DDRO delay $\mathbf{d}_{DDRO}^{(X)}(\mathbf{p}_{PVT})$ is obtained by summing up the delays of all its tiles. In analogy to Equation (3.5), the optimization goal of DDRO synthesis is to match the DDRO's delay behavior to critical path delay behavior across all n_{PVT} different PVT conditions, by selecting the number of tiles n_t and the tile types $\varphi(i), i = \{1, \dots, n_t\}$ for each tile position:

$$\underset{n_t, \varphi(i), i=\{1,\dots,n_t\}}{min} \left\| \mathbf{d}_{CP}^{'}(\mathbf{p}_{PVT}) - \mathbf{d}_{DDRO}^{'(X)}(\mathbf{p}_{PVT}) \right\|_{pn} \tag{3.22}$$

The superscript (X) stands for the method of characterizing the tile delays in a DDRO path explained in Sec. (3.3) and pn is the type of vector norm. Additionally, the length of the DDRO is decided by the number of the tiles selected for the DDRO path. The selected number of tiles is a variable in a user-defined range $n_t \in \{n_{t_{min}}, \dots, n_{t_{max}}\}$. Moreover, to equalize the delay values in the objective function the DDRO and critical path delays are normalized as:

$$\begin{aligned}\mathbf{d}_{DDRO}^{'(X)}(\mathbf{p}_{PVT}) &= \frac{\mathbf{d}_{DDRO}^{(X)}(\mathbf{p}_{PVT})}{\left\| \mathbf{d}_{DDRO}^{(X)}(\mathbf{p}_{PVT}) \right\|_2} \\ \mathbf{d}_{CP}^{'}(\mathbf{p}_{PVT}) &= \frac{\mathbf{d}_{CP}(\mathbf{p}_{PVT})}{\left\| \mathbf{d}_{CP}(\mathbf{p}_{PVT}) \right\|_2}\end{aligned} \tag{3.23}$$

3.6 DDRO synthesis using Heuristic Methods

The DDRO optimization problem is solved by choosing from a discrete set of tile types. Therefore, it can be formulated as a combinatorial integer programming problem [62]. The simplest method to solve this DDRO synthesis problem would be by enumerating all possible combinations of n_φ tile types for the length of the DDRO consisting of n_t tiles. Therefore, the number of possible combinations of n_φ tile types for n_t tiles is $n_\varphi^{n_t}$.

As an example consider $n_t = 15$ as the length of a DDRO and $n_\varphi = 6$. Here, the number of possible combinations is $\approx 510^{14}$, which cannot be enumerated.

The state-of-the-art method solves the DDRO synthesis problem by using direct solvers such as GNU-LP [22] or CPLEX [58]. Table 3.1 shows the average CPU run-time in seconds for DDRO synthesis using different DDRO synthesis methods with direct solvers. For the CPU run-time analysis, 96 DDROs are synthesized for different DDRO synthesis methods with (1) four different methods of tile characterizations, (2) quadratic integer programming, (3) varied number of tile types between 3 and 6, (4) two different optimization solvers namely GNU-LP and CPLEX optimization solvers, and (5) sensitivity-based DDROs are synthesized.

3.6.1 CPLEX Optimizer

For this analysis, DDRO synthesis using CPLEX is performed using a CPU with 24 parallel threads. The run-times may be affected based on the availability of actual CPU cores during the DDRO synthesis. Therefore, from Table 3.1 it is observed that for linear programming with 3 tile types, the CPU run-time using CPLEX increases from 1-tile (T1) to 2-tile delay (TP2, TS2) characterization for sensitivity based DDROs. However, from 2-tile delay (TP2, TS2) characterization to 3-tile delay (T3) characterization the CPU run-times almost remain constant. The CPU run-time for DDRO optimization with quadratic programming using 3 tile types increases exponentially from 1-tile to 3-tile delay characterization. The CPU run-time increases drastically from linear to quadratic programming for higher accuracy tile characterization methods using sensitivity based DDROs. In the case of 3-tile delay characterization, the average CPU run-time per DDRO increases from less than 1s to 1512s. For the synthesis of DDROs with 6 tile types, the CPU run-time for DDRO optimization is extremely large and not feasible for linear and quadratic programming for higher accuracy tile-delay characterization methods.

3.6.2 GNU-LP Optimizer

The GNU-LP optimizer used for DDRO synthesis is executed without parallel threads for linear programming. As GNU-LP solver is an integer programming solver, the run-times for quadratic programming using GNU-LP are not available. From Table 3.1 it can be seen that for sensitivity-based DDROs with 3 tile types in the tile library, the CPU run-time is feasible. The run-time using CPLEX increases from 1-tile (T1) delay to 3-tile (T3) delay characterization from 0.1s to 0.938s, respectively. Moreover, without using parallel threads, the run-time with GNU-LP is less than CPLEX for sensitivity based DDRO synthesis, 3 tile types for 1-tile (T1) and 2-tile (TP2, TS2) delay characterizations. Similar to CPLEX, for the synthesis of DDROs with 6 tile types, the CPU run-time for DDRO optimization is extremely large and not feasible for linear and quadratic programming with higher accuracy tile-delay characterization methods.

In summary, when there are 6 tiles chosen for the synthesis of DDROs, the CPU runtime for DDRO optimization using CPLEX and GNU-LP solvers are extremely large and not feasible for both linear and quadratic programming with accurate methods of 2-tile (TP2, TS2) or 3-tile (T3) delay characterizations.

Moreover, industrial standard cell libraries consist of hundreds of standard cells and using only 3 tile types would not be sufficient to represent the timing attributes of all the standard cells and thus the critical paths of a digital design. Therefore, it is necessary to increase the feasibility of the DDRO synthesis such that larger DDRO objective problems can be solved and the number of tile types selected for DDRO synthesis can be increased.

Table 3.1: DDRO synthesis with traditional solvers

Solver	Tile delay analysis	No. of Tiles	CPU time (s)
GNU-LP	T1	3	$<$ 1s
GNU-LP	TP2	3	$<$ 1s
GNU-LP	TS2	3	$<$ 1s
GNU-LP	T3	3	$<$ 1s
CPLEX	T1	3	$<$ 1s
CPLEX	TP2	3	17
CPLEX	TS2	3	43
CPLEX	T3	3	1512
GNU-LP	T1	6	$<$ 1s
GNU-LP	TP2	6	$>$ 2 weeks
GNU-LP	TS2	6	$>$ 2 weeks
GNU-LP	T3	6	$>$ 2 weeks
CPLEX	T1	6	$>$ 2 weeks
CPLEX	TP2	6	$>$ 2 weeks
CPLEX	TS2	6	$>$ 2 weeks
CPLEX	T3	6	$>$ 2 weeks

3.6.3 DDRO Partitioning and Enumeration Sets

In order to provide DDRO synthesis for larger sets of tile types and larger DDRO length, three new heuristic methods to synthesize DDROs are proposed in this thesis. The basic principle of heuristic methods is the enumeration of tile type combinations. To provide a feasible enumeration based synthesis of DDROs, a small number of $n_s < n_t$ tiles are enumerated. The hierarchical structure of a DDRO path is a concatenation of n_{tb} tile blocks, where a tile block is constructed from n_s tiles as shown in Figure 3.7. To generate a DDRO, a concept of successive optimization is used which enumerates n_s tiles at a time. After having decided the first block of n_s tiles in the first iteration, the second iteration decides the second block of n_s tiles, and so on till the last block $it = n_{tb}$ of n_s tiles is decided in the DDRO path.

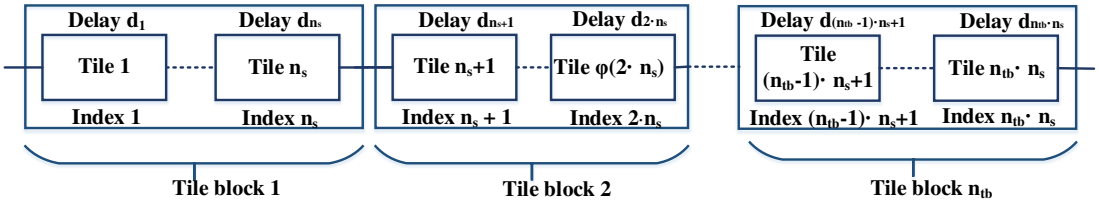


Figure 3.7: DDRO path structure with n_{tb} tile blocks (TBs), each tile block with n_s tiles

Successive optimization of selections of n_s tiles in each step requires individual consideration of the transition between subsequent tile blocks, depending on the four different methods of tile delay characterization. Figure 3.8 illustrates the dependency between two adjacent tile blocks at iterations it and $it+1$ for 2-tile and 3-tile delay characterizations. In case of a 2-tile delay characterization with succeeding tile, or, respectively, preceding tile, the delay of the last tile $it \cdot n_s$ of tile blocks it depends on the first tile $it \cdot n_s + 1$ of the succeeding tile block $it + 1$, or, respectively, vice versa. In case of a 3-tile delay characterization, the delays of the last two tiles $it \cdot n_s - 1$ and $it \cdot n_s$ of tile block it depend on the delay of the first tile $it \cdot n_s + 1$ of the succeeding tile block $it + 1$.

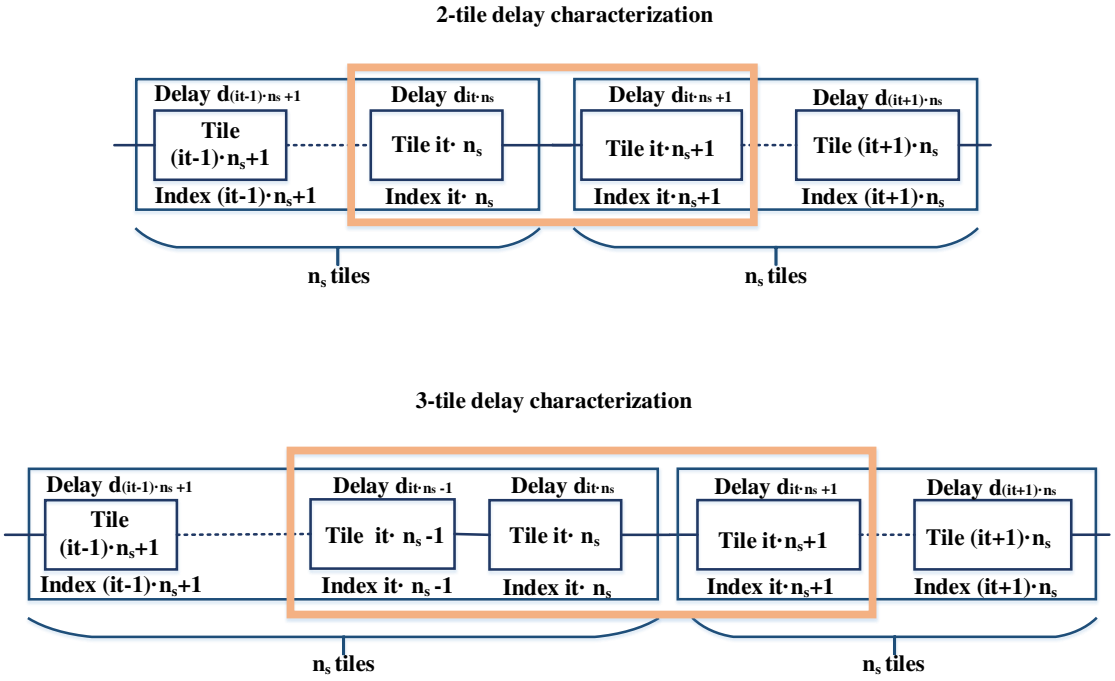


Figure 3.8: Dependency between adjacent tile blocks it and $it+1$ on delay characterization

The enumeration of combinations and the number of combinations of a tile block are different for the first tile block, $it = 1$, the last tile block, $it = n_{tb}$, and the tile blocks in between, $it = \{2, \ldots, n_{tb}\}$. The first tile block $it = 1$ includes the first tile of the

second tile block in its consideration. This multiplies the number of combinations by the number of tile types n_φ. The subsequent tile blocks $it = \{2, \ldots, n_{tb} - 1\}$, have their first tile type already determined by the preceding tile block, while including the first tile of the succeeding tile block $it + 1$ in their consideration. Additionally, the last tile block $it = n_{tb}$ has its first tile already decided. The resulting combinations of tile types in the tile blocks are collected in respective enumeration sets $S^{(X)}(it)$. The cardinality of $S^{(X)}(it)$ is given by:

$$\left|S^{(X)}(it)\right| = n_c^{(X)}(it) \tag{3.24}$$

Therefore, the number of possible tile combinations $n_c^{(X)}(it)$ for each of the cases is as follows:

$$\begin{aligned} n_c^{(X=1)} &= \quad n_\varphi^{n_s} \qquad it = \{1, \ldots, n_{tb} - 1\} \\ n_c^{(X=2,3)} &= \begin{cases} n_\varphi^{n_s+1}, & it = 1 \\ n_\varphi^{n_s}, & it = \{2, \ldots, n_{tb} - 1\} \\ n_\varphi^{n_s-1}, & it = n_{tb} \end{cases} \end{aligned} \tag{3.25}$$

The respective tile type enumeration sets are overlapping to a large extent and can be determined and stored efficiently by sharing. Figure (3.9) illustrates the overlapping. For each tile block element of the enumeration set $S^{(X)}(it)$, the corresponding tile block

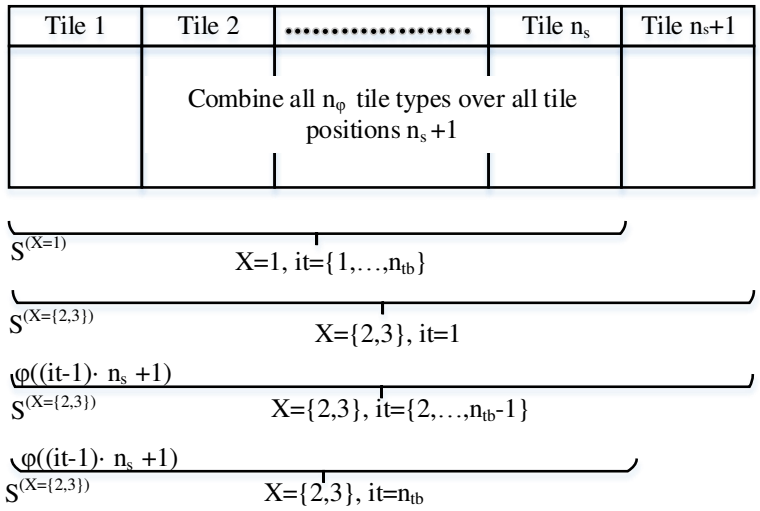


Figure 3.9: Illustration of enumeration sets $S^{(X)}(it)$ for different tile delay characterizations X and different iterations it in the successive enumeration process of heuristic algorithms

delay is computed by summing the tile delays of the respective tile block according to their tile type and delay characterization. This is done for all PVT conditions:

$$\mathbf{d}_{TB}^{(X)}(S^{(X)}(it), \mathbf{p}_{PVT}) = \sum_{i=1}^{n_s} \mathbf{d}_{i}^{(X)}(S^{(X)}(it), \mathbf{p}_{PVT})) \tag{3.26}$$

3.6.4 Iterative Computation of the DDRO Objective Function

At each iteration it, the DDRO path delay $\mathbf{d}_{DDRO}^{(X)}(it, \mathbf{p}_{PVT})$ is the sum of the delays of the already selected tile blocks until the previous iteration $it-1$ and the delay of tile block to be determined in iteration it:

$$\mathbf{d}_{DDRO}^{(X)}(it, \mathbf{p}_{PVT}) = \sum_{j=1}^{it} \mathbf{d}_{TB}^{(X)}(S^{(X)}(j), \mathbf{p}_{PVT}) \tag{3.27}$$

In this method, the objective in Equation (3.22) considers a partial DDRO length $n_s \cdot it$. Thus, the $\mathbf{d}_{DDRO}^{(X)}$ in objective function from Equation (3.22) is substituted by $\mathbf{d}_{DDRO}^{(X)}(it, \mathbf{p}_{PVT})$ and then the objective function is solved. Moreover, the objective at each iteration is computed until a given minimum length of DDRO $n_{t_{min}}$ is reached, after which in the next iteration, addition of more tile types is performed as long as the objective value reduces in comparison to the previous iteration. The DDRO synthesis process is continued until the maximum length of the DDRO $n_{t_{max}}$ is reached or at the point where the objective value no longer decreases compared to the previous iteration. The DDRO synthesis process is completed with the resulting number n_{tb} of tile blocks.

3.6.5 Multimodal-Successive Optimization

In the *Multimodal-Successive Optimization* approach, several solutions of tile block types are selected in each iteration. That is, a set of $\beta \in Z^+$ solutions with best objective values according to Equations (3.22) and (3.27) is selected. β is a user defined number. This approach enables to browse a larger enumeration space than the approach described before, while keeping the CPU run-time reasonable [63].

In iteration $it = 1$, the subset $A(1)$ of β tile block types from the $n_c^{(X)}(1)$ available alternatives of tile block types is $S^{(X)}(1, \mathbf{p}_{PVT})$ with the best objective value according to Equations (3.22) and (3.27) is determined:

$$A(1) \subseteq S^{(X)}(1, \mathbf{p}_{PVT}), \quad |A(1)| = \beta \tag{3.28}$$

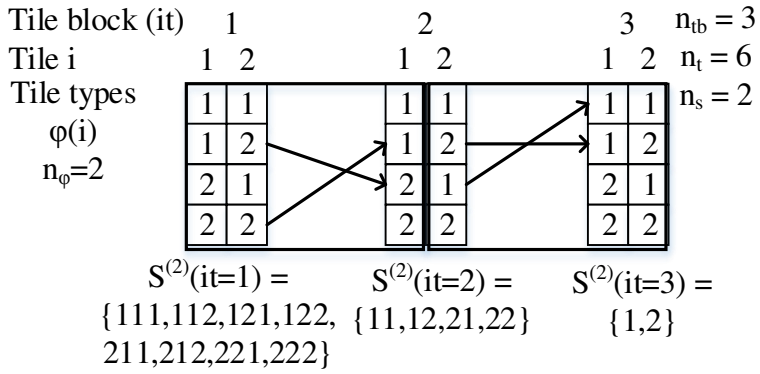
In each subsequent iteration it, a corresponding next subset $A(it)$ is computed from the $n_c^{(X)}(it)$ available alternatives of tile block types $S^{(X)}(it, \mathbf{p}_{PVT})$:

$$A(it) \subseteq S^{(X)}(it, \mathbf{p}_{PVT}), \quad |A(it)| = \beta, it = \{2, \ldots, n_{tb}\} \tag{3.29}$$

Additionally, a relation $T(it, it+1)$ is established during the Multimodal-Successive process that describes which tile block types follow each other in the respective DDRO variant:

$$\begin{aligned} T(it, it+1) = \{(k,l) \quad & |k \in A(it), l \in A(it+1)\}, \\ |T(it, it+1)| = \beta & \end{aligned} \tag{3.30}$$

The mapping T is a unique function, which results in over all β different DDROs at the end of the optimization process. These DDROs represent the β best DDROs to track the delay behavior of a critical path over all PVT conditions. Figure 3.10 illustrates an example of DDRO synthesis using Multimodal-Successive optimization method. The larger the chosen value of β, the larger is the solution space and the higher is the probability of finding the global minimum. However, the computational complexity increases with increasing β. The Multimodal-Successive Heuristic method is illustrated in Algorithm 1.



β	DDRO (as a sequence of tile types)
1	1 2 2 1 1 1
2	2 2 1 2 1 2

Figure 3.10: Example: Result of Multimodal-Successive optimization for DDRO with 6 tiles in $n_{tb} = 3$ tile blocks with 2 tiles each. There are $n_\varphi = 2$ tile types in the tile library. The numbering of $\beta = 2$ DDROs corresponds to the ordering of tracking ability

3.6.6 Single-Mode-Successive Optimization

The *Single-Mode-Successive Optimization* approach is also a Heuristic algorithm whose method and steps to synthesize DDROs is similar to that of the Multimodal-Successive optimization explained in Sec. 3.6.5 [63]. At first, all possible tile blocks of certain selected tile type and n_s tile length combination are enumerated which results in the

Algorithm 1 Multimodal-Successive Heuristic Optimization method

Require: $\varphi(i), \mathbf{d}_i^{(X)}(\mathbf{p}_{PVT}), i = \{1, \ldots, n_\varphi\}$,
$\mathbf{d}'_{CP}(\mathbf{p}_{PVT})$, $n_{t_{min}}$, $n_{t_{max}}$, n_s, β
Iteration: $it \leftarrow 1$
while $it <= n_{t_{max}}/n_s$ **do**
compute $S^{(X)}(it, \mathbf{p}_{PVT})$ from Figure 3.9
compute $\Delta d^{(X)}_{DDRO}(it, \mathbf{p}_{PVT}) = \min_{l \in \left\{1 \cdots n_c^{(X)}(it)\right\}} \Delta d^{(X)}_{DDRO_l}(it, \mathbf{p}_{PVT})$
with $\left\| \Delta \mathbf{d}'_{CP}(\mathbf{p}_{PVT}) - \mathbf{d}'^{(X)}_{DDRO_l}(it, \mathbf{p}_{PVT}) \right\|_{pn}$
from Equations (3.22) and (3.27)
compute $A(it)$ from Equations (3.28), (3.29)
compute $T(it, it+1)$ from Equation (3.30)
if $n_{t_{min}} < it \cdot n_s$ **then**
if $\Delta d^{(X)}_{DDRO}(it, \mathbf{p}_{PVT}) >= \Delta d^{(X)}_{DDRO}(it-1, \mathbf{p}_{PVT})$ **then**
compute $n_{tb} \leftarrow it$
break
end if
end if
compute $n_{tb} \leftarrow it$
compute $it \leftarrow it + 1$
end while
Return $A(it)$, $T(it+1, it), it \in \{1, \ldots, n_{tb}\}$

set of enumerated tile blocks at each iteration as given in Figure 3.9. The objective of the DDRO synthesis for the Single-Mode-Successive optimization is solved according to Equation 3.22. The set of available tile blocks for each iteration depends on the tile block selected at previous iteration and is given by the relation $T(it, it+1)$ as given in Equation 3.30. The main difference between Single-Mode-Successive and Multimodal-Successive optimization is that the number of solutions selected at each iteration β is set to one, i.e. only one tile block is selected out of all the enumeration sets at each iteration it. The tile block selected at each iteration is the one which resulted in least objective value $\Delta d^{(X)}_{DDRO}(it, \mathbf{p}_{PVT})$ among all the tile blocks in the available enumeration set $S^{(X)}(it, \mathbf{p}_{PVT})$. Therefore, at the end of n_{tb} iterations, there is only one DDRO solution obtained which is the final DDRO solution.

In this method, the CPU run-time to run the DDRO synthesis and memory consumption for the storage of all the solutions and generation of available enumeration sets of tile blocks $S^{(X)}(it, \mathbf{p}_{PVT})$ is much less than Multimodal-Successive optimization. Therefore, this method can handle larger DDRO synthesis problems with more tile types in the tile library set in comparison to the Multimodal-Successive optimization. However, due to local optimization with lower search range of DDRO solutions, the accuracies of DDRO delay-tracking ability maybe affected in comparison to Multimodal-Successive

optimization. Figure 3.11 shows an example of DDRO synthesis by Single-Mode Successive Optimization method. There are 2 tile types selected in the tile library set $n_\varphi = 2$ and the length of tile block $n_s = 2$. Therefore, there are 8, 4 and 2 possible combination of tile blocks at iterations $it = 1$, $it = 2$ and $it = 3$, respectively when number of tile blocks $n_{tb} = 3$ is selected for DDRO synthesis. As explained earlier for single-mode-successive optimization $\beta = 1$, therefore, the tile block resulting in the least objective value is selected at each iteration. Thus, a single set of tiles for a single DDRO is selected at the end of the DDRO synthesis.

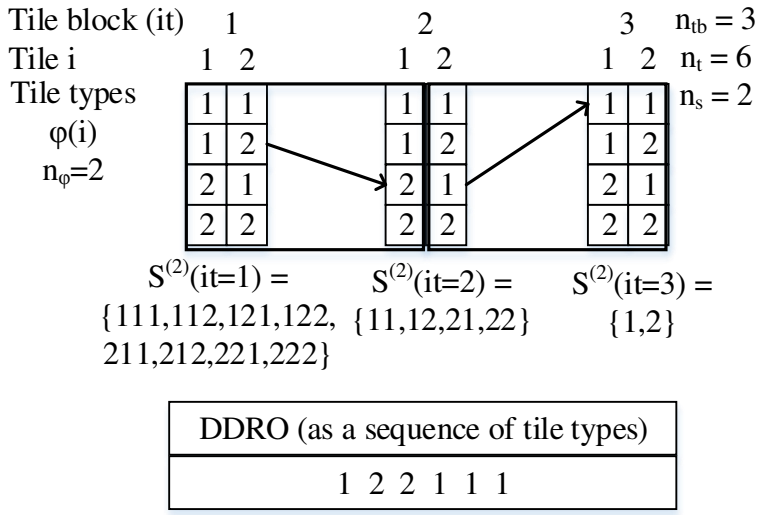


Figure 3.11: Exemplary result of Single-Mode Successive Optimization for DDRO with 6 tiles in $n_{tb} = 3$ tile blocks with 2 tiles each. There are $n_\varphi = 2$ tile types in the tile library

3.6.7 Single-Mode Back-Tracking Optimization

The third Heuristic method introduced in this thesis is the method of Single-Mode Back-Tracking Optimization [63]. Figure 3.12 shows the concept of the Single-Mode Back-Tracking Optimization method. In this method, the assumption is that the objective value of the DDRO synthesis starts with a larger value in the first iteration and gradually reduces at each iteration. Moreover, if the selected tile block does not result in a desired reduction of the objective value, the method eliminates the selected tile block to choose another tile block and back-tracks to the previous iterations until the objective value decreases as defined by the user. This approach uses less CPU memory in comparison to Multimodal-Successive Optimization explained in Sec. 3.6.5 due to the storage of only one tile block per iteration while keeping a broader search space in comparison to Single-mode-Successive Optimization explained in Sec. 3.6.6. However, due to the back-tracking the CPU run-time to synthesize the DDROs is considerably larger than both Multimodal-Successive Optimization and Singlemode-Successive Optimization.

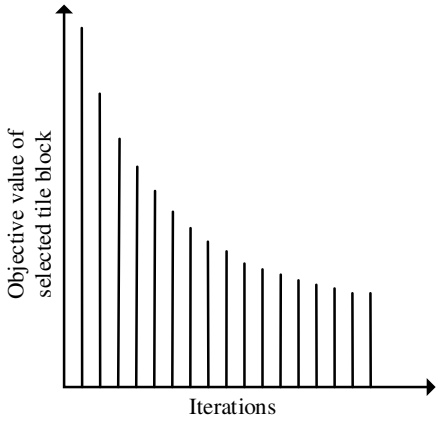


Figure 3.12: Concept of Single-Mode Back-Tracking Optimization Method

In the Single-Mode Back-Tracking Optimization method, only one solution is selected in each iteration. The tile block which results in the best objective according to Equations (3.22) and (3.27) is selected at each iteration. At iteration $it = 1$, the best objective value is chosen according to Equations (3.22) and (3.27) and is determined from the available $S^{(X)}(1, \mathbf{p}_{PVT})$ tile block type alternatives. This chosen tile block type is placed in a subset matrix $A(1)$ as per Equation 3.28.

Furthermore, the objective value $\Delta d^{(X)}_{DDRO}(it, \mathbf{p}_{PVT})$ calculated from Equation (3.27) for the selected tile block at iteration is stored. Additionally, all the tile block types selected in the subsequent iterations are also stored as per Equation 3.29.

The tile block types which follow each other in a DDRO variant is given by a relation $T(it + 1, it)$ from Equation 3.30. Furthermore, the objective value for iteration it is compared with the objective value of iteration $it - 1$:

$$\alpha(it) = \frac{\Delta d^{(X)}_{DDRO}(it, \mathbf{p}_{PVT})}{\Delta d^{(X)}_{DDRO}(it-1, \mathbf{p}_{PVT})} \tag{3.31}$$

where $\Delta d^{(X)}_{DDRO}(it, \mathbf{p}_{PVT})$ and $\Delta d^{(X)}_{DDRO}(it-1, \mathbf{p}_{PVT})$ are the objective values obtained with the selected tile block types at iteration it and $it - 1$, respectively.

If the value of $\alpha > \alpha_{max}$ where α_{max} is a user-defined maximum value of α, then the tile block types selected at iteration it and $it-1$ are deselected and the iterated number is moved back to $it - 1$. In this step, the previously selected tile block is eliminated from the choice of set of enumerated tile blocks and the tile block with the next best objective value is selected at iteration $it-1$. Moreover, for the newly selected tile block after the back tracking A(it-1), results in a new objective value $\Delta d^{(X)}_{DDRO}(it-1, \mathbf{p}_{PVT})$. This objective value is then compared with the objective value of the previous iteration $\Delta d^{(X)}_{DDRO}(it-2, \mathbf{p}_{PVT})$. Thus, the process of back tracking goes on until either $\alpha(it) <= alpha_{max}$ at each iteration or when the back tracking reaches the first iteration $it = 1$.

If $\alpha < \alpha_{max}$, the tile blocks are not eliminated and the DDRO synthesis process is continued to the next iterations.

Figure 3.13 shows an example of DDRO synthesis by Single-Mode Back-Tracking Optimization method. In this example, the number of tile blocks is fixed at $n_{tb} = 3$, the length of tile block $n_s = 2$ and number of tile types $n_\varphi = 2$. The user defined value for back-tracking $\alpha_{max} = 1.2$, where $\alpha(it)$ is calculated at each iteration according to Equation 3.31. In the first iteration, the tile block $\{2, 2\}$ results is the least objective value $\Delta d^{(X)}_{DDRO}(1, \mathbf{p}_{PVT}) = 0.5$ which is selected. Based on the tile block selected at the first iteration $\{2, 2\}$, the next tile block $\{1, 2\}$ is selected with the least objective value $\Delta d^{(X)}_{DDRO}(2, \mathbf{p}_{PVT}) = 0.7$. After which, the $\alpha(1) = 1.4$ is calculated which results in $\alpha(1) > \alpha_{max}$. Thus, first and second tile blocks selected are discarded from the result and also from the choice of tile block enumeration combinations. The iteration count is set to $it = 1$ and the tile block selection is restarted.

After the back-tracking to iteration $it = 1$ once again, the tile block $\{2, 1\}$ with the least objective value $\Delta d^{(X)}_{DDRO}(1, \mathbf{p}_{PVT}) = 0.6$ is selected at the iteration $it = 1$. Based on the tile block selected in the first iteration $\{2, 1\}$, the next tile block $\{1, 1\}$ is selected with the least objective value $\Delta d^{(X)}_{DDRO}(2, \mathbf{p}_{PVT}) = 0.5$. After which, the $\alpha(1) = 0.833$ is calculated which satisfies the condition $\alpha(1) < \alpha_{max}$. Therefore, the iteration count is proceeded to $it = 3$. At $it = 3$, based on the selection of the second tile block $\{1, 1\}$, the third tile block $\{2, 1\}$ is selected with the least objective value $\Delta d^{(X)}_{DDRO}(3, \mathbf{p}_{PVT}) = 0.4$. After which, the $\alpha(2) = 0.8$ is calculated which satisfies the condition $\alpha(2) < \alpha_{max}$. Thus, the DDRO is formed with the selection of tile blocks based on single-mode-back-track optimization.

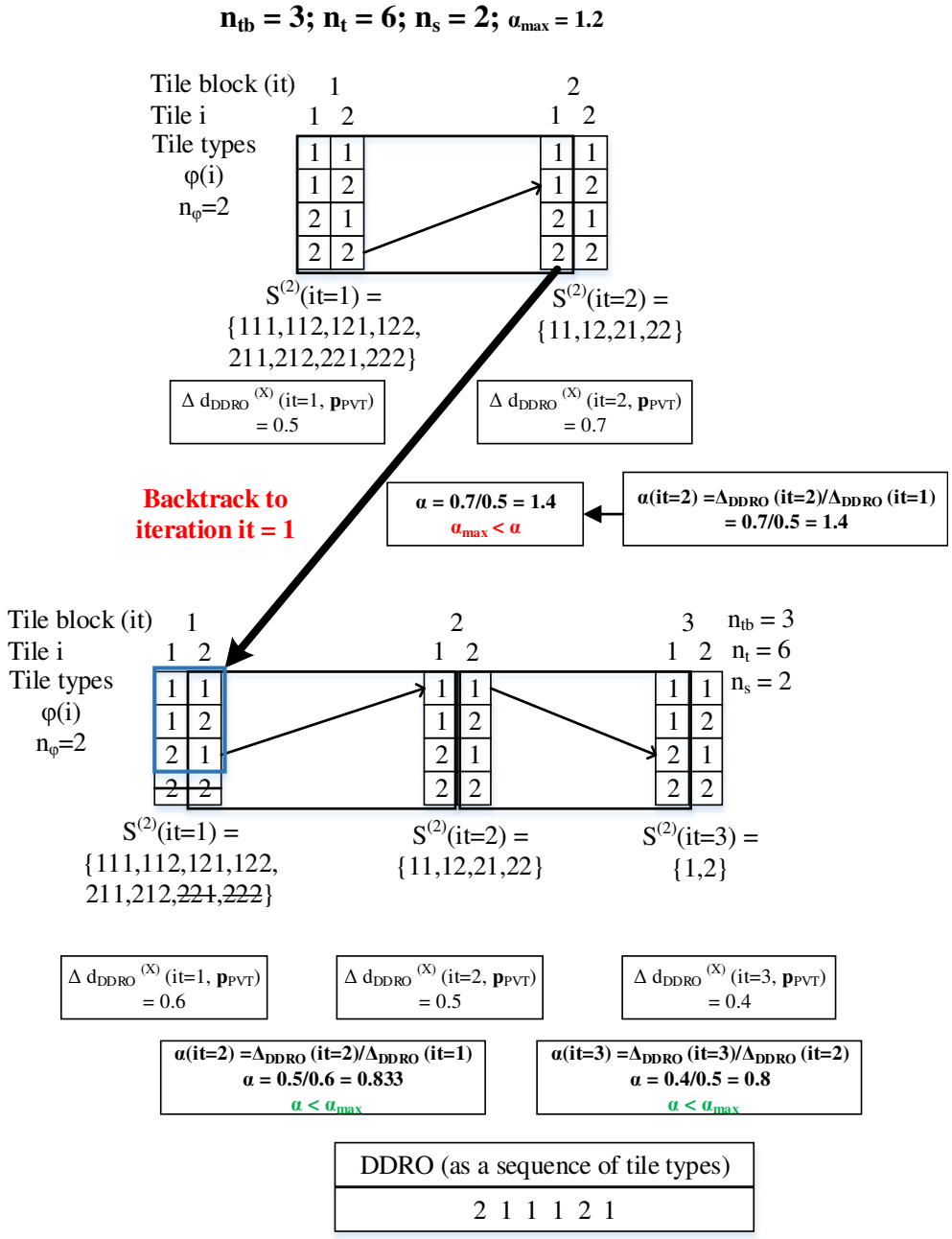


Figure 3.13: Exemplary result of Single-Mode Back-Tracking Optimization for DDRO with 6 tiles in $n_{tb} = 3$ tile blocks with 2 tiles each. There are $n_\varphi = 2$ tile types in the tile library. The user defined value $\alpha = 1.2$ is used to backtrack the DDRO synthesis to the previous iteration

Algorithm 2 Single-Mode Back-Tracking Heuristic Optimization method

Require: $\varphi(i), \mathbf{d}_i^{(X)}(\mathbf{p}_{PVT}), i = \{1, \ldots, n_\varphi\}$,
$\mathbf{d}_{CP}^{'}(\mathbf{p}_{PVT})$, $n_{t_{min}}$, $n_{t_{max}}$, n_s, α
Iteration: $it \leftarrow 1$
while $it <= n_{t_{max}}/n_s$ **do**
compute $S^{(X)}(it, \mathbf{p}_{PVT})$ from Figure 3.9
compute $\Delta d_{DDRO}^{(X)}(it, \mathbf{p}_{PVT}) = \underset{l \in \left\{1 \cdots n_c^{(X)}(it)\right\}}{min} \Delta d_{DDRO_l}^{(X)}(it, \mathbf{p}_{PVT})$
with $\left\| \Delta \mathbf{d}_{CP}^{'}(\mathbf{p}_{PVT}) - \mathbf{d}_{DDRO_l}^{'(X)}(it, \mathbf{p}_{PVT}) \right\|_{pn}$
from Equations (3.22) and (3.27)
compute $A(it)$ from Equation (3.29)
compute $T(it, it+1)$ from Equation (3.30)
compute α from Equation (3.31)
if $n_{t_{min}} < it \cdot n_s$ **then**
if $\Delta d_{DDRO}^{(X)}(it, \mathbf{p}_{PVT}) >= \Delta d_{DDRO}^{(X)}(it-1, \mathbf{p}_{PVT})$ **then**
compute $n_{tb} \leftarrow it$
break
end if
else if $\alpha > \alpha_{max}$ **then**
set $S^{(X)}(it-1, \mathbf{p}_{PVT}) \leftarrow S^{(X)}(it-1, \mathbf{p}_{PVT}) - A(it-1)$
set $A(it), A(it-1) = \mathbf{0}$
compute $it \leftarrow it - 1$
end if
compute $n_{tb} \leftarrow it$
compute $it \leftarrow it + 1$
end while
Return $A(it)$, $T(it+1, it), it \in \{1, \ldots, n_{tb}\}$

3.7 Evaluation of DDROs

In order to qualify the synthesized DDROs and compare different DDRO synthesis methods, evaluation of synthesized DDROs is necessary. Figure 3.14 shows the methods to evaluate synthesized DDROs. At first DDROs are synthesized and the netlist for the synthesized DDROs are generated. Afterwards, the synthesized DDROs are characterized for their timing attributes across given PVT conditions. Here, two different methods are used to evaluate the synthesized DDROs and are explained as follows:

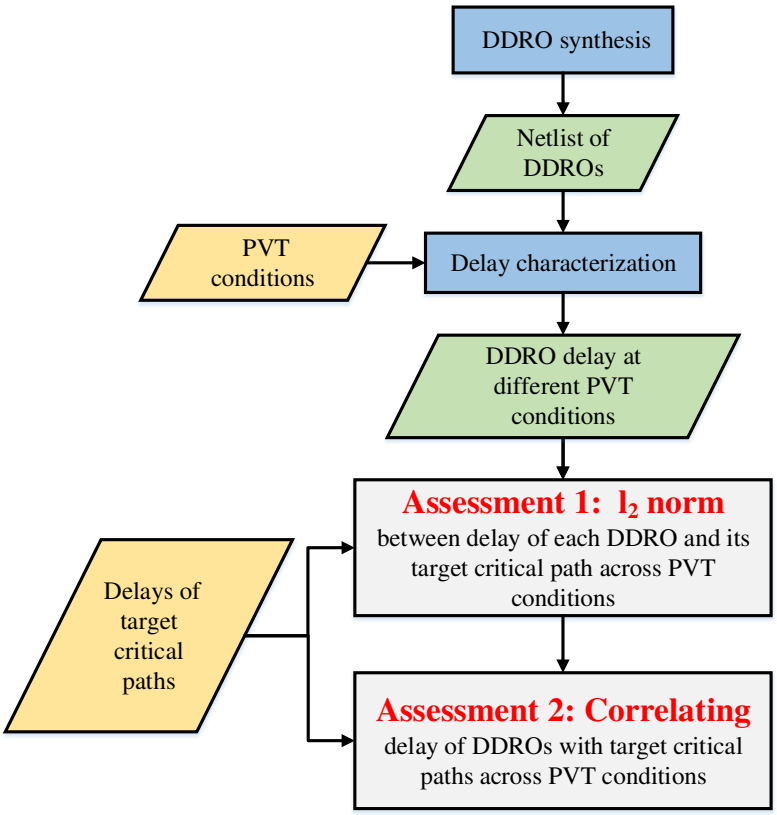


Figure 3.14: Method to evaluate the quality of synthesized DDROs

3.7.1 Assessment Method 1

In Equation (3.23), $\mathbf{d}'_{CP}(\mathbf{p}_{PVT})$ is the normalized delay of critical paths across PVT parameters and $\mathbf{d}_i'^{(X)}(\mathbf{p}_{PVT})$ is the normalized delay of tiles. The normalized delay of DDRO from Equation (3.2) is given as:

$$\mathbf{d}'^{(X)}_{DDRO}(\mathbf{p}_{PVT}) = \sum_{i=1}^{n_t} \mathbf{d}_i'^{(X)}(\mathbf{p}_{PVT}) \tag{3.32}$$

The quality of DDROs is evaluated by calculating the l_2 norm between the synthesized DDROs and their respective critical paths across PVT conditions as shown in Equation (3.33).

$$der^{'} = \left\| d^{'}_{CP}(\mathbf{p}_{PVT}) - d^{'}_{DDRO}(\mathbf{p}_{PVT}) \right\|_2 \tag{3.33}$$

3.7.2 Assessment Method 2

To analyze if the synthesized DDROs can track the delays of critical paths across PVT conditions, both DDROs and critical paths are normalized by dividing their delays at all PVT conditions by the delay at a user-defined PVT corner:

$$d_Y(\mathbf{p}_{PVT_{us}}) \in d_Y(\mathbf{p}_{PVT_i}), \quad i \in \{1, \ldots, n_{PVT}\} \tag{3.34}$$

where d_Y represents the delay of the DDRO or the critical path. The delay values of DDROs and critical paths are normalized as follows:

$$\mathbf{d}^{'}_Y(\mathbf{p}_{PVT}) = \frac{\mathbf{d}_Y(\mathbf{p}_{PVT})}{d_Y(\mathbf{p}_{PVT_{us}})} \tag{3.35}$$

The delay error in percentage $\Delta\mathbf{d}_{PE}(\mathbf{p}_{PVT})$ is calculated for normalized delays from Equations (3.34) and (3.35) between critical path and the respective DDRO using (3.36)

$$\Delta\mathbf{d}_{PE}(\mathbf{p}_{PVT}) = \begin{bmatrix} 1 - \frac{d^{'}_{CP}(\mathbf{p}_{PVT_1})}{d^{'(X)}_{DDRO}(\mathbf{p}_{PVT_1})} \\ 1 - \frac{d^{'}_{CP}(\mathbf{p}_{PVT_2})}{d^{'(X)}_{DDRO}(\mathbf{p}_{PVT_2})} \\ \cdots \\ 1 - \frac{d^{'}_{CP}(\mathbf{p}_{PVT_{n_{PVT}}})}{d^{'(X)}_{DDRO}(\mathbf{p}_{PVT_{n_{PVT}}})} \end{bmatrix} \tag{3.36}$$

3.8 Summary

This Chapter introduces the theory and formulations of the various DDRO synthesis methods developed in this thesis. At first, the fundamental goal of a DDRO to mimic the delay behavior of a group of critical paths is explained. Moreover, the structure of a DDRO constructed by concatenating tiles and the methodology for the DDRO synthesis is explained. Furthermore, the formulations of the existing state-of-the-art method of synthesizing DDROs by matching the delay sensitivities w.r.t. PVT parameters to

that of the target critical path group is explained. In addition to linear programming introduced in the state-of-the-art method [22], quadratic programming is proposed in this thesis in order to further increase the accuracies of DDROs.

Tiles are built by concatenating standard cells of equal type and are used to reduce the impact of input slope and output load on the timing attributes of a characterized tile such that the timing attributes can be linearly concatenated to construct a DDRO. However, the influence of input slope and output load is not fully eliminated by using a tile characterized by the state-of-the-art method. Therefore, three new methods of tile delay characterizations are proposed out of which 3-tile delay characterization has additional tiles as an input slope tile and output load to the tile to be characterized. This results in the most accurate but complex method of tile delay characterization.

Additionally, in order to eliminate the inconsistencies of delay sensitivity computation and to enable the usage of STA data for DDRO synthesis, a new DDRO formulation is proposed and formulated by means of delay-tracking of target critical paths across PVT conditions.

Furthermore, to enable larger DDRO synthesis problems to be solved in feasible CPU run-time and memory usage, three new Heuristic iterative algorithms are proposed namely Single-Mode Successive Optimization, Multimodal-Successive Optimization and Single-Mode Back-Tracking Optimization. Among these algorithms Single-Mode Successive Optimization is the simplest algorithm which selects one solution per iteration and requires least computational effort. However, this method does not span a wide area to search for the optimum DDRO solutions and therefore may not yield in the best matching DDROs. The Multimodal-Successive Optimization selects multiple solutions in each iteration and therefore spans a wider space than Single-Mode Successive Optimization and result in better matching DDROs. However, this method requires high CPU memory to store the multiple selected solutions in each iteration and the data following these solutions. The Single-Mode Back-Tracking Optimization method works on the concept of eliminating a selected solution at an iteration and back-tracking to the previous iteration if the objective value in the current iteration does not reduce by a user-defined value compared to the previous iteration. Thus, this method requires less memory compared to Multimodal-Successive Optimization but due to the back-tracking consumes large CPU-runtimes.

Finally, in order to evaluate the synthesized DDROs, DDRO assessment methods are proposed in this thesis, which compare the delays of the synthesized DDRO to the critical path delays.

4 Experimental Results of DDROs

The goal of design representative timing monitors as explained in Chapter 3 is to mimic the timing behavior of critical paths of a digital design across PVT conditions. In this thesis, different concepts and formulations to achieve the goal of developing more accurate design representative timing monitors have been analyzed. With that, sensitivity-based and delay-tracking-based DDRO synthesis methods illustrated in Sec. 3.2 and Sec. 3.5, respectively are investigated. Furthermore, Sec. 3.4 demonstrates the implementation method to synthesize DDROs using direct solvers such as LP solve and CPLEX. In this work, new heuristic methods are described in Sec. 3.6 which are able to handle larger DDRO synthesis problems in comparison to using direct solvers. This Chapter demonstrates various experimental results to analyze and evaluate the following: (1) sensitivity-based DDRO synthesis, (2) synthesis of DDROs using three different methods of tile characterization, (3) synthesis of DDROs using direct solver (CPLEX), (4) synthesis of DDROs using extracted data from STA, (5) synthesis of DDROs with different lengths of tile blocks, (6) synthesis of DDROs with different lengths of tiles, (7) synthesis of DDROs using three different heuristic methods, (8) quality of delay matching of DDROs versus Generic ring oscillators (GROs), and (9) the impact of selected tile types to synthesize DDROs by synthesizing DDROs for 5 different tile library sets

4.1 Experimental Setup for DDRO Synthesis

This section explains the experimental setup used to synthesize DDROs. Moreover, this section explains the different types of critical paths used to evaluate the delay-tracking quality of synthesized DDROs. Additionally, the different combinations of DDRO synthesis formulations based on tile-delay characterization and integer programming methods are explained.

In this work, the heuristic methods explained in Sec. 3.6 for DDRO synthesis are implemented in R. Here, DDROs are synthesized using an Intel Xeon9 processor with 1 thread which has a 64bit Linux architecture and 250GB memory. On the other hand, the DDROs synthesized using direct solver CPLEX use 24 threads. Moreover, it should be noted that, the CPU run-time for DDRO synthesis is dependent on the availability of actual CPU cores.

For the analysis of different DDRO synthesis methods, the experiment utilizes 96 critical paths of an industrial technology below 40nm. The DDRO synthesis methods are evaluated for a given set of critical paths. Therefore, 96 DDROs are constructed using SPICE based data for each of the DDRO synthesis methods. Additionally, to evaluate different DDRO formulations methods, the following critical path types are used: (a)

critical paths having homogenous and heterogeneous mixture of cells of various cell types, (b) critical paths having cell combinations of different drive strengths, (c) critical paths of different lengths of cells, (d) critical paths with various fan-outs and (e) critical paths with combinations of weak and strong cell drive strengths.

The following DDRO synthesis algorithms are assessed in Subsections 4.2 and 4.3 to evaluate different tile delay characterization methods and DDRO formulations: LT1, LTP2, LTS2, LT3 with linear objective, QT1, QTP2, QTS2, QT3 with quadratic objective and the 4 respective tile delay characterizations.

The parameters for the DDRO synthesis in each experiment is given in a table under each section of this Chapter. The experimental parameters table typically consists of: (1) DDRO objective(s) used which is either linear or quadratic optimization, (2) the method(s) of tile delay characterization, (3) tile types selected, (4) number of drive strengths of tile types used for DDRO synthesis, (5) length of a tile, i.e. the number of standard cells concatenated to construct a tile, (6) length of a tile block if heuristic optimization is used, (7) solver used, (8) number of PVT parameters considered for DDRO synthesis, and (9) method by which delay data is obtained i.e. SPICE or STA. Additionally, in this work for Multimodal-Successive optimization, the number of solutions selected is set to $\beta = 20$.

4.2 Sensitivity-Based DDRO

Sensitivity-based DDRO synthesis aims to track the timing attributes of target critical paths by matching delay sensitivities of the synthesized DDROs to the given critical paths. In this section, the synthesized sensitivity-based DDROs are evaluated with respect to their ability to mimic timing attributes of critical paths. Here, the critical paths are characterized for delay sensitivities w.r.t. globally varying statistically independent process parameters and voltage. Delay sensitivity w.r.t. temperature is not considered due to its low impact, but is included in the method. In this approach, sensitivity is computed by varying the process parameters by $\pm 0.1\sigma$ and the voltage by $\pm 2\%$. The process is considered in the nominal case and voltage and temperature are varied across nominal, slow, fast and inverted temperature effect (ITE) corners.

4.2.1 DDRO Synthesis using Direct Solver

Results of DDRO synthesis using the state-of-the-art method of synthesizing DDROs [22] that uses direct solvers such as CPLEX is explained.

The parameters used for the sensitivity-based DDROs using CPLEX is given in Table 4.1. Table 4.2 gives the statistical results of the l_2 norm error of DDRO defined by Equation (3.33) for 96 different critical paths. In this experiment, it can be observed that the quadratic and linear optimization have similar results with respect to the statistical quantities. Secondly, it can be seen that in the current technology node and with 3 tile types in the tile library, addition of load tile in the 2-tile delay characterization yields a mean error of around 13 for both quadratic and linear optimization (LTS2, QTS2), respectively. Moreover, the l_2 norm error does not show any improvement as

Table 4.1: Experimental parameters for the sensitivity-based DDRO using CPLEX

Parameter	Value(s)
DDRO objectives	Linear and Quadratic
Tile delay characterization	T1, TP2, TS2, T3
Tile types	INV, NAND2, NOR2
Number of drive strengths of tile types	1
Number of tile types n_φ	3
Tile length	3
Tile block length n_s	Not-applicable since the CPLEX does not need the generation of enumerated set of tile blocks
Solver	CPLEX
Number of PVT corners	4
Delay data	SPICE simulation

compared to 1-tile delay characterization (QT1, LT1) which results in a mean error of around 13 and 12.7, respectively. LT1 is the DDRO synthesis method used in [22]. However, the addition of slope tile (LTP2, QTP2) results in significant improvement of DDRO sensitivity matching which results in a mean error of around 6. The 3-tile delay characterizations (LT3, QT3) yield the best matching DDROs for their respective critical paths with a mean error of around 4.

Table 4.2: Statistical evaluation of normalized error of delays for sensitivity based DDROs using direct solver CPLEX

Method	Mean [arb. unit]	Min [arb. unit]	Max [arb. unit]	Standard deviation [arb. unit]
LT1	13.03	0.64	26.90	6.10
LTP2	9.53	1.34	18.40	4.20
LTS2	12.86	1.01	28.09	6.09
LT3	8.19	0.97	17.13	4.29
QT1	12.67	1.01	27.04	5.94
QTP2	9.62	1.34	18.40	4.14
QTS2	12.79	1.01	28.13	6.04
QT3	8.02	0.97	15.82	4.01

Figure 4.1 shows the correlation between the delay of synthesized DDRO versus the delay of critical path at 4 different PVT conditions. The delays are normalized as given in Equation (3.35) since the DDROs are designed to track the delay behavior of critical path and not the absolute value of delay. Additionally, the maximum and mean error of delay in % is calculated according to Equation (3.36). The different colors in Figure 4.1 indicate different PVT points at which the delays are characterized. A 45 deg solid line represents the ideal regression line plotted as a reference to observe the tracking of

delay for different DDRO synthesis methods. The dotted line gives the actual regression line of each DDRO synthesis methods.

From Figure 4.1, it can be observed that for both linear and quadratic optimization, the dotted line deviates from the solid line for 1-tile (LT1, QT1) and 2-tile delay characterizations with succeeding tile (LTS2, QTS2) whereas the solid line and dotted line overlay each other for 2-tile delay characterizations with preceding tile (LTP2, QTP2) and 3-tile delay characterization (LT3, QT3). The mean error reduces from around 1.5 to 0.65 from 1-tile to 3-tile delay characterization for both linear and quadratic programming, respectively. Similarly, the maximum error reduces from around 6.7 to around 4.3 from 1-tile to 3-tile delay characterization, respectively. The maximum and mean error of delay is smallest with the QT3 method.

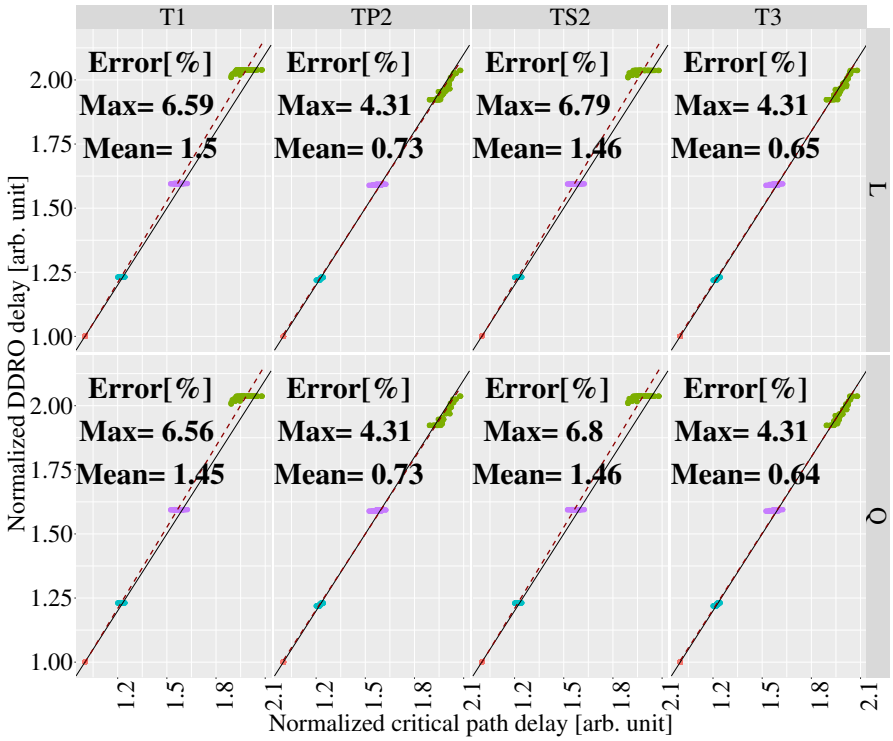


Figure 4.1: Correlation between synthesized DDROs delays and critical path delays across PVT corners for DDROs synthesized using direct solver CPLEX

Table 4.3 gives the average CPLEX CPU run-time in seconds for 24 threads to optimize 96 different DDROs for different DDRO synthesis methods. It can be observed that linear optimization requires lower run-time than quadratic optimization. Additionally, the CPU run-time for linear optimization increases linearly from 0.22 seconds for 1-tile to 0.53 seconds for 2-tile delay characterization and remains almost constant from 2-tile to 3-tile delay characterization. The CPU run-time for DDRO synthesis with quadratic optimization increases exponentially from 0.1 seconds to 1513 seconds for 1-tile to 3-tile delay characterization, respectively.

Table 4.3: Average CPU run-time for DDRO synthesis for sensitivity-based DDRO using direct solver CPLEX

Method	CPU run-time [s]
LT1	0.22
LTP2	0.57
LTS2	0.53
LT3	0.56
QT1	0.10
QTP2	17.63
QTS2	43.82
QT3	1512.78

4.2.2 DDRO Synthesis using Multimodal-Successive Optimization Method

In this section, DDROs are synthesized using a Heuristic method instead of using direct solver. The heuristic method used is the Multimodal-Successive optimization method as explained in Sec. 3.6.5.

Table 4.4: Experimental parameters for the sensitivity-based DDRO using heuristic method

Parameter	Value(s)
DDRO objectives	Linear and Quadratic
Tile delay characterization	T1, TP2, TS2, T3
Tile types	INV, NAND2, NOR2
Number of drive strengths of tile types	2
Number of tile types n_φ	6
Tile length	3
Tile block length n_s	5
Solver	Multimodal-Successive Optimization Method
Number of PVT corners	4
Delay data	SPICE simulation

The parameters used for the sensitivity-based DDROs using Multimodal-Successive optimization method is given in Table 4.4. Table 4.5 shows the overall l_2 norm error between DDROs and their respective critical path delays. It can be observed that for linear optimization from 1-tile (T1) to 3-tile (T3) delay characterization, the mean error reduces from 10% to 8.7%. For 2-tile (TP2, TS2) and 3-tile (T3) delay characterization the quadratic optimization has better results than linear optimization. The reason is, higher accuracies of tile delay characterizations, such as 2-tile and 3-tile delay characterization, yield better results than 1-tile delay characterization. However, it is important to note that there are inaccuracies observed in the results of sensitivity-based DDRO

where the 1-tile delay characterization results in lower mean and maximum error in comparison to 2-tile delay characterization (LTP2) with linear optimization.

Table 4.5: Statistical evaluation of normalized error of delays for sensitivity based DDROs using heuristic solver with Multimodal-Successive optimization methods

Method	Mean [arb. unit]	Min [arb. unit]	Max [arb. unit]	Standard deviation [arb. unit]
LT1	9.57	0.61	20.02	4.28
LTP2	10.77	0.46	24.65	5.85
LTS2	9.13	0.54	24.00	5.61
LT3	8.70	0.62	19.99	4.69
QT1	15.18	0.68	32.20	7.40
QTP2	6.44	1.04	13.35	3.42
QTS2	8.80	0.54	24.01	5.80
QT3	5.07	0.39	12.30	2.95

Figure 4.2 shows the correlation of normalized DDRO delay versus critical path delay across PVT corners for sensitivity-based DDRO synthesis. Figure 4.2 is similar to Figure 4.1 and the description of the figure is given in Sec. 4.2.1. From Figure 4.2 it can be observed that for both linear and quadratic optimization the dotted line deviates from the solid line for 1-tile (LT1, QT1) whereas the solid line and dotted line more or less coincide for all the other cases. Additionally, the maximum and mean error of delay in % is calculated according to Equation (3.36) where the maximum and mean error of delay is smallest with the QT3 method. The higher accuracy of tile delay characterization together with quadratic programming (QT3) reduces the maximum error by around 50% compared to the DDRO synthesis method in [22]. To summarize, based on the analysis of sensitivity-based DDROs, quadratic optimization (Q) is a better choice in comparison to linear optimization (L).

Table 4.6 gives the average CPU run-time to synthesize DDROs using the Multimodal-Successive optimization method. The first column describes the different DDRO synthesis methods. The second column describes the CPU run-time to generate the enumeration sets $S^{(X)}(it)$. Run-times to generate the enumeration sets is nearly the same for both linear and quadratic optimization. The third column gives the CPU run-time to execute the heuristic synthesis. It can be observed that on an average, the quadratic optimization takes less CPU run-time than the linear optimization. Moreover, the CPU run-time increases with an increase in the tile delay characterization accuracy for the generation of enumeration sets as well as for executing the heuristic synthesis.

4.2.3 Comparison of Sensitivity-based DDROs between Direct Solver (CPLEX) and Multimodal-Successive Optimization Method

Due to the CPU run-time constraints in the usage of CPLEX solver for the DDRO synthesis, only 3 tile types are used in contrast to the DDRO synthesis using Multimodal-

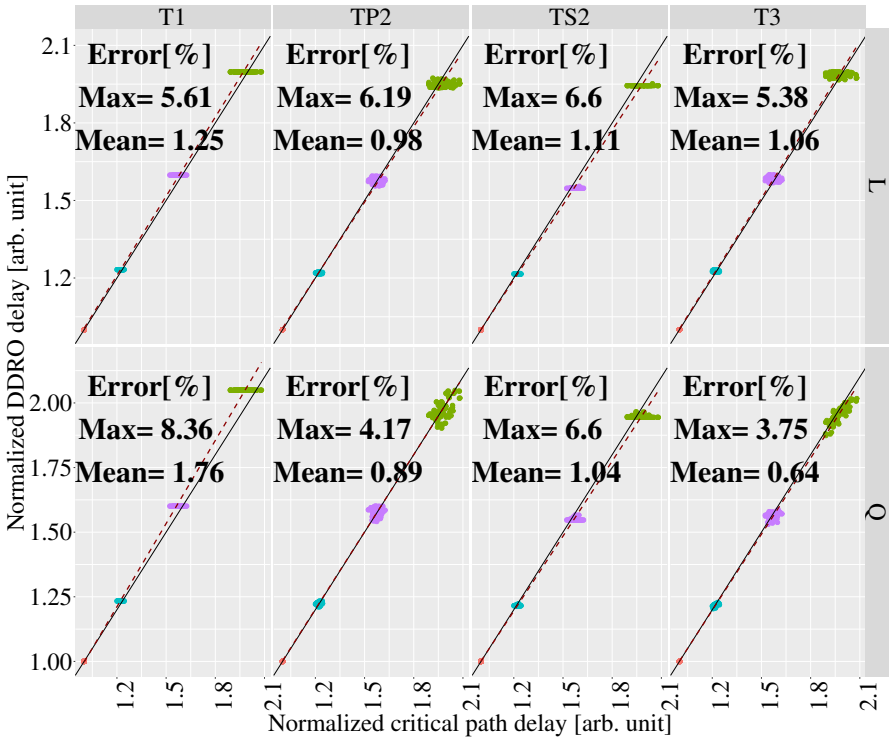


Figure 4.2: Correlation between synthesized DDROs delays and critical path delays across PVT corners for DDRO synthesized using heuristic solver with Multimodal-Successive optimization methods

Table 4.6: Average CPU run-time for DDRO synthesis for sensitivity-based DDRO using heuristic solver Multimodal-Successive optimization methods

Method	Generation of $S^{(X)}(it)$ [s]	Heuristic Synthesis [s]
LT1	1.56	64.05
LTP2	2.31	58.18
LTS2	5.8	32.25
LT3	6.4	78.81
QT1	1.6	38.26
QTP2	2.38	20.76
QTS2	2.64	22.23
QT3	6.82	40.24

Successive optimization which uses 6 tile types in the tile library set. All the other parameters w.r.t. critical paths and PVT conditions are the same for both solvers used to synthesize DDROs. From Tables 4.2 and 4.2, it can be seen that for the state-of-the-art DDRO synthesis method in [22] (LT1) using Multimodal-Successive method results

in the error reduction of around 25% in comparison to direct solver CPLEX. For the most accurate method of DDRO synthesis, which is the quadratic optimization with 3-tile delay characterization, there is a reduction in error of around 38% from the usage of CPLEX to Multimodal-Successive method.

Similarly, from Figures 4.1 and 4.2 it can be seen that, the linear optimization with 1-tile delay characterization results in a lower mean and lower max error with Multimodal-Successive optimization in comparison to CPLEX. However, 1-tile delay characterization with quadratic programming (QT1), 2-tile delay characterization with preceding tile and 3-tile delay characterization with linear programming (LTP2, LT3) result in higher mean and max error with Multimodal-Successive optimization in comparison to CPLEX. The best matching DDROs are from quadratic optimization with 3-tile delay characterization using the Multimodal-Successive optimization method, resulting in around, 12% decrease in maximum error in comparison to CPLEX solver.

As mentioned earlier, Tables 4.3 and 4.5 gives the CPU run-time for the CPLEX and Multimodal-Successive optimization method, respectively. Although for linear programming (LT1) the CPU run-time with CPLEX is much lower than Multimodal-Successive optimization, the CPU run-time for the quadratic optimization and 3-tile delay characterization (QT3) with Multimodal-Successive optimization is much lower. Thus, Multimodal-Successive optimization can handle much larger DDRO synthesis problems in a feasible run-time time which also results in much better matching DDROs in comparison to direct solvers such as CPLEX.

In order to synthesize DDROs based on delay-sensitivity matching, delay sensitivity characterization of tiles has to be performed for every technology node and for each standard cell library. Additionally, the PVT parameters contributing majorly to the delay sensitivity needs to be analyzed for every technology node. Moreover, it is not feasible to compute delay sensitivities for thousands of critical paths on a design. This makes the delay sensitivity calculation computationally intensive. Additionally, as seen in this section, the DDRO synthesis with delay sensitivity matching leads to inconsistent results. These drawbacks are overcome by the usage of delay-tracking-based DDROs.

4.3 Delay-Tracking-Based DDRO

The concept of delay-tracking-based DDROs as explained in Sec. 3.5 is to match the normalized delay of critical paths directly to that of the DDRO path instead of the delay sensitivities. This avoids the need to perform computational intensive sensitivity analysis on tiles and critical paths. Moreover, delay of a critical path or tile may be computed using static timing analysis (STA). The computationally intensity of STA is substantially less than SPICE. Thus, usage of STA enables the delay characterization of thousands of critical paths on an industrial chip. In this section, delay-tracking-based DDROs are evaluated using (1) SPICE characterized data and compared to sensitivity-tracking-based DDROs (2) SPICE based DDRO synthesis for larger range of PVT conditions (3) delay simulation by STA and STA based DDRO synthesis.

As explained in Sec. 4.2.3, Multimodal-Successive optimization method results in much better DDROs and can handle larger DDRO synthesis problems in comparison to direct solvers such as CPELX. Therefore, in this section delay-tracking-based DDROs are synthesized using the heuristic method with Multimodal-Successive optimization.

4.3.1 SPICE Data-Based DDRO Synthesis

In this section, results of delay-tracking-based DDROs synthesis are presented. In order to compare the delay-tracking-based DDRO synthesis to sensitivity-based DDRO synthesis, critical paths and tile delays are characterized using SPICE simulation with all the conditions with respect to tile types and critical paths being identical to sensitivity-based DDRO synthesis explained in Sec. 4.2.2. Table 4.7 shows the l_2 norm between DDRO delays and its respective critical path delays for all PVT corners. It can be observed that for 1-tile (T1) and 2-tile succeeding (TS2) characterization the linear optimization yields better results than the quadratic optimization. However, the quadratic optimization for TP2 and T3 tile characterization results in nearly 50% decrease in the maximum error compared to the linear optimization. Moreover, quadratic programming shows a significant improvement from 1-tile (T1) to 3-tile (T3) delay characterization. Thus, 3-tile delay characterization with the quadratic optimization (QT3) yields in an average error of 1.52 which is over 6 times less than the linear optimization with 1-tile delay characterization (LT1).

Table 4.7: Statistical evaluation of normalized error of delays for delay-tracking based DDROs using Multimodal-Successive Optimization

Method	Mean [arb. unit]	Min [arb. unit]	Max [arb. unit]	Standard deviation [arb. unit]
LT1	9.59	0.97	24.09	5.48
LTP2	9.26	0.79	23.56	5.63
LTS2	10.14	1.15	25.71	5.62
LT3	8.50	0.55	22.33	4.58
QT1	15.26	0.76	32.05	7.41
QTP2	5.97	1.89	12.89	2.30
QTS2	15.02	0.56	28.86	7.24
QT3	1.52	0.06	12.41	1.87

Figure 4.3 shows the correlation of normalized DDRO delay versus normalized critical path delay across PVT corners for delay-tracking-based synthesis. The Figure 4.3 is similar to the Figure 4.1 and the description is given in Sec. 4.2.1. From Figure 4.3 it can be observed that the different corners do not lie on the solid line for all the linear optimization methods (LT1, LTS2, LTP2, LT3). Therefore, linear optimization for delay-tracking-based DDROs is not a suitable method to synthesize DDROs. Moreover, it is seen that for quadratic optimization the dotted line deviates from the solid line for 1-tile and 2-tile with succeeding tile (QT1, QTS2) whereas the solid line and dotted line

overlay for 2-tile delay with preceding tile and 3-tile delay characterization (QTP2, QT3). Furthermore, the average and maximum error are significantly lower with a reduction of maximum error from 2.65% to 1.98% and mean error from 0.57% to 0.17% for QTP2 to QT3 method of synthesize DDROs. Thus, quadratic optimization with 3-tile delay characterization results in the best method to synthesize delay-tracking-based DDROs.

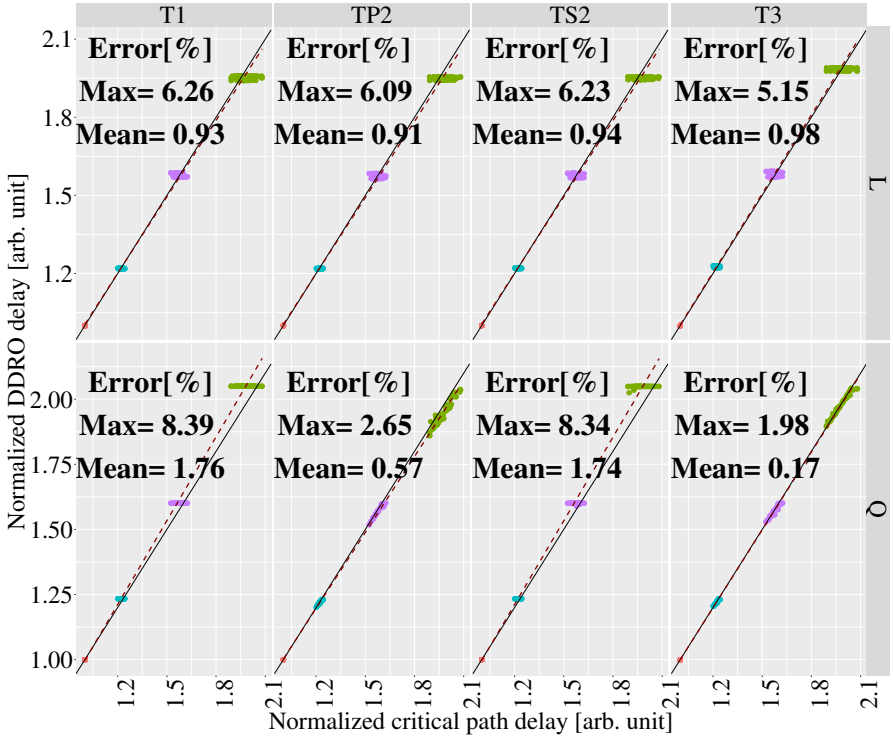


Figure 4.3: Correlation between synthesized DDROs delays and critical path delays across PVT corners for delay-tracking-based DDROs synthesized using heuristic solver with Multimodal-Successive optimization methods

From Figures 4.2 and 4.3, the quadratic optimization with 3-tile delay characterization (QT3) results in reduction of mean error from 0.64% to 0.17% and reduction of maximum error from 3.75% to 1.98% for sensitivity-based to delay-tracking-based DDRO, respectively. Therefore, delay-tracking-based DDROs synthesized using quadratic optimization with 3-tile delay characterization (QT3) result in best matching DDROs and emerges as the best method for DDRO synthesis.

4.3.2 SPICE-Based DDRO Synthesis for Larger-Range of PVT conditions

It was established in the previous sections that the delay sensitivity characterization for tiles and critical paths consumes extremely large CPU run-times and would not be feasible for larger number of PVT corners. Thus, this experiment is aimed to evaluate

the ability of delay-tracking-based DDROs to track critical path delays for larger number of PVT conditions.

Table 4.8: Experimental parameters for the SPICE-based DDRO synthesis for larger-range of PVT conditions using Multimodal-Successive optimization method

Parameter	Value(s)
DDRO objectives	Quadratic
Tile delay characterization	T3
Tile types	INV, NAND2, NOR2
Number of drive strengths of tile types	2
Number of tile types n_φ	6
Tile length	3
Tile block length n_s	5
Solver	Multimodal-Successive Optimization Method
Number of PVT corners	9
Delay data	SPICE simulation

The parameters used in this experiment is given in Table 4.8. Also, this experiment used 9 instead of 4 PVT corners for the DDRO synthesis. As explained in the previous section the quadratic optimization with 3-tile delay characterization (QT3) results in the best DDROs. Therefore, in this section DDROs are synthesized only with the QT3 method.

Figure 4.4 shows the correlation of normalized DDRO delay versus normalized critical path delay across PVT corners for the delay-tracking-based DDRO synthesis. It can be seen from Figure 4.4 that, the correlation points between the 96 different DDROs and their respective critical paths lie on the 45 degree line for all the 9 different corners of PVT. Additionally, the maximum and mean error of delay in % is calculated according to Equation (3.36). The maximum error for 9 different PVT corners is 1.74% and the mean error is 0.25%. These values are comparable to the DDRO synthesis with 4 corners shown in Figure 4.3 which results in a maximum and mean error of 1.98% and 0.17%, respectively with the quadratic optimization and 3-tile delay characterization (QT3). Thus, the delay-tracking-based DDRO synthesis is feasible for a wide range of corners and results in a good match to critical paths.

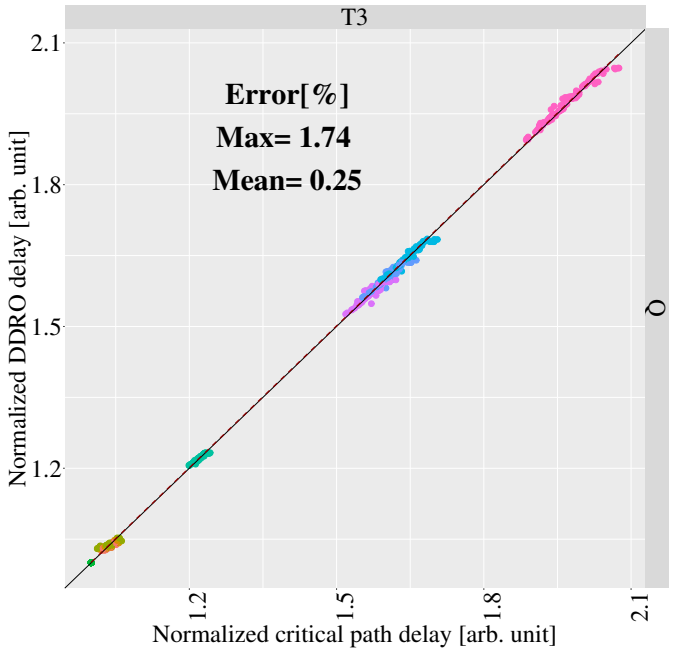


Figure 4.4: Correlation between synthesized DDROs delays and critical path delays across 9 PVT corners for delay-tracking-based DDRO synthesized using heuristic solver with Multimodal-Successive optimization methods

4.4 STA-Based DDRO Synthesis

In this section the delay-tracking-based DDROs are synthesized using delays for tiles and critical paths obtained from static timing analysis (STA) [61]. STA provides a simple and faster method to analyze timing of digital circuits in comparison to SPICE. The CPU run-time to obtain delay values of 30 tile types for 3-tile delay characterization takes more than 1 week with SPICE simulation whereas with STA, the tile delays are obtained within a few seconds. Moreover, timing analysis of thousands of critical paths is only feasible with the usage of STA and not with SPICE.

The parameters used for the experiment of STA-based DDRO synthesis for larger-range of PVT conditions using Multimodal-Successive optimization method is given in Table 4.9. The selected critical paths used in this section are extracted from a physical design placed and routed by commercial tools which also includes parasitic RC elements.

Figure 4.5 shows the correlation of normalized DDRO delay versus normalized critical path delay across PVT corners for STA-based delay-tracking-based DDRO synthesis. In Figure 4.5, a 45 degree solid line represents the ideal regression line plotted as a reference. The dotted line gives the actual regression line of each DDRO synthesis methods. The different colors represent the different PVT conditions. Moreover, ±5% regression lines are also plotted. It can be observed that all the correlation points across all PVT conditions lie within ±5% error lines. Additionally, the maximum and mean

Table 4.9: Experimental parameters for the STA-based DDRO synthesis for larger-range of PVT conditions using Multimodal-Successive optimization method

Parameter	Value(s)
DDRO objectives	Quadratic
Tile delay characterization	T3
Tile types	INV, NAND2, NOR2
Number of drive strengths of tile types	2
Number of tile types n_φ	6
Tile length	3
Tile block length n_s	5
Solver	Multimodal-Successive Optimization Method
Number of PVT corners	9
Delay data	RC extracted STA

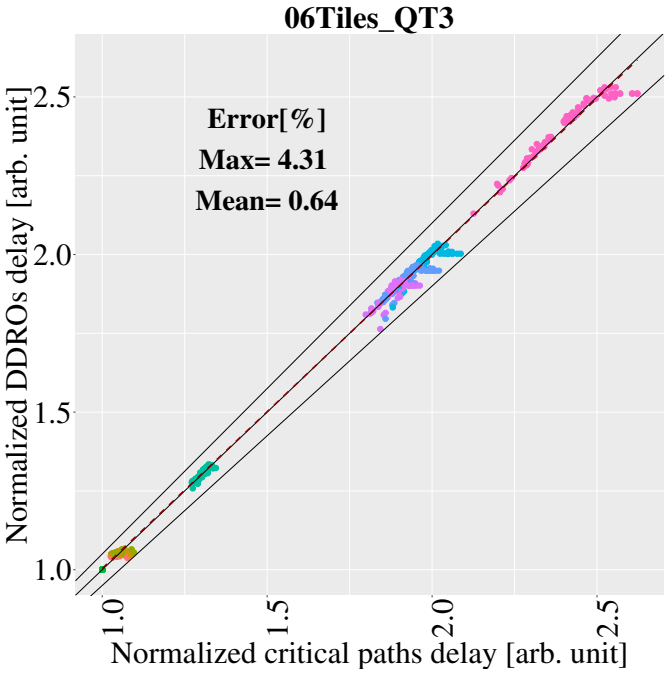


Figure 4.5: Correlation between synthesized DDROs delays and critical path delays across 9 PVT corners for delay-tracking-based DDROs synthesized using STA data

error of delay in % is calculated according to Equation (3.36). In Figure 4.5, it can be observed the maximum error is 4.3% and mean error is 0.6%. Thus, DDROs synthesized with STA data result in a good compromise between simulation time and delay-tracking ability to synthesize DDROs for large scale industrial design by obtaining delay values with STA for thousands of critical paths on a chip.

DDROs are synthesized in the next sections, namely Sec. 4.5, Sec. 4.6, Sec. 4.7, Sec. 4.8 and Sec. 4.9 using delay obtained from STA data and these DDROs are evaluated for variations in the experimental parameters of the DDRO synthesis algorithms such as (2) heuristic algorithm, (2) tile block length, (3) tile length, and (4) selection of tile types.

4.5 Comparison of Heuristic Methods

In this thesis, three different Heuristic methods are developed for the synthesis of DDROs. These methods are explained in Sec. 3.6. DDRO synthesis could not be executed in feasible time for larger DDRO synthesis problems using direct solvers such as CPLEX and LP solve. Therefore, the Heuristic methods were developed. Until this section, DDROs are synthesized with the Multimodal-Successive optimization method. In this section, the synthesized DDROs are evaluated for two other methods (1) Single-Mode-Successive optimization explained in Sec. 3.6.6 and (2) Single-Mode Back-Tracking optimization explained in Sec. 3.6.7. The parameters used for the experiment of comparing different Heuristic methods are given in Table 4.10.

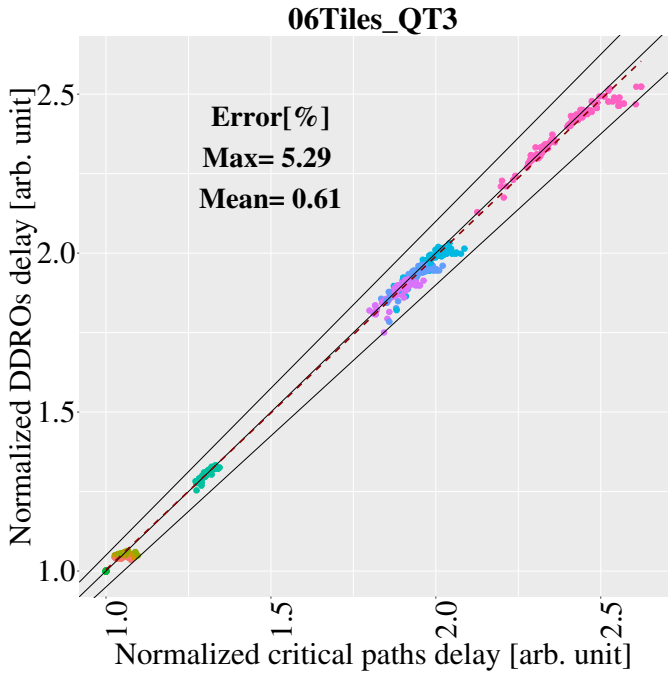


Figure 4.6: Correlation between synthesized DDROs delays and critical path delays across 9 PVT corners for delay-tracking-based DDROs synthesized using STA data for Single-Mode-Successive optimization method

Figures 4.6 and 4.7 shows the correlation of the normalized delays across 9 different PVT corners between 96 different critical paths and synthesized DDROs based on Single-Mode-Successive and Single-Mode Back-Tracking optimization, respectively. Figures 4.6

Table 4.10: Experimental parameters for the STA-based DDRO synthesis using different Heuristic optimization methods

Parameter	Value(s)
DDRO objectives	Quadratic
Tile delay characterization	T3
Tile types	INV, NAND2, NOR2
Number of drive strengths of tile types	2
Number of tile types n_φ	6
Tile length	3
Tile block length n_s	5
Solver	Multimodal-Successive optimization, Single-Mode Successive optimization, Single-Mode Back-Tracking optimization
Number of PVT corners	9
Delay data	RC extracted STA

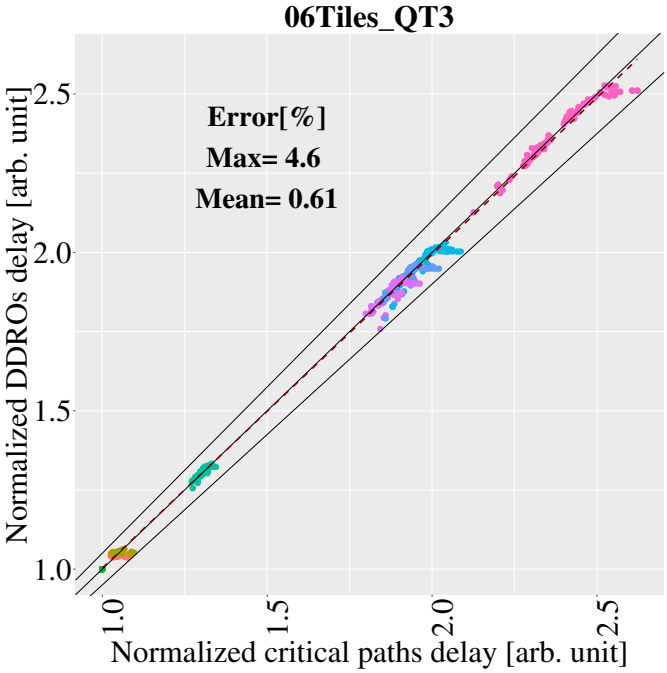


Figure 4.7: Correlation between synthesized DDROs delays and critical path delays across 9 PVT corners for delay-tracking-based DDROs synthesized using STA data for Single-Mode Back-Tracking optimization

and 4.7 are similar to Figure 4.5 and the description of the Figures is explained in Sec. 4.4. From Figures 4.5, 4.6, 4.7 it can be seen that for all the three different methods of

heuristic optimization, the mean error is the same and is around 0.6%. However, there are differences in maximum error among the three methods. The maximum error is lowest for the Multimodal-Successive optimization with an error of around 4.3% and, for Single-Mode Back-Tracking optimization is around 4.6%. Among the three algorithms, the maximum error is the largest for the most simple heuristic method developed which is the Single-Mode-Successive optimization with an error of around 5.3%.

Table 4.11: CPU run-time for the three different Heuristic algorithms

Heuristic method	Generation of $S^{(X)}(it)$ [s]	Heuristic Synthesis [s]
Single-Mode Successive	7.25	6.51
Multimodal Successive	6.10	29.07
Single-Mode Back-Tracking	8.21	8.92

Table 4.11 shows the CPU run-time for the three different heuristic algorithms (Single-Mode-Successive, Multimodal-Successive and Single-Mode Back-Tracking). The second column presents the CPU run-time to generate the Enumeration sets $S^{(X)}(it)$. The third column shows the average run-time of the Heuristic synthesis. It can be seen that the run-time to generate the enumeration sets $S^{(X)}(it)$ is nearly the same for all the methods and takes around 7 seconds to generate the complete enumeration sets of tile blocks for 6 tile types in the tile library. Additionally, the run-time to perform the Heuristic synthesis is nearly equal for both Single-Mode-Successive and Single-Mode Back-Tracking algorithms with an average run-time of around 7 seconds. However, the Multimodal-Successive takes more CPU run-time with an average of round 29 seconds. Moreover, Multimodal-Successive optimization as explained in Sec. 3.6.5 selects more than one solution in each iteration and therefore has a wider search space which may lead to be a more global in comparison to the other two methods. This was also observed in this experiment where maximum error is 4.3%.

Thus to summarize, in this thesis three new Heuristic algorithms were developed in contrast to the direct solvers to solve larger DDRO synthesis problems. Among the three methods, although CPU run-time is the highest with the Multimodal-Successive optimization, this method resulted in the best delay-tracking DDROs.

4.6 Impact of Tile-Block Length

For the three different Heuristic methods, the basic concept as explained in Sec. 3.6.3 is to generate enumeration sets of all possible tile blocks of certain tile block length n_s for all given tile types in the tile library set. In this section, the synthesized DDROs are evaluated for different lengths of tile blocks $n_s = \{3, 5, 7\}$. The larger the tile block length, the larger is the number of possible combinations of tile blocks at each iteration $n_c^{(X)}(it)$. Thus, increase in tile block length should increases the CPU run-time for the DDRO synthesis. The parameters used for the experimental to analyze the impact of different tile block lengths in the DDRO synthesis is given in Table 4.12.

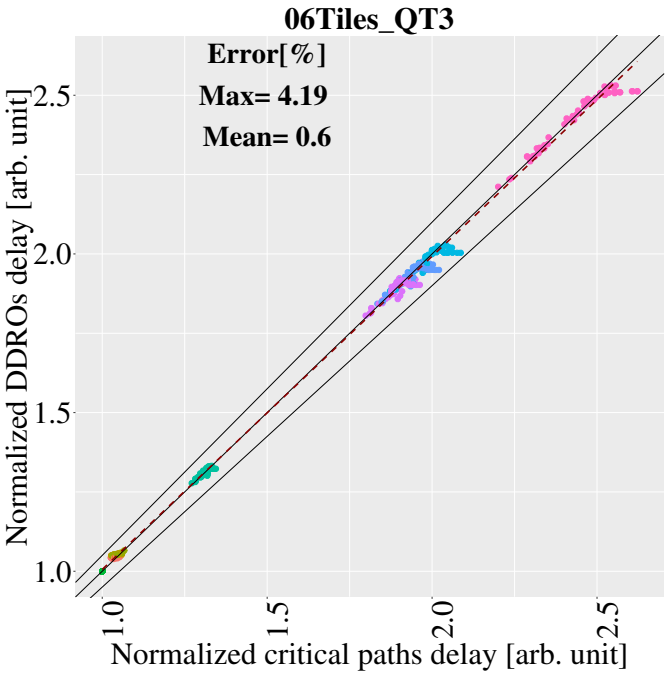


Figure 4.8: Synthesized DDRO delays vs. critical path delays across 9 PVT corners for delay-tracking-based DDROs synthesized using STA data for $n_s = 3$

Table 4.12: Experimental parameters for the STA-based DDRO synthesis for different tile block lengths

Parameter	Value(s)
DDRO objectives	Quadratic
Tile delay characterization	T3
Tile types	INV, NAND2, NOR2
Number of drive strengths of tile types	2
Number of tile types n_φ	6
Tile length	3
Tile block length n_s	3, 5, 7
Solver	Multimodal-Successive optimization
Number of PVT corners	9
Delay data	RC extracted STA

From Figures 4.8, 4.5 and 4.9 where $n_s = \{3, 5, 7\}$, it can be seen that the mean error is around 0.6% for all the tile block lengths. However, the maximum error is the least for $n_s = 3$ with 4.2% error and is the highest for $n_s = 7$ with 4.7% error. The maximum error for $n_s = 5$ is around 4.3%.

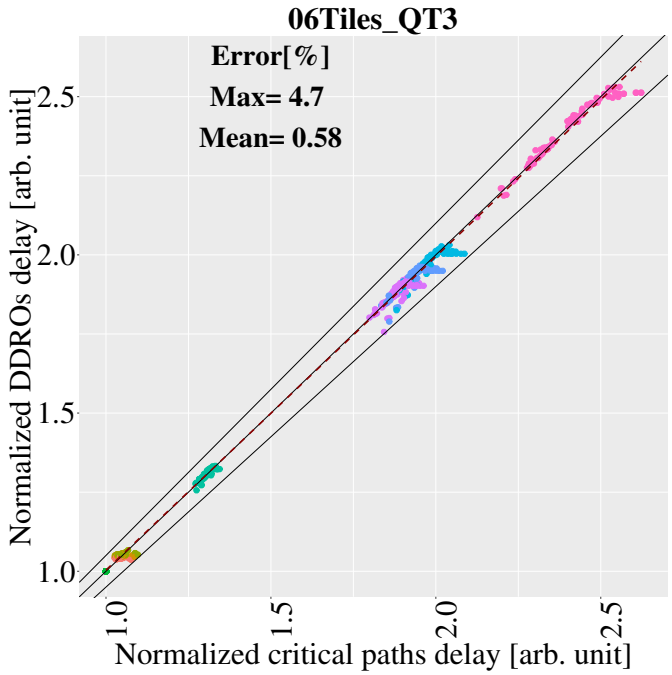


Figure 4.9: Synthesized DDRO delays vs. critical path delays across 9 PVT corners for delay-tracking-based DDROs synthesized using STA data for $n_s = 7$

Table 4.13: CPU run-time for three different tile block lengths n_s

Tile block length	Generation of $S^{(X)}(it)$ [s]	Heuristic Synthesis [s]
$n_s = 3$	1.17	9.47
$n_s = 5$	6.10	29.07
$n_s = 7$	151.59	1087.14

Table 4.13 shows the CPU run-time for the three different tile block lengths $n_s = \{3, 5, 7\}$. It can be seen that the average run-time to generate the enumeration sets $S^{(X)}(it)$ and the heuristic synthesis execution increases exponentially from 1.17 to 151.79 and from 9.47 to 1087.14 seconds from $n_s = 3$ to $n_s = 7$, respectively. Moreover, as explained earlier, increasing the tile block length n_s does not result in better DDROs. Thus, tile block lengths $n_s = \{3, 5\}$ are identified as good candidates for tile block length to synthesize DDROs using STA data for the given set of critical paths.

4.7 Impact of Tile Length

DDROs are built by concatenating a set of tiles where tiles are concatenation of standard cells of equal type. The concept of using tiles instead of individual standard cells is to reduce the impact of input slew and output load on the inner cells of the tile such that the

delays of tiles can be linearly added up to form the DDRO's delay across PVT variations. Thus, the tile length or the number of standard cells concatenated to construct a tile is user-defined. In this section, the synthesized DDROs are evaluated based on the tile length. The different tile lengths chosen for the experiment are $\{3, 5, 7\}$. The parameters used for the experiment to analyze the impact of tile length in the DDRO synthesis is given Table 4.14.

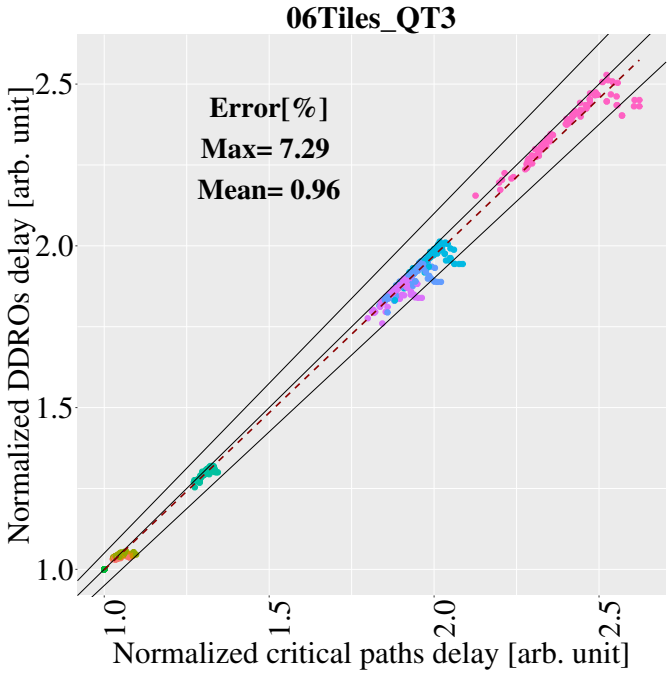


Figure 4.10: Correlation between synthesized DDRO delays and critical path delays across 9 PVT corners for delay-tracking-based DDROs synthesized using STA data for 5 standard cells in a tile

From Figures 4.5, 4.10 and 4.11, it can be seen that tile length 3 has the least mean and maximum error of around 0.6% and 4.3%, respectively. As the length of a tile is increased from 3 to 5 cells, the mean and maximum error increase to around 1% and 7.3%, respectively. With the tile length set to 7 cells, the mean error reduces slightly as compared to tile length 5 to around 0.9% and maximum error further increases to 7.4%. Moreover, as the length of a tile is increased, the number of tiles selected to synthesize a DDRO of a specific length reduces. Thus, smaller the tile length the more freedom there is to choose more tiles types that match the delay behavior of critical paths.

Table 4.15 shows the CPU run-times for the three different tile lengths $\{3, 5, 7\}$. The second column presents the CPU run-time to generate the Enumeration sets $S^{(X)}(it)$. The third column shows the average run-time of the Heuristic synthesis. It can be seen that the average run-time is nearly equal for all the three different tile lengths with an average of 6 and 30 seconds for generation of the enumeration sets $S^{(X)}(it)$ and run the Heuristic synthesis, respectively. Thus, to summarize it was seen that tile length 3

Table 4.14: Experimental parameters for the STA-based DDRO synthesis for different tile lengths

Parameter	Value(s)
DDRO objectives	Quadratic
Tile delay characterization	T3
Tile types	INV, NAND2, NOR2
Number of drive strengths of tile types	2
Number of tile types n_φ	6
Tile length	3, 5, 7
Tile block length n_s	3
Solver	Multimodal-Successive optimization
Number of PVT corners	9
Delay data	RC extracted STA

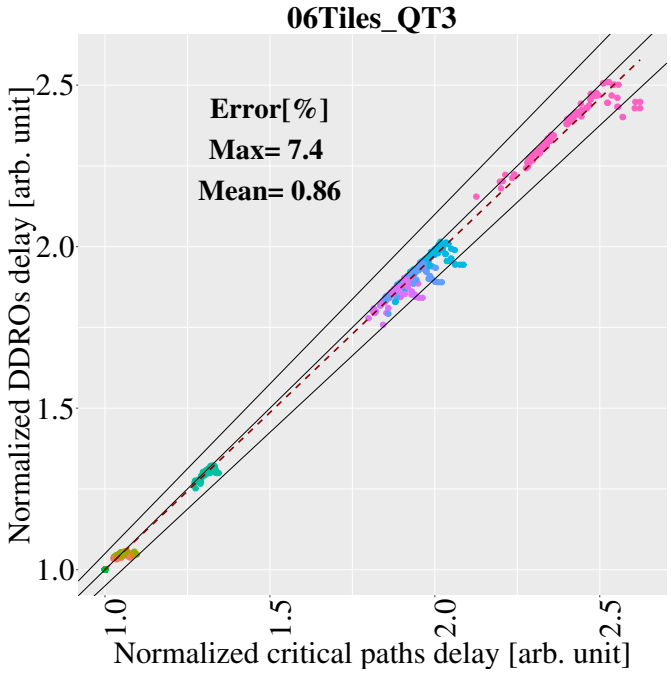


Figure 4.11: Correlation between synthesized DDRO delays and critical path delays across 9 PVT corners for delay-tracking-based DDROs synthesized using STA data for 7 standard cells in a tile

resulted in the best matching DDROs with acceptable CPU run-times. Therefore, in this technology node and for the critical paths considered in this thesis, tile length of 3 is the best choice to synthesize DDROs.

Table 4.15: CPU run-time for the three different tile lengths 3, 5, 7

Tile length	Generation of $S^{(X)}(it)$ [s]	Heuristic Synthesis [s]
3	6.18	30.97
5	6.10	29.07
7	6.83	31.63

4.8 Impact of Tile Selection

In this section, the results of the delay-tracking-based DDRO synthesis are discussed with respect to selection of different tile libraries and the parameters used for the experiment is given Table 4.14. The tile library consists of NAND2,NOR2,INV in $\{1, 2, 3, 4\}$ drive strengths which leads to libraries with $\{3, 6, 8, 9, 12, 15\}$ elements for the synthesis. For the DDRO synthesis with 8 tile types, there are two additional tile types in addition to NAND2, NOR2, INV for the DDRO synthesis.

Table 4.16: Experimental parameters for the STA-based DDRO synthesis for different tile library sets

Parameter	Value(s)
DDRO objectives	Quadratic
Tile delay characterization	T3
Tile types	INV, NAND2, NOR2, 2 additional tile types
Number of drive strengths of tile types	1, 2, 3, 4
Number of tile types n_φ	3, 6, 8, 9, 12
Tile length	3
Tile block length n_s	3
Solver	Multimodal-Successive optimization
Number of PVT corners	9
Delay data	RC extracted STA

Table 4.17: The error of tracking delays of critical paths using DDROs (QT3) for different tile library sets

RO Type	Mean error [%]	Max error [%]
QT3 $n_\varphi = 3$	1.02	7.27
QT3 $n_\varphi = 6$	0.55	4.31
QT3 $n_\varphi = 8$	0.44	4.31
QT3 $n_\varphi = 9$	0.52	4.26
QT3 $n_\varphi = 12$	0.52	4.26

Figure 4.12 shows the correlation between normalized delays of DDROs and critical paths for the different tile library sets. Comparing the results of different tile library sets in Figure 4.12 and Table 4.17, it can be seen that with an increase in number of tiles in the tile library from 3 to 6 tile types, there is a significant decrease in the mean error and around 3% percent decrease in the maximum error. With increase in number of tile types from 6 to 12, there is a slight decrease in the maximum error to around 4.26% along with a slight increase in the mean error to 0.52% . Although local optimization using Multimodal-Successive method results in slightly reduced performance of the synthesized DDROs, this reduction is negligible in comparison to the benefits of having reduced computational complexity.

Table 4.18: CPU run-time for different tile library sets

Tile length	Generation of $S^{(X)}(it)$ [s]	Heuristic Synthesis [s]
3	2.37	7.8
6	6.823	28.06
8	24.15	125.068
9	52.48	276.94
12	408.166	2091.24

Table 4.18 shows the CPU run-times for five different tile library sets. It can be seen that on an average less CPU run-time is required to generate the enumeration sets $S^{(X)}(it)$ in comparison to the heuristic synthesis. Moreover, the CPU run time increases exponentially with increase in number of tile types in the tile library sets. Furthermore, it was already presented in this section that the maximum and the mean errors do not reduce significantly with an increase in tile sets beyond 6 tile types. Therefore, using 6 different tile types of NAND2, NOR2, INV is a good compromise between the DDRO matching quality and short CPU run-times.

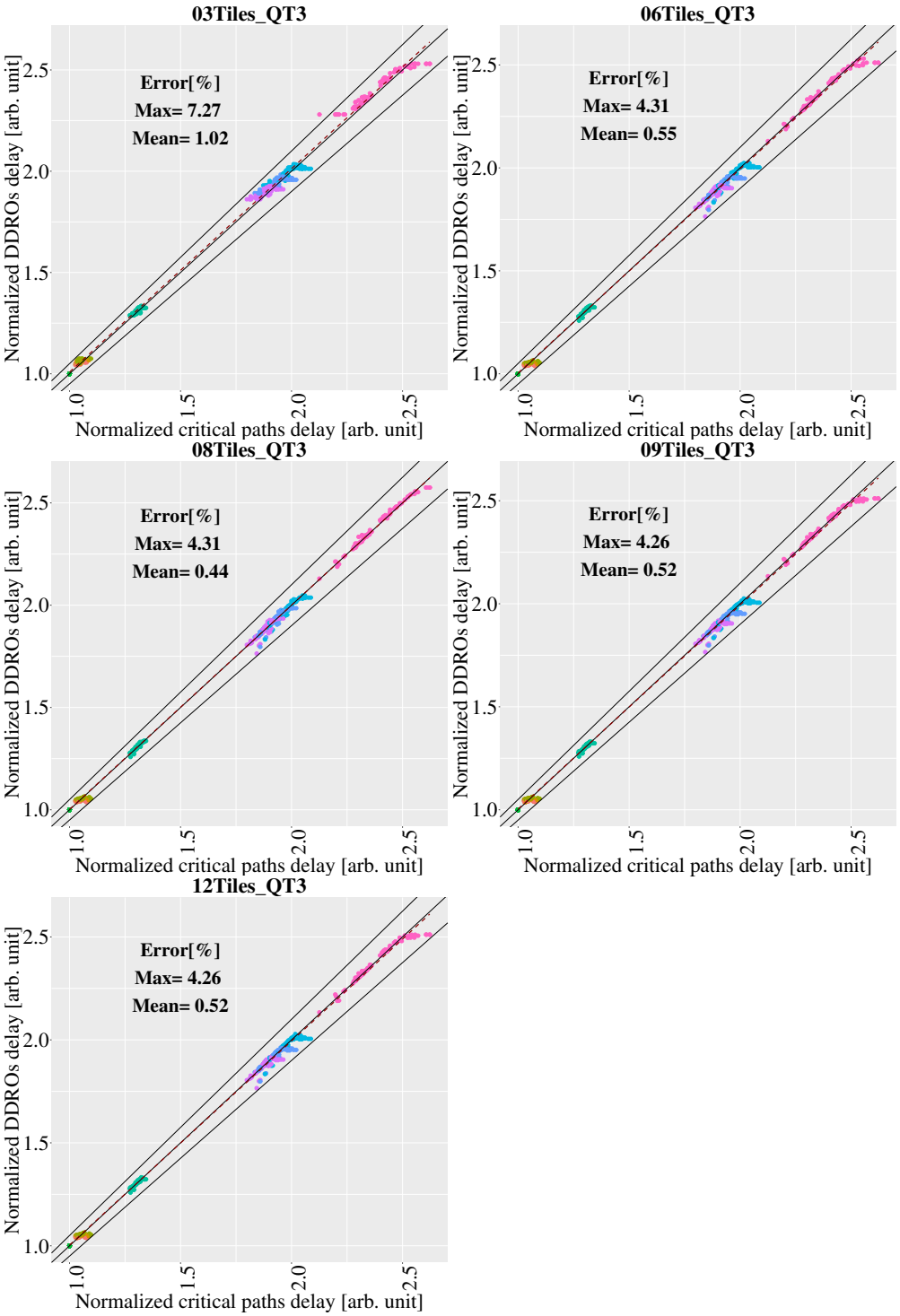


Figure 4.12: Correlation between synthesized DDROs delays and critical path delays across 9 PVT corners for delay-tracking-based DDROs synthesized using STA data constructed using different tile library sets (03T, 06T, 08T, 09T, 12T)

4.9 Comparison of GROs versus DDROs

Generic-ring oscillator monitors (GROs) are traditionally used timing monitors. GROs are made of single cell type ring oscillators and are not constructed based on a specific design. In this section, the design representativeness of GROs is analyzed and is compared with DDROs. The GROs used in this experiment are inverter (INV), NAND2 and NOR2 ring oscillators.

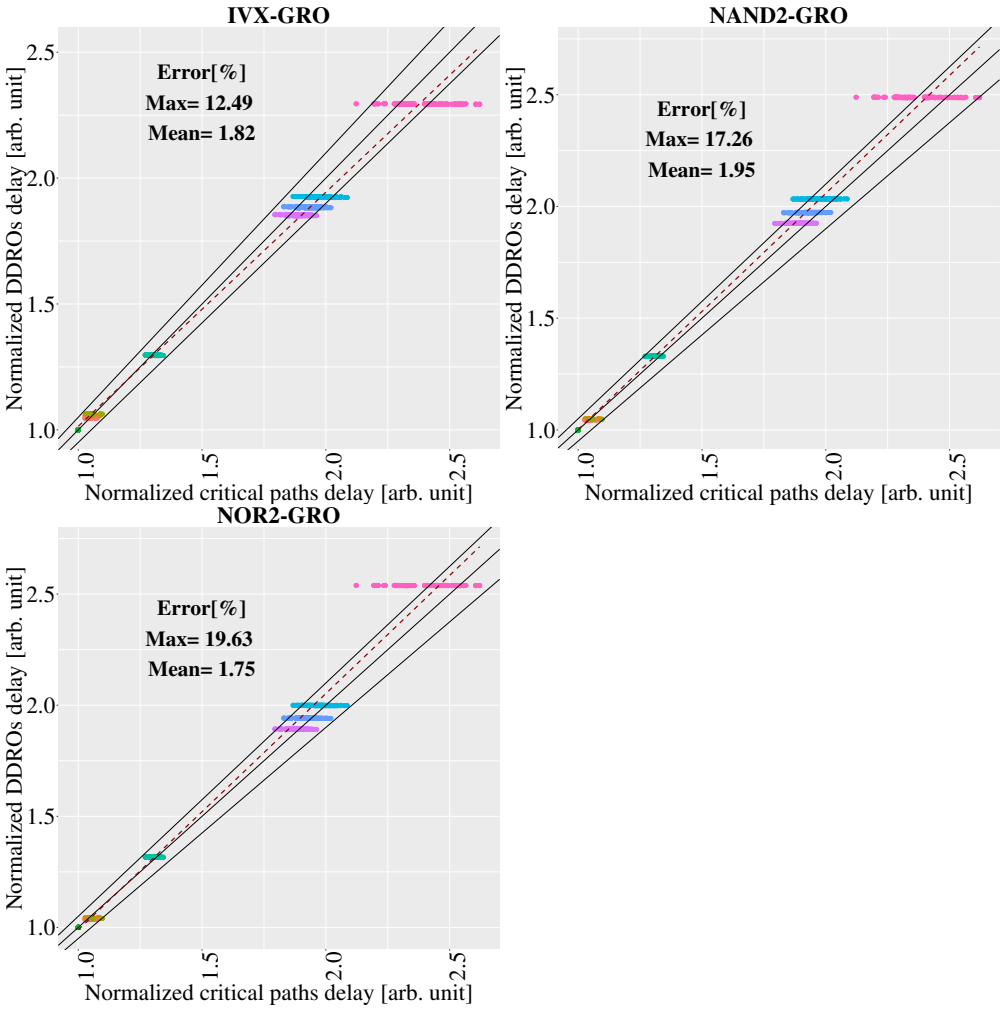


Figure 4.13: Correlation between INV, NAND2 and NOR2 GRO delay and critical path delays across 9 PVT corners

Figure 4.13 shows the correlation between normalized delays of GROs (INV, NAND2 and NOR2) and critical paths. The different colors in Figure 4.13 indicate different PVTs. A 45 degree solid line represents the ideal regression line plotted as a reference. The dotted line gives the actual regression line of each DDRO synthesis methods. From Figures 4.13 and 4.12 it can be seen that, all GROs have a mean error of $> 1.75\%$ which

is around 4 times larger than the mean value of DDRO with 8 tile types in Figure 4.12. From Figure 4.13, the maximum error is the lowest for IVX-GRO with 12.49% error and the highest for NOR2-GRO with 19.63% error. On the other hand, the maximum error for 8 tile type case in Figure 4.12 is 4.31% which is around 3 times less than the IVX-GRO. Thus, DDROs can represent the design much better than the existing state-of-the-art GROs.

4.10 Summary

This Chapter illustrates the experimental results of various DDRO synthesis methods explained in Chapter 3. The synthesized DDRO delays are characterized at various PVT conditions. Additionally, the ability of DDROs to track the delay of target critical paths are evaluated. Table 4.19 summarizes some of the important results of the DDRO synthesis shown in this Chapter. The first column of Table 4.19 *RO type* indicates the type of RO that is compared with the timing behavior of target critical paths. The second column *Synthesis method* indicates if the DDROs are synthesized by sensitivity-based or delay-tracking-based DDROs. The third column *Simulation method* indicates the method of tile-delay characterization. The state-of-the-art method uses 1-tile delay characterization. The 3-tile delay characterization method has a preceding and succeeding tile to the tile to be characterized. The fourth column *Objective* indicates if the DDROs are synthesized using linear or quadratic objective. The fifth column *Data used* shows if the delays for the DDRO synthesis are obtained using STA or SPICE. The sixth column *Number of tiles types* simply show the number of tiles types used for the DDRO synthesis. The last column *mean error in [%]* indicates the mean value of the delay tracking error for all the DDROs synthesized for a specific input configuration calculated based on the Equation 3.36.

Table 4.19: Summary of experimental results of the DDROs

RO type	Synthesis method	Simulation method	Objective	Data used	Number of tiles types	Mean error [%]
DDRO	Sens-based	1-tile delay	Linear	SPICE	6	1.25
DDRO	Sens-based	3-tile delay	Quadratic	SPICE	6	0.64
DDRO	Delay-based	3-tile delay	Quadratic	SPICE	6	0.17
DDRO	Delay-based	3-tile delay	Quadratic	STA	6	0.55
DRO	Delay-based	3-tile delay	Quadratic	STA	12	0.52
IVX-GRO	-	-	-	STA	-	1.75

The first row in Table 4.19 shows the DDRO synthesis using the state-of-the-art method. Additionally, the delays are obtained for all the sensitivity-based DDRO synthesis methods using SPICE. The second row shows the DDRO synthesis using quadratic objective and 3-tile delays for sensitivity-based DDROs. It can be seen that the mean

delay tracking error reduces from 1.25% to 0.64% in the state-of-the-art method to quadratic objective with 3-tile delay characterization by sensitivity-based DDROs, respectively. The third row of Table 4.19 shows the results of the DDRO using the delay-tracking-based DDRO synthesis. In order to compare the sensitivity-based DDROs to delay-tracking-based DDROs the delays are obtained using SPICE simulation and all the other parameters are kept constant for the DDRO synthesis. It can be seen that the delay drastically reduces from 0.64% to 0.17% from the sensitivity-based to delay-tracking-based DDROs using 3-tile delay characterization, respectively. Thus delay-tracking-based DDRO synthesis method is better suited to synthesize DDROs that mimic the delay behavior of given critical paths.

The fourth row in Table 4.19 shows the delay-tracking results for DDROs synthesized using STA data. STA aids in the reduction of CPU run-times for delay computation in comparison to SPICE. It can be seen that the mean delay-tracking error with STA data is 0.55% for 6 tile types. Furthermore, the impact of tile type selection for the DDRO synthesis is investigated. It can be seen that the mean error reduces to 0.52% by using 12 tile types. The difference in the tile type selection between 12 and 6 lies in the usage of different drive-strengths. Thus, addition of drive strengths of the same tile types in the tile library did not have a high impact of the DDRO results. Finally, the DDROs are compared with the generic ring oscillators (GRO). The result of IVX-GRO is show in the last row of Table 4.19. As can be seen from Table 4.19, the mean error increases from 0.52% to 1.75% from using DDROs to GROs, respectively. To conclude, this Chapter has shown that DDROs are can accurately track the delays of critical paths across various PVT conditions. Moreover, the newly proposed DDRO synthesis methods of this work have much better delay-tracking ability in comparison to the state-of-art method DDRO synthesis method as well as GROs.

5 Methods to Group Critical Paths

An industrial digital design consists of thousands of critical paths. The goal of a design dependent ring oscillator (DDROs) as explained in Sec. 3.1 is constructed to mimic the delay of critical paths in a design across various PVT conditions. However, constructing DDROs for each of these critical paths would be impractical due to the large area overhead and power consumption. Thus the aim is to build fewer DDROs such that each DDRO represents the timing of a group of critical paths. Thus, critical paths on a design are grouped based on their similarity in behavior w.r.t. PVT and one DDROs is built for each of the target critical path group. In this Chapter, an overview on the concept of clustering algorithms and different clustering methods used to group critical paths based on similarity in delay behavior is explained.

Clustering is a method of unsupervised learning algorithm where these algorithms typically have an optimality criterion which is solved by heuristic methods or by using mathematical models [64]. Sec. 5.1, Sec. 5.2 and Sec. 5.3 explains the grouping of critical paths using Hierarchical, KMeans++ and Expectation-Maximization clustering algorithms, respectively. Furthermore, in order to evaluate the best suitable method to cluster critical paths for DDRO synthesis, the results of the three algorithms are compared in Sec. 5.4.

5.1 Hierarchical Clustering

In this section, the delays of critical paths across PVT conditions are grouped using Hierarchical clustering algorithm. The concept of Hierarchical clustering is to successively partition the critical paths where all the critical paths are at first, placed in its individual cluster and combined in each step until all the critical paths are placed in one cluster. Thus, the given set of critical paths are clustered in a successive manner. Additionally, in Hierarchical clustering, once two or more critical paths are placed within one group, these critical paths cannot be grouped into different clusters in subsequent iterations

In this thesis, Hierarchical clustering is performed using *hlcust* function in R [65]. Figure 5.1 shows the working principle of hierarchical clustering. Critical path delays $\mathbf{d}'_{CP}(\mathbf{p}_{PVT})$ are characterized at various corners of PVT. After which, critical path delays are normalized according to Equation (3.23) in order to cluster based on the timing behavior across PVT conditions and not on the absolute value of delay. The normalized critical path delays are fed as input to the Hierarchical clustering algorithm. Hierarchical clustering algorithm first generates a series of partitions of critical paths such that each critical path is partitioned into its individual cluster. Thus, the initial number of clusters is equal to the number of critical paths n_{CP} to be clustered:

$$K(st = 1) = n_{CP} \tag{5.1}$$

The general equation for cluster sets during each successive step is written as:

$$\begin{aligned} &S_{CP_j}(st), j \in \{1, \ldots, K(st)\} \\ &\mathbf{d}^{'}_{CP_{kj}}(st, \mathbf{p}_{PVT}) \in S_{CP_j}(st) \end{aligned} \tag{5.2}$$

where $\mathbf{d}^{'}_{CP_{kj}}(st, \mathbf{p}_{PVT})$ is the critical path of index k is the index of critical path which belong to cluster j at step st. $S_{CP_j}(st)$ is the group of critical paths at step st. At each step st, the l_2 norm distance between all critical paths which do not belong to the same cluster is computed and two critical paths groups resulting in closest critical paths with least distance is found:

$$\begin{aligned} \underbrace{min}_{\substack{k_1 \in \{1 \ldots n_{CP}\} \\ k_2 \in \{1 \ldots n_{CP}\}}} & \left\| \mathbf{d}^{'}_{CP_{k_1 j_1}}(st, \mathbf{p}_{PVT}) - \mathbf{d}^{'}_{CP_{k_2 j_2}}(st, \mathbf{p}_{PVT}) \right\|_2 \quad s.t. \\ & j_1 \in \{1, \ldots, K(st)\} \\ & j_2 \in \{1, \ldots, K(st)\} \\ & j_1 \neq j_2 \end{aligned} \tag{5.3}$$

The two critical paths groups found to have the least distance are placed into one single cluster. Thus, a certain group of critical paths $S_{CPj}(st)$ at step st is a union of the critical path group $S_{CP_{j1}}(st-1)$ and the critical path group $S_{CP_{j2}}(st-1)$ found to have least distance in step st and the number of cluster sets is reduced by one:

$$\begin{aligned} S_{CP_j}(st) &= \left\{ S_{CP_{j1}}(st-1) \cup S_{CP_{j2}}(st-1) \right\} \\ K(st+1) &= K(st) - 1 \end{aligned} \tag{5.4}$$

This process goes on successively until all the critical paths are grouped into a single cluster. Figure 5.2 shows a simple example of the Hierarchical clustering method to cluster $n_{CP} = 4$ critical path delays across 2 PVT corners.

Once all the steps are completed and all the critical paths are placed into one single cluster, the initialized number of clusters given as an user-input is used to extract the required critical path cluster information [66] [67].

The results of Hierarchical clustering can be visualized by a Dendogram plot. Figure 5.3 shows a Dendogram plotted for critical paths clustered using Hierarchical clustering. Here, 256 critical paths are used for clustering which consists of cells made from two

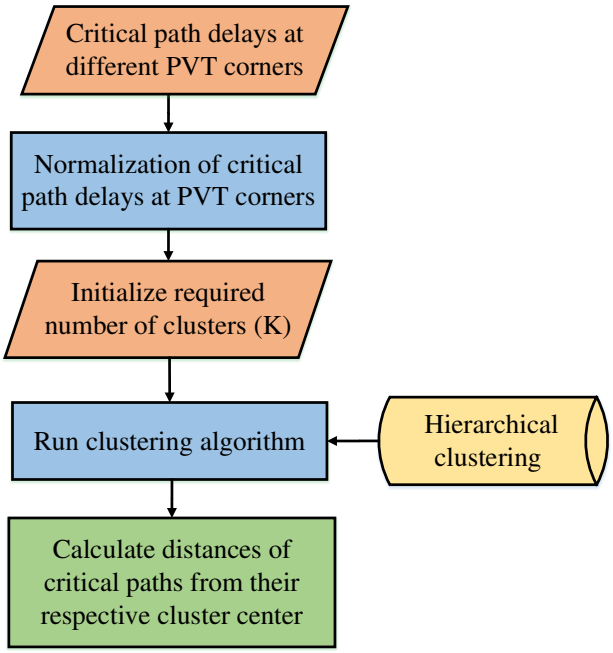


Figure 5.1: Flow of clustering critical paths using Hierarchical clustering

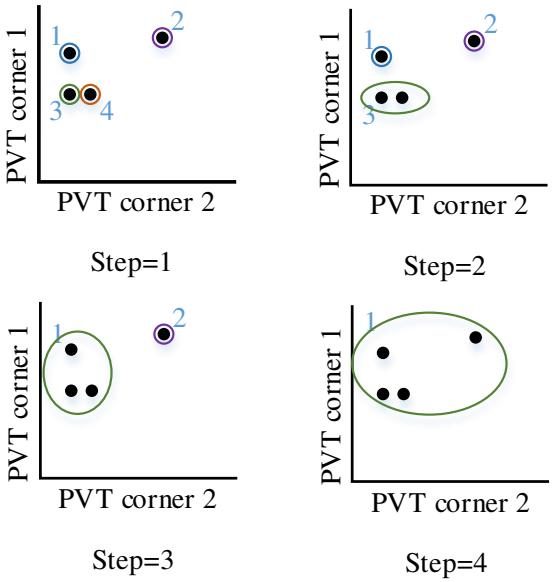


Figure 5.2: Example of Hierarchical clustering method to cluster $n_{CP} = 4$ critical path delays across 2 PVT corners

different standard cell libraries and the given set of critical paths are clustered until grouped into 13 different clusters. The structure of a Dendogram plot is similar to that of an evolutionary tree. This plot starts with one cluster and the tree evolves into

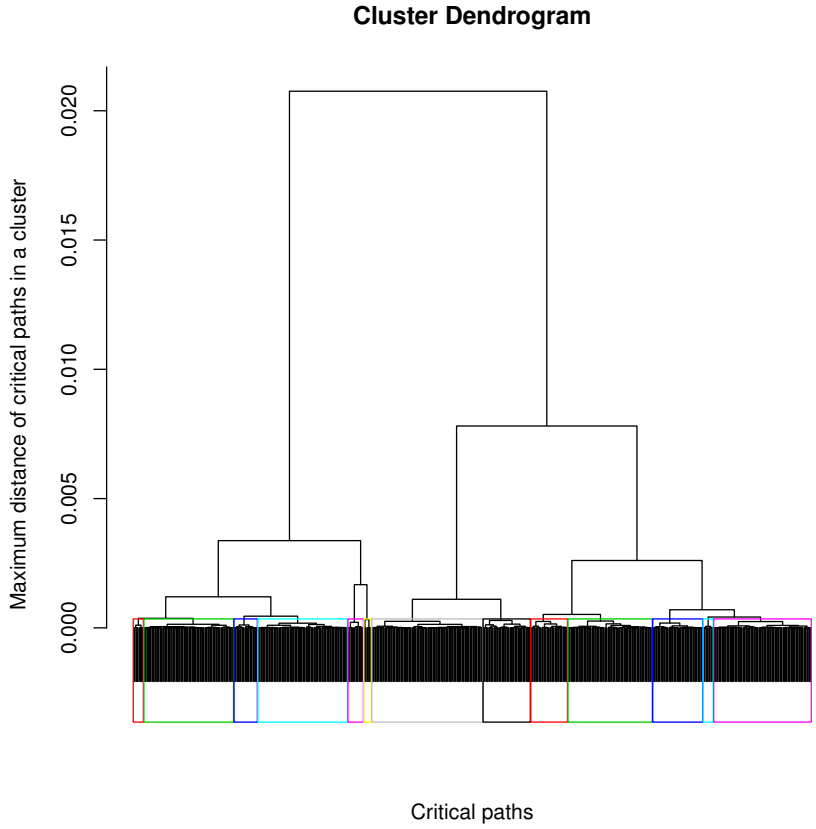


Figure 5.3: Dendogram of hierarchical clustered critical paths

multiple branches based on the size of the cluster. In this work, the maximum l_2 norm distance between two critical paths within a cluster is used as a measure to find the size of a cluster.

From Figure 5.3, it can be observed that for 13 clusters, the maximum l_2 norm distance between two critical paths within a cluster is less than 0.001. The rectangular boxes with various colors at the bottom of the Dendogram plot represents the different clusters. The number of critical paths in cluster can be observed from the Dendogram plot. It can be observed that certain clusters have many more cluster behavior where as certain critical paths contains only a couple of critical paths. The cases where only a couple of critical paths are placed in a cluster might be due to outlier timing behavior of those in comparison to other critical paths considered across all PVT conditions.

5.2 KMeans++ Clustering

KMeans++ clustering is classified into the group of unsupervised statistical learning algorithms which works by the method of iterative descent. This clustering algorithm partitions a given set of multivariate data into mutually exclusive clusters such that resulting clusters contain data points which are similar [66], [68] and [69].

In this thesis, a modified KMeans algorithm known as Kmeans++ algorithm is used where at first the data, in this case, the critical path data is fed into the algorithm. After which, the initial centroids are chosen at the first iteration. Finally, the objective in Equation (5.2) which is the within group sum of squares (WGSS) between clusters in minimized.

$$WGSS = \underset{S_{CP_j}}{min} \sum_{j=1}^{K} \sum_{j \in k} \left\| \mathbf{d}'_{CP_{kj}}(\mathbf{p}_{PVT}) - \boldsymbol{\mu}_j(\mathbf{p}_{PVT}) \right\|^2 \tag{5.5}$$

S_{CP_j} is the set of critical paths in the j cluster index and $\boldsymbol{\mu}_j(\mathbf{p}_{PVT})$ is the cluster center for cluster index j. Figure 5.4 shows the pictorial representation of WGSS. After the objective of WGSS is minimized, the data points across clusters are reassigned and the centroid values are recalculated. This step of minimizing the WGSS is continued until the maximum number of iterations is reached. Finally, the results of clustered data is returned as output.

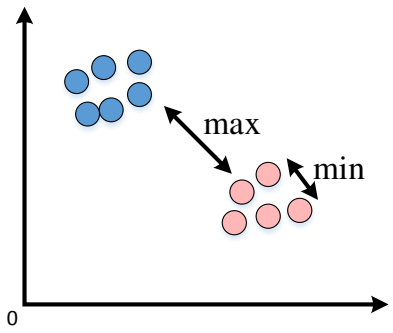


Figure 5.4: Pictorial Demonstration of WGSS [66]

In addition to the $WGSS$, the between sum of squares $Between_{SS}$ and total sum of squares $Total_{SS}$ are also computed by which the variance explained by the clusters is computed:

$$
\begin{aligned}
Total_{SS} &= \sum_{j=1}^{K}\sum_{k=1}^{g_j}\left\|\mathbf{d}'_{CP_{kj}}(\mathbf{p}_{PVT}) - \boldsymbol{\mu}_j(\mathbf{p}_{PVT})\right\|^2 \\
Between_{SS} &= \sum_{j=1}^{K}\left\|\boldsymbol{\mu}_T(\mathbf{p}_{PVT}) - \boldsymbol{\mu}_j(\mathbf{p}_{PVT})\right\|^2 \\
Var_K &= \frac{Between_{SS}}{Total_{SS}}
\end{aligned}
\tag{5.6}
$$

where g_j is the total number of critical paths in each cluster j, $\boldsymbol{\mu}_T(\mathbf{p}_{PVT})$ is the total mean of all the critical paths across PVT conditions.

Figure 5.5 gives the method to cluster critical paths using Kmeans++ clustering. At first, critical paths are identified in a design and characterized for various PVT conditions. The critical path delays are then normalized so that critical paths are clustered based on the similarity in their behavior across PVT conditions. An initial value of number of clusters is initialized by the user. After which, Kmeans++ algorithm is run and set of clusters of critical path data is obtained.

In order to determine the minimum number of clusters for the given set of data, an elbow plot is used as shown in Figure 5.6 which gives the variance of clustering Var_K for a range of clusters. Moreover, the distance of each critical path delay from its respective cluster center is calculated as:

$$
\boldsymbol{\epsilon}_{kj}(\mathbf{p}_{PVT}) = \begin{bmatrix} \epsilon_{kj}(\mathbf{p}_{PVT_1}) \\ \epsilon kj(\mathbf{p}_{PVT_2}) \\ \cdots \\ \epsilon_{kj}(\mathbf{p}_{PVT_{n_{PVT}}}) \end{bmatrix} = \begin{bmatrix} 1 - \frac{d'_{CP_{kj}}(\mathbf{p}_{PVT_1})}{\mu_j(\mathbf{p}_{PVT_1})} \\ 1 - \frac{d'_{CP_{kj}}(\mathbf{p}_{PVT_2})}{\mu_j(\mathbf{p}_{PVT_2})} \\ \cdots \\ 1 - \frac{d'_{CP_{kj}}(\mathbf{p}_{PVT_{n_{PVT}}})}{\mu_j(\mathbf{p}_{PVT_{n_{PVT}}})} \end{bmatrix}
\tag{5.7}
$$

The maximum distance of critical path from its respective cluster is limited by a user defined coefficient ϵ_{max}, which should hold for all the clusters and across all PVT conditions. If necessary the number of clusters K is increased and the clustering algorithms along with the distance computations are re-run until the above condition is satisfied.

Figure 5.6 gives the elbow plot for variance of clustering Var_K for a range of clusters. Here, 256 critical paths are used as cluster data which consist of cells made from two different standard cell libraries. The critical paths are characterized at 9 different PVT corners using STA. In Figure 5.6, the point of elbow of the plot represents the minimum number of clusters to be selected and it is observed that the minimum number of clusters required is around 8.

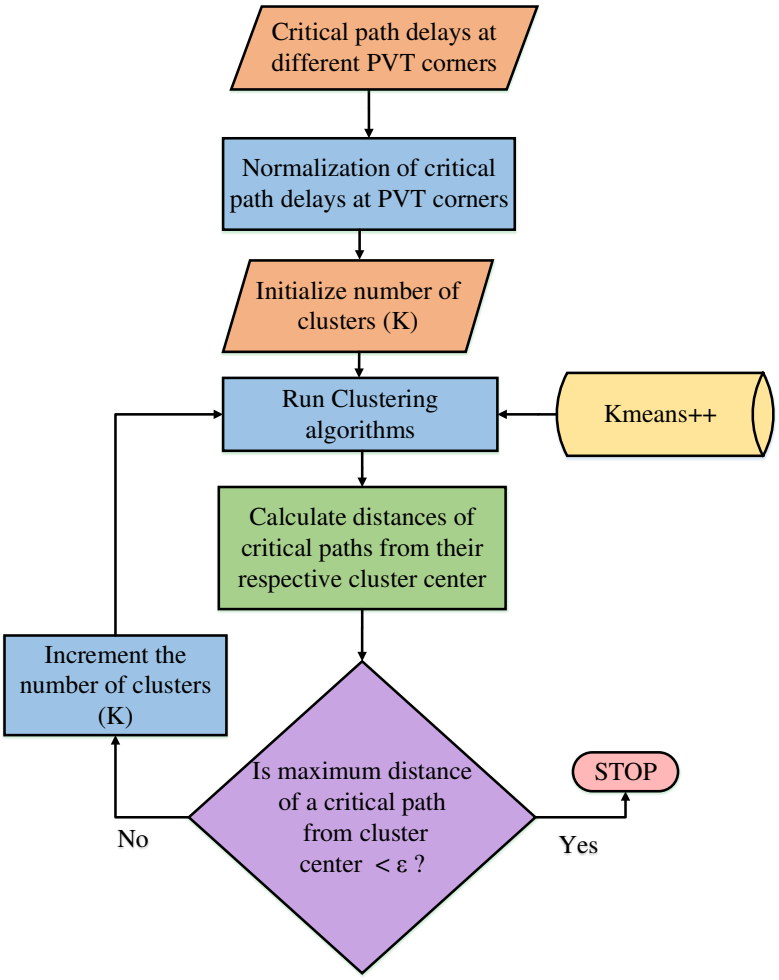


Figure 5.5: Method for clustering critical paths using Kmeans++

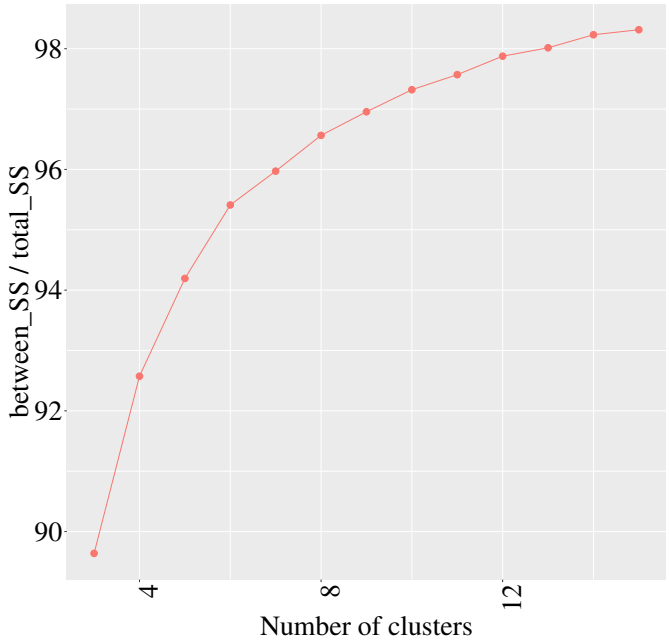


Figure 5.6: Minimum number of clusters using elbow plot

5.3 Model-Based Clustering

Hierarchical and KMeans++ clustering algorithms explained earlier in Sec. 5.1 and Sec. 5.2, respectively are heuristic algorithms. Expectation maximization (EM) clustering is a model based clustering method which assumes a statistical model for the given set of critical paths delays. This model assumes that critical paths contain statistical data and can be modeled into further sub-models with various multivariate probability density functions. These multivariate probability density functions result in a finite mixture density for the whole set of critical paths [66] [70]. Unlike Hierarchical and KMeans++ algorithm, EM clustering is performed only once. Moreover, EM clustering is a method of soft clustering where each data point or critical path is not fully sorted into a certain cluster but the algorithm defines the probability of a certain critical path point fitting into a certain model and cluster. However, EM algorithm is only advantageous if the critical path data can be well fit into statistical models. As critical path data is not computed by statistical evaluation, EM clustering may not be the best method of clustering of the target critical paths in comparison to the simpler Heuristic methods such as Hierarchical and KMeans++ clustering.

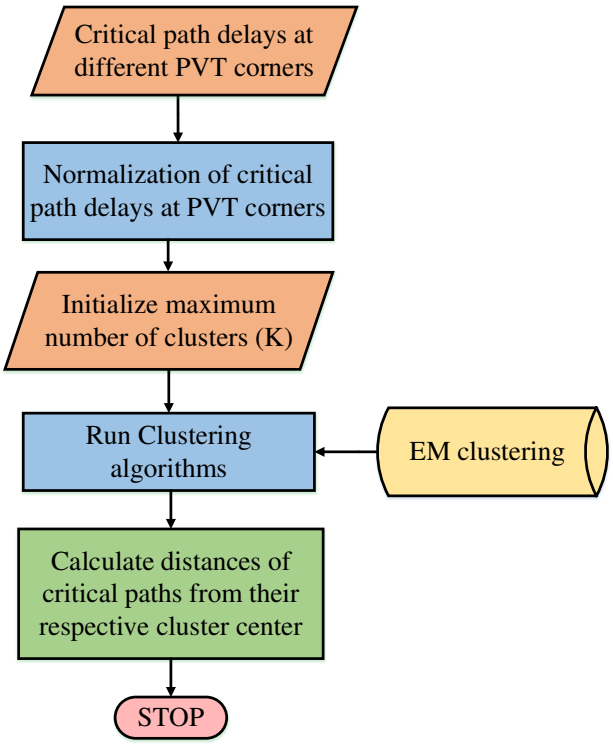


Figure 5.7: Flow of Expectation-Maximization (EM) clustering algorithm

Figure 5.7 shows the flow diagram of EM clustering algorithm. The critical path delays are characterized at various conditions of PVT and the normalized critical path data is fed as input into the EM clustering method. Additionally, a maximum value of clusters

K_{max} is fed as input into the EM algorithm. The first stage of the EM algorithm involves the computation of the Bayesian Inversion Criterion (BIC) values where the BIC function tries to fit the critical path delay data into a range of cluster numbers. The BIC function, for a range of clusters $\{1, 2, \ldots, K_{max}\}$ generates the finite mixture models and their probability values. Moreover, for the range of cluster sets with different cluster numbers, several types of mixture models are generated. The mixture models considered are Gaussian distribution functions. Finally, the actual cluster number K and finite mixture model type is set based on the results of a probability function resulting in the highest BIC value for the given set of critical path delay data. The sections below give more details on the Expectation-Maximization algorithm.

5.3.1 Finite Mixture Densities

As explained earlier, based on the highest BIC value, the finite mixture model type and number of clusters K is set in the EM algorithm. Equation 5.8 represents the finite mixture model which are a family of probability density functions:

$$f\left(\mathbf{d}'_{CP_k}(\mathbf{p}_{PVT}), \mathbf{pb}, \boldsymbol{\theta}\right) = \sum_{j=1}^{K} pb_j \cdot g_j\left(\mathbf{d}'_{CP_k}(\mathbf{p}_{PVT}); \boldsymbol{\theta}_j\right) \qquad k \in \{1, \ldots, n_{CP}\} \tag{5.8}$$

$\mathbf{d}'_{CP_k}(\mathbf{p}_{PVT})$ is the normalized delay of a critical path k across PVT condition, $\mathbf{pb}$ is the vector of the mixing proportions of each critical path into various clusters j. g_j is the density function with parameters $\boldsymbol{\theta}_j$ for each cluster j. The mixing probabilities are non-negative and $\sum_{j=1}^{K} pb_j = 1$. From the estimation of the parameters of the assumed finite mixture model and density functions, the critical path delays can be associated with specific clusters based on the maximum value of estimated posterior probability [66] [70]:

$$\mathbf{P}\left(j|\mathbf{d}'_{CP_k}(\mathbf{p}_{PVT})\right) = \frac{\mathbf{pb} \cdot g_j\left(\mathbf{d}'_{CP_k}(\mathbf{p}_{PVT}); \boldsymbol{\theta}_j\right)}{f\left(\mathbf{d}'_{CP_k}(\mathbf{p}_{PVT}), \mathbf{pb}, \boldsymbol{\theta}\right)} \qquad j \in 1, \ldots, K \qquad k \in 1, \ldots, n_{CP} \tag{5.9}$$

5.3.2 Maximum Likelihood Estimation of Finite Mixture Models of Critical Paths Delays

The finite mixture density of n_{CP} critical paths is given in Equation 5.8. The log-likelihood function l of the critical path delays is given as:

$$l\left(\mathbf{pb}, \boldsymbol{\theta}\right) = \sum_{k=1}^{n_{CP}} \ln f\left(\mathbf{d}^{'}_{CP_k}(\mathbf{p}_{PVT}), \mathbf{pb}, \boldsymbol{\theta}\right) \\ k \in 1, \ldots, n_{CP} \tag{5.10}$$

In the case of finite mixture densities, the maximum likelihood function is solved using iterative methods. For a finite mixture model of Gaussian distributions, the properties of mean $\boldsymbol{\mu}_j$ and covariance matrix of $\boldsymbol{\Sigma}_j$ of each cluster j is computed iteratively as shown below [66]:

$$\mathbf{p}_j = \frac{1}{n_{CP}} \sum_{k=1}^{n_{CP}} \mathbf{P}\left(j|\mathbf{d}^{'}_{CP_k}(\mathbf{p}_{PVT})\right) \tag{5.11}$$

$$\boldsymbol{\mu}_j(\mathbf{p}_{PVT}) = \frac{1}{n_{CP} \cdot \mathbf{p}_j} \sum_{i=1}^{n_{CP}} \mathbf{d}^{'}_{CP_k}(\mathbf{p}_{PVT}) \cdot \mathbf{P}\left(j|\mathbf{d}^{'}_{CP_k}(\mathbf{p}_{PVT})\right) \tag{5.12}$$

$$\boldsymbol{\Sigma}_j = \frac{1}{n_{CP}} \sum_{i=1}^{n_{CP}} \left(\mathbf{d}^{'}_{CP_k}(\mathbf{p}_{PVT}) - \boldsymbol{\mu}_j\right)\left(\mathbf{d}^{'}_{CP_k}(\mathbf{p}_{PVT}) - \boldsymbol{\mu}_j\right)^T \cdot \mathbf{P}\left(j|\mathbf{d}^{'}_{CP_k}(\mathbf{p}_{PVT})\right) \tag{5.13}$$

In this work, EM algorithm from Mclust function in R [71] is used to cluster the critical path data. Bayesian approach is used by calculating BIC function and the result of the BIC is fed into the EM algorithm, where the EM clustering algorithm generates the results of posterior probabilities of the selected finite mixture models along with the mean and covariance matrix in an iterative manner. Here, 256 critical paths are used for clustering where the delays of critical paths are characterized at 9 different conditions of PVT using STA. Figure 5.8 shows the results of BIC of critical path delay data. The different finite mixture models available in the Mclust function of R are given in Table 5.1. It can be seen that the BIC values increase initially with increase in number of clusters and reach a maximum point after which they remain nearly the same or decrease in value. In Figure 5.8, 22 clusters with model VEV results in the highest BIC value. Model VEV from [71] consists of ellipsoidal distributions of clusters with variable volume and equal shape among the clusters.

Table 5.1: Finite mixture models in Mclust in R

Abbreviation	Gaussian distribution Model
EII	Spherical, equal volume
VII	Spherical, unequal volume
EEI	Diagonal, equal volume and shape
VEI	Diagonal, varying volume, equal shape
EVI	Diagonal, equal volume, varying shape
VVI	Diagonal, varying volume and shape
EEE	Ellipsoidal, equal volume, shape, and orientation
EEV	Ellipsoidal, equal volume and equal shape
VEV	Ellipsoidal, equal shape
VVV	Ellipsoidal, varying volume, shape, and orientation

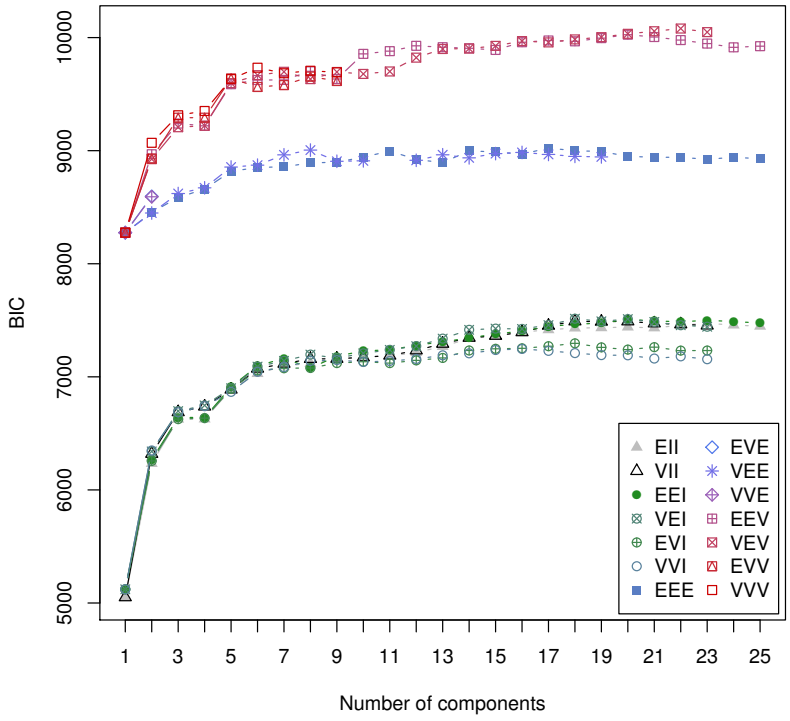


Figure 5.8: Results of BIC for critical path data

5.4 Analysis of Clustered Critical Paths

The clustered critical path data is analyzed using distances of critical path delays from their cluster centers using Equation (5.7) . The maximum distance ϵ_{max} of a critical path

Figure 5.9: Maximum distance of critical paths from hierarchical cluster center

from it's cluster is computed amongst all clusters for each PVT condition. Figures 5.9, 5.10 and 5.11 show the maximum absolute distance of critical paths from their cluster centers for a range of cluster numbers using Hierarchical, KMeans++ and EM clustering, respectively. The red line in the Figures indicate 2% distance and blue line indicates 1% distance. It can be seen that in Figure 5.11 with EM clustering, for clusters between 2 to 23, the maximum distance of critical paths from cluster center is always greater than 2%. Additionally, in Figure 5.10 with KMeans++ clustering, the maximum distance from cluster center is below 2% for 12 clusters and the maximum distance is greater than 1% until 25 clusters. Furthermore, in Figure 5.9 with hierarchical clustering, the maximum distance from cluster center is below 2% for 11 clusters and the maximum distance is lesser than 1% at 24 clusters. Therefore, Hierarchical clustering results in the least maximum distance between critical paths and cluster centers and requires fewer number of clusters. This means that with Hierarchical clustering, the number of DDROs to synthesize for a given set of critical paths is lesser in number compared to KMeans++ and EM clustering. Moreover, Hierarchical clustering also results in compact clusters which means that distance between cluster center and critical paths is the lesser than EM and KMeans++ clustering. In summary, although Hierarchical clustering is the simplest algorithm amongst the three different algorithms analyzed, it results in the most compact clusters with minimum cluster sizes and maximum distance between two clusters. Thus, Hierarchical clustering is considered as the best choice of algorithm to cluster critical paths for DDRO synthesis.

Figure 5.10: Maximum distance of critical paths from KMeans++ cluster center

Figure 5.11: Maximum distance of critical paths from EM cluster center

5.5 Summary

In this Chapter, in order to synthesize DDROs for a large number of critical paths of a real design, grouping of critical path delays across PVT conditions is explained so that one DDRO can be synthesized per critical path group. In this work, three different unsupervised clustering algorithms are investigated for the critical path clustering namely Hierarchical, KMeans++ and Expectation-Maximization (EM) clustering. Hierarchical and KMeans++ are heuristic algorithms where the number of clusters and the size of a cluster is user-defined. However, EM clustering is a model-based clustering which tries to fit the critical paths into different Gaussian distribution functions in a finite mixture model. Each Gaussian distribution is then assumed to be one cluster. Moreover, the

number of clusters and mixture model is also selected by the EM algorithm. Furthermore, in order to compare the different methods of clustering performed on a given set of critical path data, an assessment method is proposed where the distance of each of the critical paths is computed from its cluster center for each PVT condition and for all the three different clustering methods. Evaluating the size of a cluster based on maximum distance of a critical path from its cluster center, Hierarchical clustering results in the most compact clusters. Although Hierarchical and KMeans++ are simple heuristic algorithms, they emerge as good candidates to cluster critical paths to synthesize DDROs. Therefore, in the next Chapter 6, the DDROs are synthesized for critical path data clustered by Hierarchical and KMeans++ algorithms.

6 DDRO Synthesis for Large-Scale Digital Design

As explained earlier, an industrial chip design typically consists of tens of thousands of critical paths thus, making it infeasible to generate DDROs for each of these critical paths. To generate DDROs which can match the timing behavior of all critical paths, the identified critical paths are grouped based on their similarities of timing behavior and a single DDRO is synthesized for one critical path group. Thus, the number of DDROs required for a chip is the number of groups of critical paths. In this Chapter, DDROs are synthesized for clustered critical paths where the critical paths are clustered using algorithms explained in Chapter 5.

In this Chapter, tile delays and critical path delays are characterized using static timing analysis (STA). STA is substantially less computationally intensive than SPICE simulation. Therefore, using STA makes it feasible to characterize delays of thousands of critical paths and tiles. The number of critical paths used for clustering is increased from 96 to 256 to be more representative of an industrial design. Moreover, the selected critical paths are extracted from a physical design placed and routed by commercial tools including parasitic RC elements. Additionally, it was found that, Kmeans++ and Hierarchical clustering are better suited to synthesize DDROs in Chapter 5. Therefore, critical path delays have been grouped using KMeans++ and Hierarchical Clustering algorithm into 13 distinct clusters. The centroid value of a cluster is considered as the target to synthesize a corresponding DDRO. The parameters used for the DDRO synthesis for large-scale digital design is given in Table 6.1.

Delay-tracking-based DDROs are synthesized using two different tile library sets. The first tile library set consists of 6 tile types and the second tile library contains two additional tile types. These two additional tile types in the second tile library are identified based on the evaluation of the timing behavior of chosen critical paths. The tile delays and critical path delays are characterized at 9 different PVT corners which include the process slow, nominal and fast corners. The DDROs are synthesized using quadratic 3-tile delay characterization (QT3) and Multimodal-Successive optimization method. The length of a tile is set to 3. A tile block length of $n_s = 5$ is used for the DDRO synthesis. The number of selected solutions in each iteration of the Multimodal-Successive optimization $\beta = 20$.

Table 6.1: Experimental parameters the DDRO synthesis for large-scale digital design

Parameter	Value(s)
DDRO objectives	Quadratic
Tile delay characterization	T3
Tile types	INV, NAND2, NOR2, 2 additional tile types
Number of drive strengths of tile types	2
Number of tile types n_φ	6, 8
Tile length	3
Tile block length n_s	5
Solver	Multimodal-Successive optimization
Number of PVT corners	9
Delay data	STA
Target	Clustered critical paths
Number of clusters	13
Clustering algorithms	Hierarchical, KMeans++

6.1 DDRO Delays versus Critical Path Cluster Centers

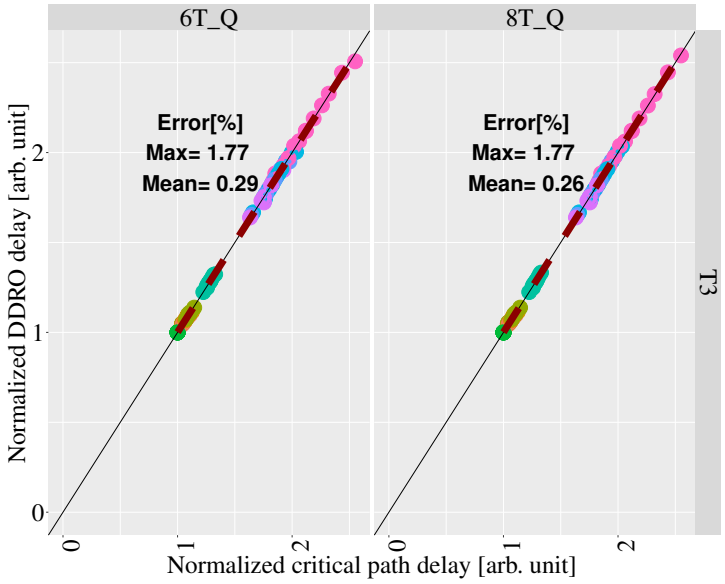


Figure 6.1: Delay-based DDRO synthesis using STA data with KMeans++ clustering

Figures 6.1 and 6.2 show the correlation between the normalized delay of DDROs and their respective critical paths clusters using KMeans++ and Hierarchical clustering,

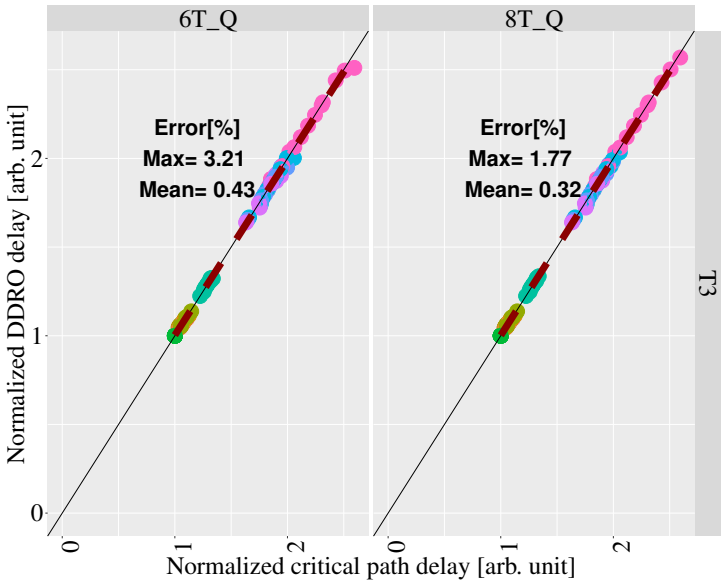


Figure 6.2: Delay-based DDRO synthesis using STA data with Hierarchical clustering

respectively. The delays are normalized as per Equation (3.35). The different colors in Figures 6.1 and 6.2 indicate the PVT corners. A 45 degree solid line represents the ideal regression line plotted as a reference.

It can be seen that the mean error in percentage across all PVT corners with KMeans++ clustered data is 0.28% and the maximum error is 1.77% and does not show any improvement with the selection of 6 or 8 tile types in the tile library. The DDRO synthesis with Hierarchical clustered data results in a slightly larger mean than KMeans++. The mean error reduces from 0.43% to 0.32% and the maximum error reduces from 3.21% to 1.77% for 6 to 8 tile types for Hierarchical clustered data. Thus, showing an improvement with the selection of 8 tile types in comparison to 6 tile types in the tile library. Additionally, maximum and mean error between DDRO and critical paths is lower for clustered data in comparison to non-clustered data in Figure 4.5 having a mean and maximum error of 4.31% and 0.64%, respectively. This proves that the DDRO synthesis flow is suitable for the clustered STA based data that includes critical paths with parasitic RC elements.

In order to evaluate the distribution of delay errors, Figures 6.3 and 6.4 show the distribution of DDRO mismatch with respect to the reference critical paths cluster centers, with 8 tile types per standard cell library for KMeans++ and Hierarchical clustering, respectively. The different bars in the figures represent bins for different delay mismatch values. The height gives the number of DDROs in the respective bin, the colors distinguish between the different critical paths. It is observed that the distribution is skewed with error points tending towards the origin. Furthermore, with both the clustering methods and for each PVT corner, 90% of DDROs have less than 1% error.

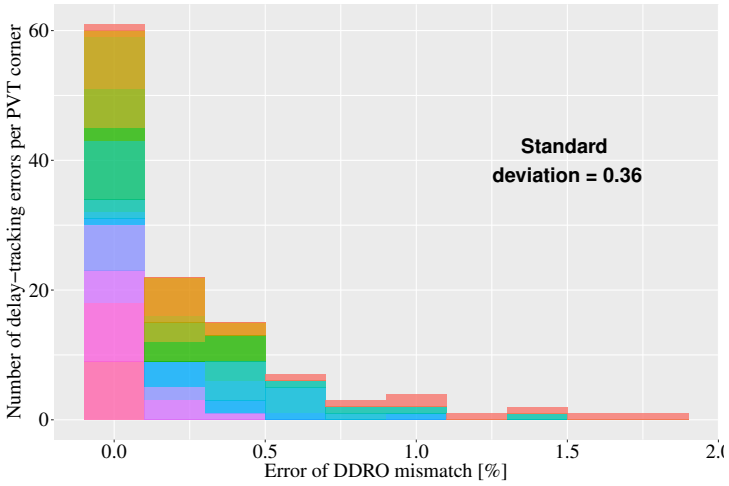


Figure 6.3: Distribution of delay-tracking errors of DDROs per corner with KMeans++ clustering

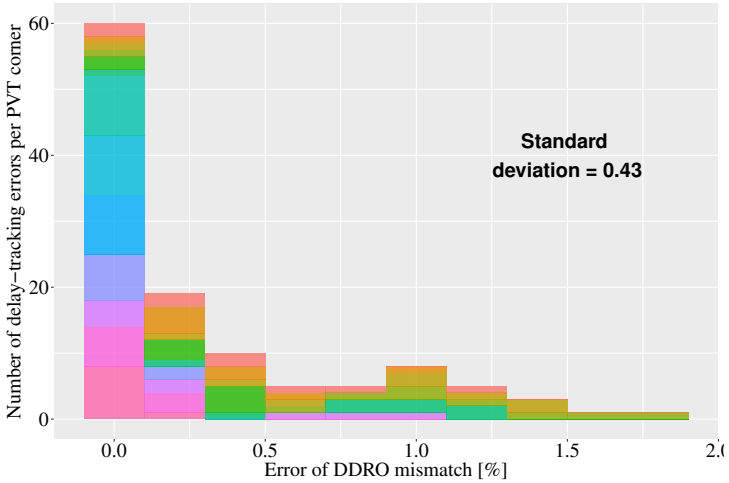


Figure 6.4: Distribution of delay-tracking errors of DDROs per corner with Hierarchical clustering

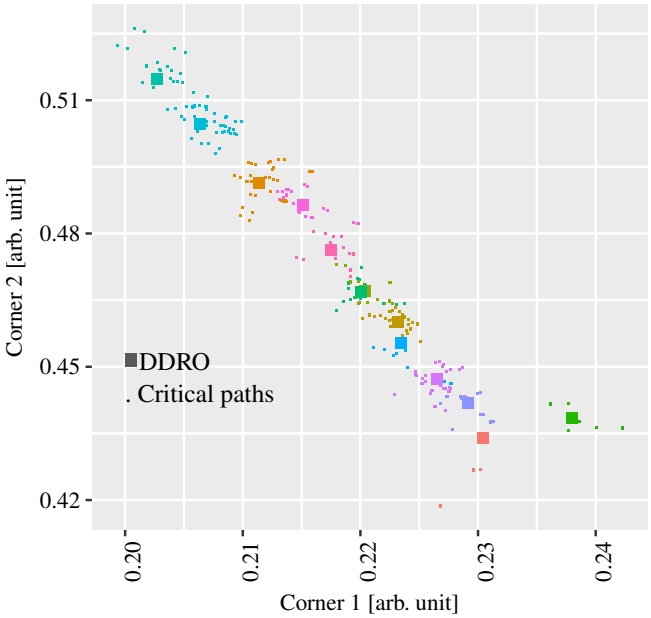


Figure 6.5: Scatter plot of DDROs synthesized for clustered critical paths with KMeans++ clustering

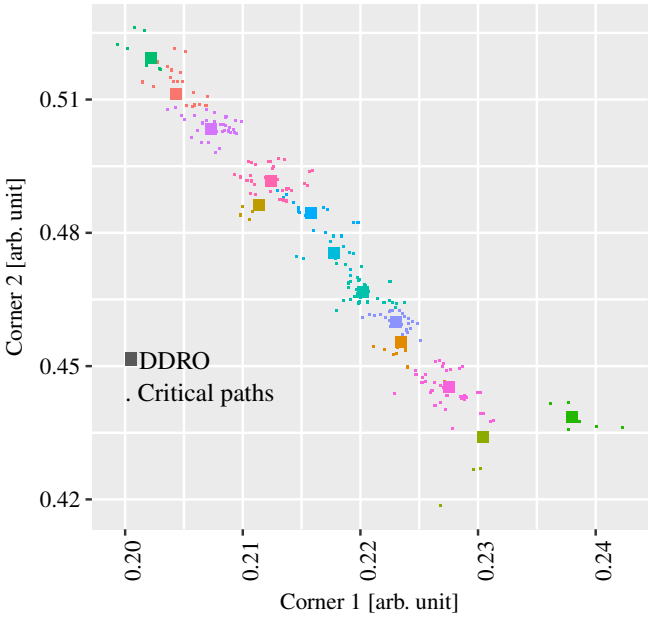


Figure 6.6: Scatter plot of DDROs synthesized for clustered critical paths with Hierarchical clustering

6.2 DDRO Delays versus Critical Path Delays

The earlier analysis looked into the mean and maximum error of DDROs with respect to the cluster center of the clustered critical paths. In order to evaluate if the DDROs are located close to the center of their respective critical path cluster, Figures 6.5 and 6.6 shows the scatter plots of the normalized delays for each DDRO and critical path for two PVT corners (corner 1 and corner 2) for DDROs synthesized using KMeans++ and Hierarchical clustering, respectively. Each color variant is a specific cluster. The smaller dots represent the critical paths for each cluster. The squares represent the DDROs for the respective cluster. It can be observed that the DDROs are well centered within their respective cluster and therefore are reliable representatives of their clusters.

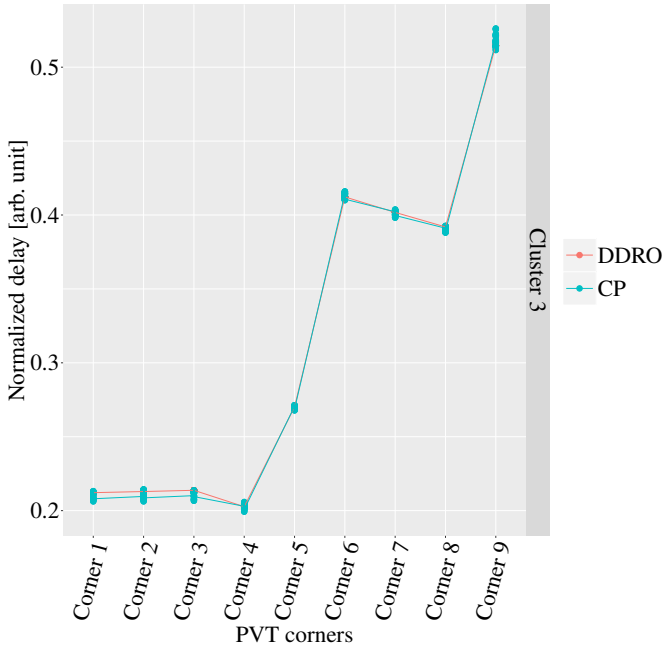


Figure 6.7: Delay tracking of synthesized DDROs for cluster-3 of critical paths

Furthermore, Figures 6.7 and 6.8 shows the delay tracking of syntheized DDROs for two clusters of critical paths. The DDROs are synthesized for KMeans++ clustered data. These plots show the ability of a DDRO to track the critical path delays across all 9 PVT corners. It can be seen from Figures 6.7 and 6.8 that, both critical path clusters differ in their timing behavior across PVT corners and the synthesized DDROs are able to track their respective clusters.

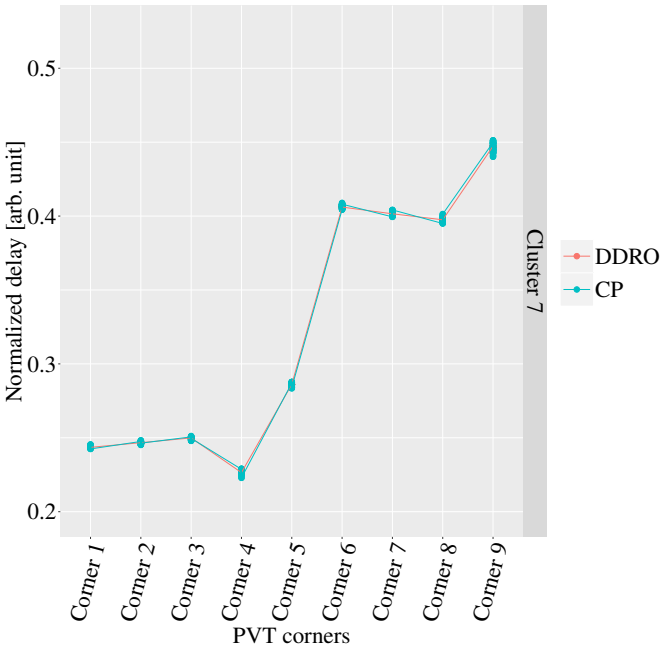


Figure 6.8: Delay tracking of synthesized DDROs for cluster-7 of critical paths

6.3 Summary

In this Chapter, DDROs are synthesized for clustered critical paths to demonstrate the DDRO synthesis for large scale digital design. Thereby, 256 critical paths are identified and grouped using KMeans++ and Hierarchical clustering into 13 different clusters. The DDROs are synthesized using the best optimization method as observed in Chapter 4 which is the delay-tracking-based DDRO synthesis using quadratic objective and 3-tile delay characterization (QT3).

It is found that KMeans++ and Hierarchical clustering result in a maximum error of 1.77% using 8 tile types in the tile library. Moreover, DDROs are plotted amongst the critical paths in a scatter plot between two PVT corners. It was found that the DDROs are well within their respective target critical path group. Furthermore, the DDROs delay-tracking capability across various PVT conditions is analyzed where, it is observed that the DDROs are able to track the delays of their respective clusters across all the PVT conditions.

Thus, the delay-tracking-based DDRO synthesis is a systematic and accurate method to monitor the critical path timing of digital circuits. Furthermore, due to grouping of critical paths and synthesizing one DDRO per group of critical paths, it is feasible to construct DDROs for a complete digital design in large scale industrial chips.

7 Summary and Outlook

The advancement of technology nodes has resulted in increased variations of digital circuit timing. Timing variations in worst-possible conditions could lead to errors and breakdown of electronic systems. Therefore, addition of guardbands during the chip design is a solution to avoid failure. These guardbands ensure that the required performance is met. However, result in elevated area, power, cost and design time. In order to verify if timing criteria are met by manufactured chips, post-silicon chip validation is performed. During this, additional test guardbands are added which is calculated based on the accuracy of chip timing prediction. To reduce test guardbands and increase manufacturing yield, frequency is evaluated by on-chip test structures where their timing behavior is highly correlated to the behavior of the chip under given fabricated conditions, and across various voltage and temperature conditions.

Design-dependent timing monitors introduced in [22] [23] [24] are used to evaluate the timing of specific chip design. In this thesis, design dependent ring oscillators (DDROs) are the proposed timing monitors. In the DDRO approach critical paths are clustered based on their similarities in timing behavior across PVT conditions. For each path group, a DDRO consisting of a chain of tiles is constructed, where a tile is a chain of standard cells of equal type. The goal of the DDRO synthesis is to match the timing behavior of the target critical path group.

This thesis focuses on improving the quality of design-dependent ring oscillators (DDROs) by using delay-tracking for construction of DDROs in contrast to the sensitivity tracking as explained in Chapter 3. In addition to the linear objective introduced in the state-of-the-art method, quadratic objective is proposed in this thesis to further increase the accuracies of DDROs. This thesis also proposes tile delay characterization with realistic environment considering the impact of input slope and output load on tile delay. Thereby, three new methods of tile delay characterizations are investigated out of which the 3-tile delay characterization results in the most accurate but complex method of tile delay characterization. Furthermore, to solve larger DDRO synthesis optimization problems, 3 new Heuristic iterative algorithms are proposed, namely, Single-Mode Successive, Multimodal-Successive and Single-Mode Back-Tracking optimization. Additionally, DDRO assessment methods are developed to evaluate the quality of synthesized DDROs.

Chapter 4 demonstrates the experimental results of the synthesized DDROs. Delay-tracking-based and sensitivity based DDROs are constructed for an industrial library below 40nm and their quality as well as run-time behavior is evaluated. Delay-tracking-based DDROs are compared to the existing state-of-the-art sensitivity-based DDROs where the maximum error reduces approximately from 4% to 2% for sensitivity-based to delay-tracking-based DDROs, respectively based on SPICE evaluation. Additionally, the

quadratic objective for delay-tracking-based DDROs are evaluated for different tile delay characterization methods where the maximum error reduces from 8.39% to 1.98% from the state-of-the-art method of 1-tile delay to 3-tile delay characterization, respectively. Therefore, among many different DDRO synthesis methods investigated, delay-tracking-based DDROs with quadratic objective and 3-tile delay characterization emerges as the most preferred DDRO synthesis method.

Furthermore, a tile library consisting of various tile types is selected for DDRO synthesis. Studies show that DDRO synthesis using direct solvers using 6 tile types in the tile library, quadratic objective and 3-tile delay characterization have infeasible CPU run-times. On the other hand, DDRO synthesis using the developed heuristic method of Multimodal-Successive optimization requires merely 30 seconds which makes it feasible for larger DDRO optimization problems. Thus, the developed heuristic synthesis methods are computationally much less intensive than traditionally used direct solvers.

In this thesis, three different unsupervised clustering algorithms are investigated for the critical path clustering namely Hierarchical, KMeans++ and Expectation Maximization (EM) clustering so that a single DDRO can be synthesized for a cluster of critical paths. To compare different clustering methods, an assessment method is proposed for measuring the cluster size. The distance of each of the critical paths is computed from its cluster center for each PVT condition. Assuming the size of a cluster as the maximum distance of critical paths from its cluster center, the clustering algorithms were compared for the smallest cluster size. The Hierarchical clustering results in the most compact clusters, followed by KMeans++ clustering. Although Hierarchical and KMeans++ are Heuristic clustering algorithms with low computational complexity, they emerge as good candidates in comparison to EM clustering to group critical paths.

The experimental results for the DDRO synthesis targeting clustered critical paths where clustering is performed using KMeans++ and Hierarchical clustering, are explained in Chapter 6. The DDRO synthesis method resulting in the most accurate DDROs in Chapter 4 is used, where the delays are computed using STA. Evaluation of these DDROs results in the maximum error between target cluster centers and DDROs of 1.77% for both KMeans++ and Hierarchical clustering. Furthermore, the DDROs delay-tracking capability across various PVT conditions is analyzed and the DDROs can accurately track the delays of their respective clusters across all given PVT conditions.

Thus, this thesis proposes new mathematical formulations to synthesize design dependent ring oscillators (DDROs) which increases the accuracy of existing DDROs. Moreover, Heuristic algorithms developed result in reduced computational complexity to synthesize DDROs. STA characterized delays for DDRO synthesis is used which further reduces the computational complexity of delay computation. Additionally, different clustering algorithms are investigated to group critical paths such that one DDRO is synthesized for one critical path group to synthesize DDROs for large scale digital designs.

The outlook of this thesis is manifold. At first, the DDROs have to be developed with more advanced optimization algorithms, which can include more tile types in the tile library. Moreover, the DDRO synthesis method should be further reformulated to

include RC extraction such that the DDROs can track not only the delay across PVT but also the extraction (X) parameters. The frequency measured from the DDROs shall indicate the timing of the design critical paths, which enables the tuning of the chip's parameters such as voltage and frequency to increase the efficiency of the chip.

A List of Symbols

$A(it)$	The subset of tile blocks selected at each iteration from the enumerated set of tile blocks $S^{(X)}(it)$
$\alpha(it)$	Ratio of DDRO objective values of two adjacent iterations used in the Single-Mode Back-Tracking optimization
α_{max}	User-defined maximum allowed ratio of the DDRO objective between two different iterations in the Single-Mode Back-Tracking optimization
β	User defined number of solutions selected at each iteration it in the Multimodal-Successive optimization
$\mathbf{b}$	Vector notation for the critical path delay sensitivities used in the direct solver
$C(i-1,i)$	Constraint describing the relationship between two adjacent tile positions for 2-tile and 3-tile delay characterizations
$\delta\mathbf{p}$	Vector of PVT parameter variations
$\nabla d_{CP}(\mathbf{p}_0)$	Delay sensitivities of critical path at nominal PVT values
$\nabla d'_{CP}(\mathbf{p}_0)$	Normalized delay sensitivities of critical path at nominal PVT conditions
$\nabla d_{CP}(\mathbf{p}_{PVT})$	Delay sensitivities of critical path across PVT conditions
$\nabla d'_{CP}(\mathbf{p}_{PVT})$	Normalized delay sensitivities of critical path across PVT conditions
$\mathbf{d}_{CP}(\mathbf{p}_{PVT})$	Delay vector of critical path across PVT conditions
$\mathbf{d}'_{CP}(\mathbf{p}_{PVT})$	Normalized delay of critical path across PVT conditions
$\frac{d_i}{d_t}$	Current derivative in digital circuit

$\nabla d_i^{(X)}(\mathbf{p}_0)$	Delay sensitivities of tiles at nominal PVT values
$\nabla d_i^{(X)}(\mathbf{p}_{PVT})$	Delay sensitivities of tiles across PVT conditions
$\mathbf{d}_i^{(X)}(\mathbf{p}_{PVT})$	Delay vector of tiles across PVT conditions
$\nabla d_i^{'(X)}(\mathbf{p}_0)$	Normalized delay sensitivities of tiles at nominal PVT values
$\mathbf{d}_i^{'(X)}(\mathbf{p}_{PVT})$	Normalized delay vector of tiles across PVT conditions
$\nabla d_Y(\mathbf{p}_0)$	Delay sensitivities of a circuit at nominal PVT conditions
$\nabla d^{'(X)}_{\varphi_1^{(X)}(i)}(\mathbf{p}_{PVT})$	Delay sensitivities of a tile type $\varphi_1(i)$ at position i across PVT conditions
d_{er}	l_2 norm error between critical path and DDRO delay across PVT conditions
$\mathbf{p}_{PVT}$	Vector of PVT parameters
$\mathbf{p}_0$	Nominal vector value of PVT parameters
$\varphi(i)$	Type of tile at position i
K	Number of clusters of critical paths
$K(st)$	Number of clusters of critical paths at step st
$\Delta\mathbf{d}_{PE}(\mathbf{p}_{PVT})$	Delay error in percentage calculated between normalized delays of critical path and DDRO across PVT conditions
$\Delta\mathbf{d}_{DDRO}$	Minimum objective value at iteration it by the Multimodal-Successive optimization
$\mathbf{d}_{TB}^{(X)}$	Tile block delay of a tile block from enumeration set $S^{(X)}(it)$ across PVT conditions
$\mathbf{d}_{TB}^{(X)}(S^{(X)}, \mathbf{p}_{PVT})$	Delay of tile block where the tile block is enumerated at iteration it
$\mathbf{d}_{DDRO}^{(X)}$	DDRO delay across PVT conditions

$d_Y(\mathbf{p}_0)$	Propagation delay of a circuit at nominal PVT values
$d_i^{(T)}(\mathbf{p})$	Delay of a tile with 1-tile delay characterization at a PVT corner
$d_i^{(TP2)}(\mathbf{p})$	Delay of a tile with 2-tile delay characterization with preceding tile at a PVT corner
$d_i^{(TS2)}(\mathbf{p})$	Delay of a tile with 1-tile delay characterization with suceeding tile at a PVT corner
$d_i^{(T3)}(\mathbf{p})$	Delay of a tile with 1-tile delay characterization at a PVT corner
$d'_{CP_{kj}}(st, \mathbf{p}_{PVT})$	Normalized critical path delay of index k in cluster j at step st
$\boldsymbol{\epsilon}_{kj}(\mathbf{p}_{PVT})$	Distance of each critical path from its respective cluster center
ϵ_{max}	User defined maximum permissible distance of critical paths from their respective cluster center per PVT corner
g_j	Density function with parameters $\boldsymbol{\theta}_j$ for cluster j
$i(t)$	Current flow through the parasitic capacitance of the power grid and is dependent on the time t
i	Index of the tile position
it	Iteration index in the DDRO synthesis using Heuristic iteration method
$\boldsymbol{\mu}_j(\mathbf{p}_{PVT})$	Critical path cluster center j, where critical paths are characterized across PVT corners
$L_{parasitic}$	Parasitic inductance of digital circuit
$l_{\mathbf{pb},\boldsymbol{\theta}}$	Log likelihood function of critical paths based on mixing proportions and density functions
$n_c^{(X)}(it)$	The number of possible tile combinations of iteration it

n_s	Length of a tile block
n_φ	Number of tiles types for the DDRO synthesis
n_{cp}	The total number of target critical paths
n_T	Number of characterized delay sensitivities of 1-tile/2-tile/3-tile delay
n_{it}	Number of iterations for the heuristic algorithm
n_t	Number of tiles in DDRO
$n_{t_{max}}, n_{t_{min}}$	User defined maximum and minimum permissible length of DDRO (number of tiles)
n_{tb}	Number of tile blocks in the DDRO synthesis
n_{PVT}	Number of PVT conditions used for the DDRO synthesis
$pn = 1, 2$	Dimension of norm (using l_1 or l_2 norm) to find the minimum of the objective function for sensitivity-based and delay-tracking-based DDROs
$\mathbf{\Delta p}$	Small variations of PVT parameters $\mathbf{p}$
$\mathbf{p}$	Vector of PVT parameters
$\mathbf{pb}$	Vector of mixing proportions of each critical path into various clusters
$P(j/\mathbf{d}'_{CP_k}(\mathbf{p}_{PVT})$	Posterior probability of certain critical path k belonging to cluster j
R_{grid}	Parasitic capacitance of the power grid
$S^{(X)}(it)$	Enumeration set of tile blocks with length n_s at iteration it
$S_{CP_j}(st)$	Group of critical paths j at step st
t_{gate}	Delay of a CMOS logic gate

$T(it-1, it)$	Unique mapping function which results in β different DDROs at the end of DDRO optimization using mulitmodal-successive optimization
$\mathbf{TM}_i^{(X)}$	Matrix of normalized tile delays of all selected and characterized tiles for a certain tile position i used in the matrix notation for the direct solver
θ	Technology dependent parameter to calculate the delay of CMOS logic gate
ΔV_{IRdrop}	Voltage variation induced due to IR drop
$\Delta V_{\frac{di}{dt}}$	Voltage variation due to current derivative noise
V_{DD}	Supply voltage of digital circuits
V_{TH}	Threshold voltage of transistors
X	$X \in \{T1, TP2, TS2, T3\}$ method of characterization of tiles
$\mathbf{x}$	Unknown column vector which describes the tile type for each position of the tile used in the DDRO synthesis by direct solver
$\mathbf{Z}$	Matrix of all the collected delay sensitivities of tile types for all tile positions actoss PVT conditions to be fed to the direct solver for the DDRO synthesis

B List of Abbreviations

BIC	Bayesian Inversion Criterion.
BOEL	Back End of Line.
DDRO	Design-Dependent Ring Oscillator.
EM	Expectation-Maximization.
GRO	Generic Ring Oscillator.
ITE	Inverted Temperature Effect.
L	Linear Optimization.
LER	Line Edge Roughness.
NOM	Nominal.
O(N)	Order of N.
OPC	Optical Proximity Correction.
OPE	Optical Proximity Effect.
PSRO	Process Specifc Ring Oscillator.
PVT	Process Voltage Temperature.
Q	Quadratic Optimization.
RC	Resistance-Capacitance.
RDF	Random Dopant Fluctuation.
RO	Ring Oscillator.

SL	Input-slew and Output-load.
STA	Static Timing Analysis.
T1	1-Tile Delay Characterization.
T3	3-Tile Delay Characterization.
TP2	Preceding 2-Tile Delay Characterization.
TRC	Tunable Replica Circuits.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] N. R. Storey. *Safety Critical Computer Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

[2] M. Wirnshofer. *Variation-Aware Adaptive Voltage Scaling for Digital CMOS Circuits*. Springer Publishing Company, Incorporated, 2013.

[3] Y. Higuchi, K. Shinkai, M. Hashimoto, R. Rao, and S. Nassif. Extracting device-parameter variations using a single sensitivity-configurable ring oscillator. In *2013 18th IEEE European Test Symposium (ETS)*, pages 1–6, May 2013. doi:10.1109/ETS.2013.6569366.

[4] P. S. Zuchowski, P. A. Habitz, J. D. Hayes, and J. H. Oppold. Process and environmental variation impacts on ASIC timing. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pages 336–342, Nov 2004. doi:10.1109/ICCAD.2004.1382597.

[5] E. H. Cannon, A. KleinOsowski, R. Kanj, D. D. Reinhardt, and R. V. Joshi. The Impact of Aging Effects and Manufacturing Variation on SRAM Soft-Error Rate. *IEEE Transactions on Device and Materials Reliability*, 8(1):145–152, March 2008. doi:10.1109/TDMR.2007.912983.

[6] I. S. Esqueda, H. J. Barnaby, and M. P. King. Compact Modeling of Total Ionizing Dose and Aging Effects in MOS Technologies. *IEEE Transactions on Nuclear Science*, 62(4):1501–1515, Aug 2015. doi:10.1109/TNS.2015.2414426.

[7] B. Nikolic and L. Pang. Measurements and analysis of process variability in 90nm CMOS. In *2006 8th International Conference on Solid-State and Integrated Circuit Technology Proceedings*, pages 505–508, Oct 2006. doi:10.1109/ICSICT.2006.306337.

[8] I. A. K. M. Mahfuzul, A. Tsuchiya, K. Kobayashi, and H. Onodera. Variation-sensitive monitor circuits for estimation of Die-to-Die process variation. In *2011 IEEE ICMTS International Conference on Microelectronic Test Structures*, pages 153–157, April 2011. doi:10.1109/ICMTS.2011.5976878.

[9] S. Reda and S. R. Nassif. Accurate Spatial Estimation and Decomposition Techniques for Variability Characterization. *IEEE Transactions on Semiconductor Manufacturing*, 23(3):345–357, Aug 2010. doi:10.1109/TSM.2010.2051752.

[10] M. Bhushan, M. B. Ketchen, S. Polonsky, and A. Gattiker. Ring oscillator based technique for measuring variability statistics. In *2006 IEEE International Conference on Microelectronic Test Structures*, pages 87–92, March 2006. doi:10.1109/ICMTS.2006.1614281.

[11] T. Chan, A. Pant, L. Cheng, and P. Gupta. Design-Dependent Process Monitoring for Wafer Manufacturing and Test Cost Reduction. *IEEE Transactions on Semiconductor Manufacturing*, 25(3):447–459, Aug 2012. doi:10.1109/TSM.2012.2196709.

[12] C. L. L. Soon Jyh Chang and J. E. Chen. Functional test pattern generation for CMOS operational amplifier. In *Proceedings. 15th IEEE VLSI Test Symposium (Cat. No.97TB100125)*, pages 267–272, April 1997. doi:10.1109/VTEST.1997.600287.

[13] M. Bhushan, A. Gattiker, M. B. Ketchen, and K. K. Das. Ring oscillators for CMOS process tuning and variability control. *IEEE Transactions on Semiconductor Manufacturing*, 19(1):10–18, Feb 2006. doi:10.1109/TSM.2005.863244.

[14] S. Reda and S. R. Nassif. Analyzing the impact of process variations on parametric measurements: Novel models and applications. In *2009 Design, Automation Test in Europe Conference Exhibition*, pages 375–380, April 2009. doi:10.1109/DATE.2009.5090692.

[15] L. T. . Wang, N. Xu, S. . Toh, A. R. Neureuther, T. . K. Liu, and B. Nikolic. Parameter-specific ring oscillator for process monitoring at the 45 nm node. In *IEEE Custom Integrated Circuits Conference 2010*, pages 1–4, Sept 2010. doi:10.1109/CICC.2010.5617624.

[16] A. Gattiker, M. Bhushan, and M. B. Ketchen. Data Analysis Techniques for CMOS Technology Characterization and Product Impact Assessment. In *2006 IEEE International Test Conference*, pages 1–10, Oct 2006. doi:10.1109/TEST.2006.297743.

[17] K. K. Das, S. G. Walker, and M. Bhushan. An Integrated CAD Methodology for Evaluating MOSFET and Parasitic Extraction Models and Variability. *Proceedings of the IEEE*, 95(3):670–687, March 2007. doi:10.1109/JPROC.2006.890091.

[18] D. Blaauw, S. Kalaiselvan, K. Lai, W. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull. Razor II: In-Situ Error Detection and Correction for PVT and SER Tolerance. In *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 400–622, Feb 2008. doi:10.1109/ISSCC.2008.4523226.

[19] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pages 7–18, Dec 2003. doi:10.1109/MICRO.2003.1253179.

[20] Q. Liu and S. S. Sapatnekar. Capturing Post-Silicon Variations Using a Representative Critical Path. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(2):211–222, Feb 2010. doi:10.1109/TCAD.2009.2035552.

[21] J. Tschanz, K. Bowman, S. Walstra, M. Agostinelli, T. Karnik, and V. De. Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance. In *2009 Symposium on VLSI Circuits*, pages 112–113, June 2009.

[22] T. Chan, P. Gupta, A. B. Kahng, and L. Lai. DDRO: A novel performance monitoring methodology based on design-dependent ring oscillators. In *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, pages 633–640, March 2012. doi:10.1109/ISQED.2012.6187559.

[23] L. Lai and P. Gupta. Accurate and inexpensive performance monitoring for variability-aware systems. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 467–473, Jan 2014. doi:10.1109/ASPDAC.2014.6742935.

[24] T. Chan, P. Gupta, A. B. Kahng, and L. Lai. Synthesis and Analysis of Design-Dependent Ring Oscillator (DDRO) Performance Monitors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(10):2117–2130, Oct 2014. doi:10.1109/TVLSI.2013.2282742.

[25] J. K. Rangan, N. P. Aryan, L. Wang, J. Bargfrede, C. Funke, and H. Graeb. Design-dependent Monitors Based on Delay Sensitivity Tracking. In *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 633–636, Dec 2018. doi:10.1109/ICECS.2018.8617873.

[26] J. K. Rangan, N. P. Aryan, L. Wang, J. Bargfrede, C. Funke, and H. Graeb. Synthesis of DDRO Timing Monitors by Delay-Tracking and Static Timing Analysis. *Submitted to IEEE Transactions on Circuits and Systems I*, January 2020.

[27] T. Liu, C. Chen, and L. Milor. Comprehensive Reliability-Aware Statistical Timing Analysis Using a Unified Gate-Delay Model for Microprocessors. *IEEE Transactions on Emerging Topics in Computing*, 6(2):219–232, April 2018. doi:10.1109/TETC.2016.2588724.

[28] S. Nassif, K. Bernstein, D. J. Frank, A. Gattiker, W. Haensch, B. L. Ji, E. Nowak, D. Pearson, and N. J. Rohrer. High Performance CMOS Variability in the 65nm Regime and Beyond. In *2007 IEEE International Electron Devices Meeting*, pages 569–571, Dec 2007. doi:10.1109/IEDM.2007.4419002.

[29] V. Champac and J. Gervacio. *Timing Performance of Nanometer Digital Circuits Under Process Variations*. Frontiers in Electronic Testing. Springer International Publishing, 2018.

[30] M. Madou. *Fundamentals of Microfabrication and Nanotechnology, Third Edition, Three-Volume Set*. Taylor & Francis, 2011.

[31] M. Dietrich and J. Haase. *Process Variations and Probabilistic Integrated Circuit Design*. Springer Publishing Company, Incorporated, 2011.

[32] S. K. Saha. Modeling Process Variability in Scaled CMOS Technology. *IEEE Design Test of Computers*, 27(2):8–16, March 2010. doi:10.1109/MDT.2010.50.

[33] P. Bai, C. Auth, S. Balakrishnan, M. Bost, R. Brain, V. Chikarmane, R. Heussner, M. Hussein, J. Hwang, D. Ingerly, R. James, J. Jeong, C. Kenyon, E. Lee, S. . Lee, N. Lindert, M. Liu, Z. Ma, T. Marieb, A. Murthy, R. Nagisetty, S. Natarajan, J. Neirynck, A. Ott, C. Parker, J. Sebastian, R. Shaheed, S. Sivakumar, J. Steigerwald, S. Tyagi, C. Weber, B. Woolery, A. Yeoh, K. Zhang, and M. Bohr. A 65nm logic technology featuring 35nm gate lengths, enhanced channel strain, 8 Cu interconnect layers, low-k ILD and 0.57 /spl mu/m/sup 2/ SRAM cell. In *IEDM Technical Digest. IEEE International Electron Devices Meeting, 2004.*, pages 657–660, Dec 2004. doi:10.1109/IEDM.2004.1419253.

[34] C. H. Diaz, H.-J. Tao, Y.-C. Ku, A. Yen, and K. Young. An experimentally validated analytical model for gate line-edge roughness (LER) effects on technology scaling. *IEEE Electron Device Letters*, 22(6):287–289, June 2001. doi:10.1109/55.924844.

[35] A. Asenov, S. Kaya, and A. R. Brown. Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness. *IEEE Transactions on Electron Devices*, 50(5):1254–1260, May 2003. doi:10.1109/TED.2003.813457.

[36] K. Nayak, S. Agarwal, M. Bajaj, K. V. R. M. Murali, and V. R. Rao. Random Dopant Fluctuation Induced Variability in Undoped Channel Si Gate all Around Nanowire n-MOSFET. *IEEE Transactions on Electron Devices*, 62(2):685–688, Feb 2015. doi:10.1109/TED.2014.2383352.

[37] D. J. Frank, Y. Taur, M. Ieong, and H. . P. Wong. Monte Carlo modeling of threshold variation due to dopant fluctuations. In *1999 Symposium on VLSI Technology. Digest of Technical Papers (IEEE Cat. No.99CH36325)*, pages 169–170, June 1999. doi:10.1109/VLSIT.1999.799397.

[38] A. Asenov, S. Kaya, and J. H. Davies. Intrinsic threshold voltage fluctuations in decanano MOSFETs due to local oxide thickness variations. *IEEE Transactions on Electron Devices*, 49(1):112–119, Jan 2002. doi:10.1109/16.974757.

[39] M. Altieri scarpato. *Digital circuit performance estimation under PVT and aging effects*. Theses, Université Grenoble Alpes, December 2017. URL: https://tel.archives-ouvertes.fr/tel-01773745.

[40] N. P. Aryan. *Monitoring Concepts for Degradation Effects in Digital CMOS Circuits*. PhD thesis, Technical University of Munich, July 2015. An optional note.

[41] W. Liu, X. Jin, J. Chen, M.-C. Jeng, Z. Liu, Y. Cheng, K. Chen, M. Chan, K. Hui, J. Huang, R. Tu, P. Ko, and C. Hu. BSIM 3v3.2 MOSFET Model Users' Manual. Technical Report UCB/ERL M98/51, EECS Department, University of California, Berkeley, 1998. URL: `http://www2.eecs.berkeley.edu/Pubs/TechRpts/1998/3486.html`.

[42] C. Roth, L. John, and B. Lee. *Digital Systems Design Using Verilog.* Cengage Learning, 2015.

[43] L. W. Nagel. *SPICE2: A Computer Program to Simulate Semiconductor Circuits.* PhD thesis, EECS Department, University of California, Berkeley, 1975. URL: `http://www2.eecs.berkeley.edu/Pubs/TechRpts/1975/9602.html`.

[44] J.-J. L. L. . Wang and K.-T. Cheng. Critical path selection for delay fault testing based upon a statistical timing model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(11):1550–1565, Nov 2004. `doi:10.1109/TCAD.2004.835137`.

[45] A. Gattiker, S. Nassif, R. Dinakar, and C. Long. Timing yield estimation from static timing analysis. In *Proceedings of the IEEE 2001. 2nd International Symposium on Quality Electronic Design*, pages 437–442, March 2001. `doi:10.1109/ISQED.2001.915268`.

[46] S. Eggersglüß and R. Drechsler. *High Quality Test Pattern Generation and Boolean Satisfiability.* SpringerLink : Bücher. Springer New York, 2012.

[47] M. Bhushan, A. Gattiker, M. B. Ketchen, and K. K. Das. Ring oscillators for CMOS process tuning and variability control. *IEEE Transactions on Semiconductor Manufacturing*, 19(1):10–18, Feb 2006. `doi:10.1109/TSM.2005.863244`.

[48] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, and A. C. and. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. In *2002 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.02CH37315)*, volume 1, pages 422–478 vol.1, Feb 2002. `doi:10.1109/ISSCC.2002.993112`.

[49] J. W. Tschanz, S. Narendra, R. Nair, and V. De. Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performance microprocessors. *IEEE Journal of Solid-State Circuits*, 38(5):826–829, May 2003. `doi:10.1109/JSSC.2003.810053`.

[50] J. Tschanz, S. Narendra, R. Nair, and V. De. Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performance microprocessors. In *2002 Symposium on VLSI Circuits. Digest of Technical Papers (Cat. No.02CH37302)*, pages 310–311, June 2002. `doi:10.1109/VLSIC.2002.1015112`.

[51] S. Wang, J. Chen, and M. Tehranipoor. Representative critical reliability paths for low-cost and accurate on-chip aging evaluation. In *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 736–741, Nov 2012.

[52] F. Firouzi, F. Ye, K. Chakrabarty, and M. B. Tahoori. Representative critical-path selection for aging-induced delay monitoring. In *2013 IEEE International Test Conference (ITC)*, pages 1–10, Sep. 2013. doi:10.1109/TEST.2013.6651924.

[53] A. Vijayan, A. Koneru, S. Kiamehr, K. Chakrabarty, and M. B. Tahoori. Fine-Grained Aging-Induced Delay Prediction Based on the Monitoring of Run-Time Stress. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(5):1064–1075, May 2018. doi:10.1109/TCAD.2016.2620903.

[54] T. McConaghy, K. Breen, J. Dyck, and A. Gupta. *Variation-Aware Design of Custom Integrated Circuits A Hands-on Field Guide: A Hands-on Field Guide*. Springer, New York, 2013. URL: http://cds.cern.ch/record/1500187.

[55] J. K. Rangan, N. P. Aryan, J. Bargfrede, C. Funke, and H. Graeb. Timing Variability Analysis of Digital CMOS Circuits. In *Reliability by Design; 9. ITG/GMM/GI-Symposium*, pages 1–2, Sept 2017.

[56] J. Lee and S. Leyffer. *Mixed Integer Nonlinear Programming*. Springer Publishing Company, Incorporated, 2011.

[57] ILP package. https://cran.r-project.org/web/packages/lpSolve/index.html. Accessed: 2017-03-06.

[58] IBM ILOG CPLEX Optimizer. urlhttp://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, December 2010.

[59] R. Horst and P. Pardalos. *Handbook of global optimization*. Number v. 1 in Nonconvex optimization and its applications. Kluwer Academic Publishers, 1995.

[60] A. M. Tillmann. Equivalence of Linear Programming and Basis Pursuit. In *PAMM (Proceedings in Applied Mathematics and Mechanics*, volume 15, pages 735–738, 2015. DOI: 10.1002/PAMM.201510351.

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

[62] D.-Z. D. et.al. *Multilevel Optimization in VLSICAD*. Springer Science, 1st edition, 2003.

[63] L. Wang. Synthesis and Analysis of DDROs to Track Chip Timing. Master's thesis, Technical University of Munich, Munich, Germany, 2018.

[64] R. Gentleman and V. J. Carey. *Unsupervised Machine Learning*, pages 137–157. Springer New York, New York, NY, 2008. URL: `https://doi.org/10.1007/978-0-387-77240-0_10`, `doi:10.1007/978-0-387-77240-0_10`.

[65] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik. *Cluster: Cluster Analysis Basics and Extensions*, 2018. R package version 2.0.7-1 — For new features, see the 'Changelog' file (in the package source).

[66] B. Everitt and T. Hothorn. *An Introduction to Applied Multivariate Analysis with R*. Use R! Springer New York, 2011.

[67] B. Everitt. *Cluster Analysis*. Heinemann Educ. Books., London, 1974.

[68] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[69] D. Arthur and S. Vassilvitskii. K-means++: the advantages of careful seeding. In *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[70] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, New York, 1997.

[71] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):205–233, 2017. URL: `https://journal.r-project.org/archive/2017/RJ-2017-008/RJ-2017-008.pdf`.