An Efficient Timing and Clock Tree Aware Placement Flow with Multibit Flip-Flops for Power Reduction

by

Jasmine Kaur Gulati
MT14081

Under the Supervision of
Dr. Sumit Darak
Bhanu Prakash, ST Microelectronics

Submitted
in partial fulfillment of the requirements for the degree of
Master of Technology in Electronics & Communication
Engineering with specialization in VLSI & Embedded Systems

to

Indraprastha Institute of Information Technology Delhi
June, 2016

Abstract

Power consumption has become a bottleneck for modern system-on-chip (SoC) designs. With the advancement towards the deep sub-micron technology, the SoC design consists of components that prompt to a higher power density. In VLSI designs, the performance of an integrated circuit (IC) is governed by the frequency of the clock at which it operates, thus clocking is the major source of power dissipation in a design. Designing clock network is a critical task for high-performance circuits as it directly impacts clock skew, jitter, chip power and area of SoC under process variations.

Multi-bit flip-flops (MBFFs) have appeared as a low-power solution for the nanometer technology. The number of clock sinks reduces during clock tree synthesis (CTS) with the application of MBFFs. As a result, the clock network shows increment in core utilization, improvement in routing, reduction in power consumption and timing violations. The clock insertion delay (CID) is another key metric of clock network and decreasing CID results in shorter clock network, less impact on crosstalk, less impact of process variation, and reduction in hold penalties.

This work introduces a novel placement strategy in integration with the electronic design automation (EDA) tool for MBFF generation having the prerequisite knowledge of clock tree architecture. The strategy irrespective of traditional placement flow consists of MBFFs that are generated by replacing single-bit FFs iteratively during placement. FF merging and MBFF generation algorithm have been proposed. The approach is made timing-aware with useful skew optimization. Experiment results show improvement in chip power by 44%, core density by 11.3% and clock power by 10.4%.

In addition to the above, another algorithm for minimizing the CID of the design has been proposed. This algorithm splits up the clock tree sinks with maximum CID to a separate pool, after the deep analysis of the clock tree structure. It also takes into account the floorplan of the chip, placement pin and the macro placement changes on the sinks. The results show that the average CID reduces by 9.2%.

Certificate

This is to certify that the thesis titled "**An Efcient Timing and Clock Tree Aware Placement ow with Multibit Flip-Flop Generation for Power Reduction**" submitted by **Jasmine Kaur Gulati** to Indraprastha Institute of Information Technology, Delhi for the award of the *Master of Technology* in *Electronics and Communication & Engineering* is an original research work carried out by her under my guidance and supervision.

The results enclosed in the thesis have not been submitted in any other university or institute for the reward of any other degree.

Dr. Sumit Darak
Indraprastha Institute of Information Technology Delhi

Bhanu Prakash
Consumer Product Division, ST Microelectronics
Greater Noida

Acknowledgements

This study would not have been successful without the support and direction of number of people who have contributed and assisted in the completion of my thesis work in multiple ways.

I would like to express my sincere gratitude to Dr. Sumit Darak, Assitant Professor, IIIT Delhi, for his valuable guidance and encouragement throughout my thesis work.

Bhanu Prakash, CPD, ST Microelectronics, for his motivation, patience, and concern for my research work. I am also thankful to him for all the technical knowledge he has shared with me.

Last but not the least, I would like to thank my father, my mother and my brother for being patient and supportive throughout my life.

Contents

List of Figures

List of Tables

List of Abbreviations

ASIC Application-Specific Integrated Circuit
CID Clock Insertion Delay
CTS Clock Tree Synthesis
DRC Design Rule Constraints
ECO Engineering Change Order
EDA Electronic Design Automation
FF Flip-Flop
HFN High Fan-Out Net
IC Integrated Circuit
IP Intellectual Property
MBFF Multi-bit Flip-Flop
OCV On-Chip Variation
PLL Phase-Locked Loop
PnR Place and Route
PPA Power, Performance and Area
PPO Post-Placement Optimization
RTL Register-Transfer Level
SDC Synopsys Design Constraint
SI Signal Integrity
SoC System on Chip
VLSI Very Large Scale Integration
WL Wire-length

Chapter 1

Introduction

Design of a System-on-chip (SoC) is becoming complicated as more functionality pertaining to the user requirements are being integrated onto the chip [1]. The semiconductor industry is moving on the path of Moores law and hence the process node is shrinking with an increase in the complexity of the design. With the integration of more and more components on a SoC, power density of the chip also increases. The various blocks on the chip may have different frequency requirements and hence, the design cycle has evolved to cater the need of high-speed implementations [2]. The high-frequency design consumes huge amount of power, therefore, power reduction is essential for improvement in the battery life and overheating situations.

In a SoC, clock distribution network synchronizes the flow of data signals across data paths. Design of these networks is a challenging task due to technology scaling which in turn can affect the system-wide performance and reliability [3]. Deep sub-micron technology nodes exhibit wire delay variation, temperature inversion, crosstalk penalty on signal and clock paths. Also, the clock networks consume a major amount of power of the whole design. Hence, with limited power budgets, clock power consumption is a key problem area in the modern VLSI designs as it influences the correctness, area, speed, and reliability of the synthesized system.

In the SoC design flow [4] [5], the steps from the specification of the design to the final tape-out of SoC are shown in Fig. 1.1. The entire flow is divided into two phases: front-end design and the back-end (physical) design. For the front-end design, the architecture according to the design specifications is laid out and the register-transfer level (RTL) code is generated. After the RTL verification, logic synthesis is carried out to map the RTL to gate-level logic. The next and the important step is the static timing analysis (STA) to verify and optimize the timing constraints of the design. The entire process is iteratively performed from RTL optimization to STA until the violations are controlled. After the final verification, the final gate-level netlist is handed to the physical design stage. The physical design flow is explained in the following section.

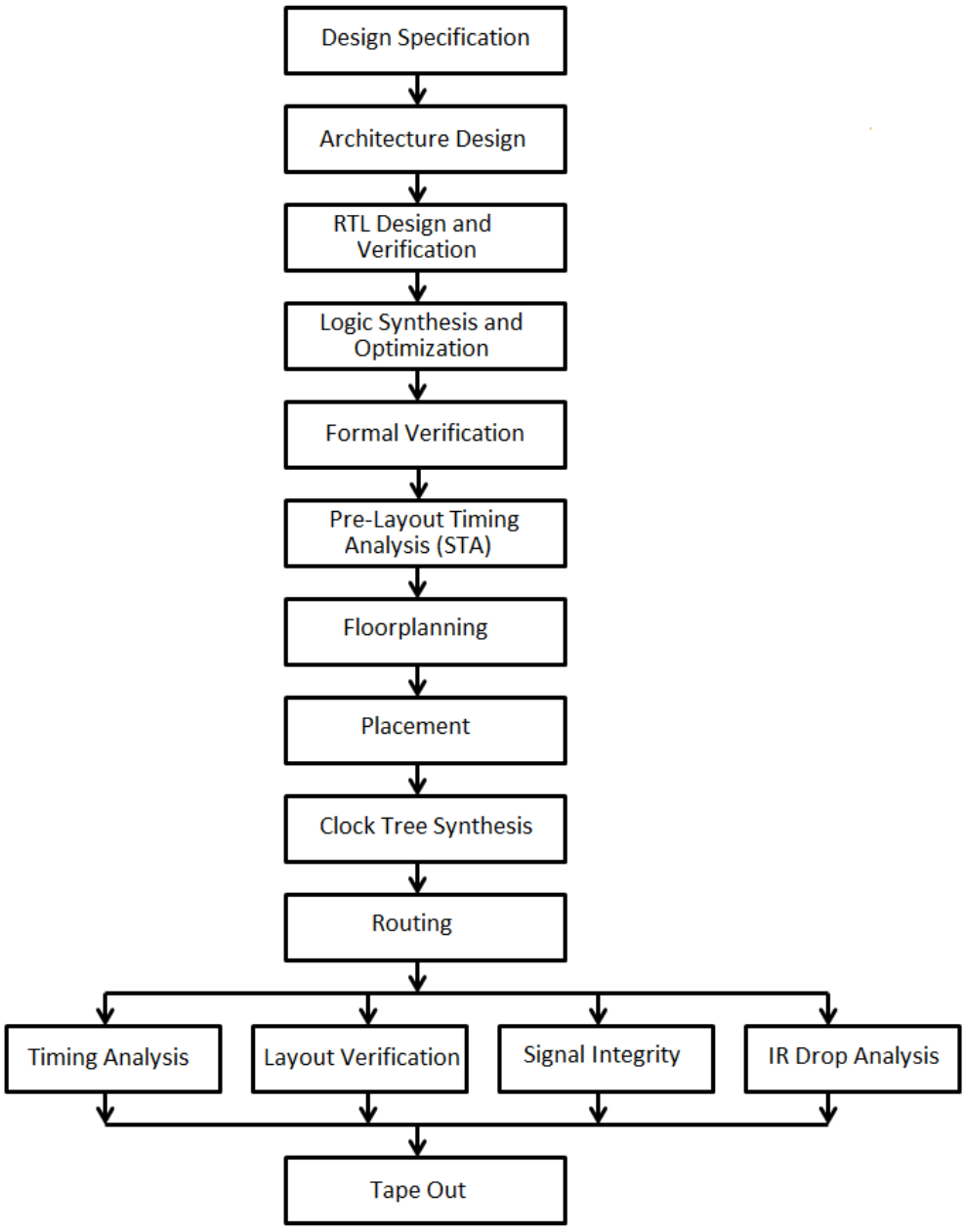


Figure 1.1: SoC Design flow

With the emergence of new technology, design methodologies to reduce the power consumption of the chip are required to achieve the power, performance and area (PPA) targets. The need of the hour is to find an approach which can attain the targets with efficient timing closure. The objective of this work is to address the PPA targets of

the real designs and provide a solution in integration with the current electronic design tools (EDA).

1.1 Overview of Physical Design Flow

Physical design is an integral part of design planning process [6]. The gate-level netlist, timing constraints, the technology file, the cell library file of the design are the inputs of this stage. It begins with floorplanning and continues till the final tape-out of the chip. The following sub-sections explain the steps involved in physical design flow in detail.

1.1.1 Floorplanning

Floorplanning is the first and the most challenging stage in the physical design process [7]. As a SoC consists of the various intellectual property (IP) blocks, memories, analog blocks and multiple power domains, thereby the positioning of blocks on the die affects the quality of the design. The process of floorplanning aims to provide a proper arrangement of blocks, pins and power grids for their efficient operation in parallel. The process includes the following integral steps:

- IO Pad Placement
  In this step, the signal pad, power supply pad, and analog pads positions are defined by a padring.

- Macro Placement
  The objective here is to arrange the hard macros to optimize the remaining area for placement of logic cells. Macro and routing blockages are designed in this step to prevent the congestion on the chip.

- Power Grid Placement
  To supply the power to all the blocks in the design, power grids are designed. The top two metal layers are reserved for the power grids to prevent the voltage drop [8]. If the power rails are routed in the lower metal layers, the usage of lower routing resources will increase and hence, the congestion.

As the floorplan significantly affects the performance of the design and the timing violations, the floorplan designers undergo various iterations to achieve the best quality floorplan.

1.1.2 Placement

Once the optimized floorplan is ready, the next step is the placement of standard logic cells to locations on the chip [9]. Placement tool determines the location of the components on the die corresponding to the timing constraints, interconnect and wire

lengths and power dissipation. Another objective of placement is to optimize the timing of the design by removing the violations due to timing. The placement algorithm used by the tool determines the interconnect length and hence, the routability of the design. The placement optimization is performed in four stages:

- Pre-placement optimization
  In this phase, before the placement begins, design netlist is optimized. The high fan-out nets (HFNs) such as reset, enable are synthesized during the logic synthesis in front-end flow without the placement information. Therefore, during placement HFNs are collapsed for re-synthesis. Since the clock latency is not known, clocks are checked to be ideal before the synthesis of HFNs, otherwise, the HFN synthesis will be done on clock nets.

- In-placement optimization
  Further optimization of the design involves re-sizing and change in position of cells. Timing optimization based on setup and hold requirements is performed. Congestion analysis and optimization are a part of this phase of placement.

- Post Placement Optimization (PPO) before CTS
  Before the clocks are in the propagated mode after the CTS, this phase analyses and fixes the timing violations.

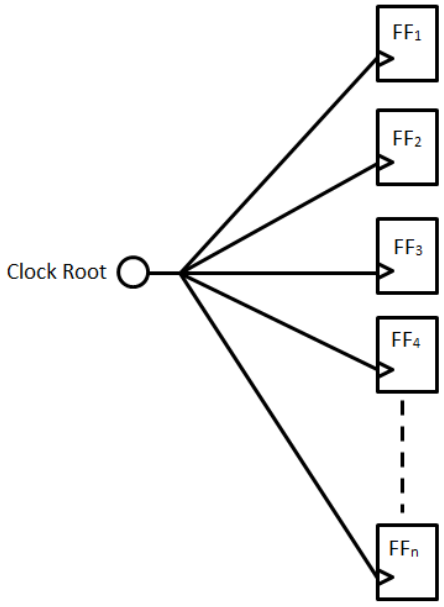
- Post Placement Optimization (PPO) after CTS
  After the CTS, clocks are in the propagated mode and hence to preserve the skew in the design, timing check is performed again [10].

1.1.3 Clock Tree Synthesis

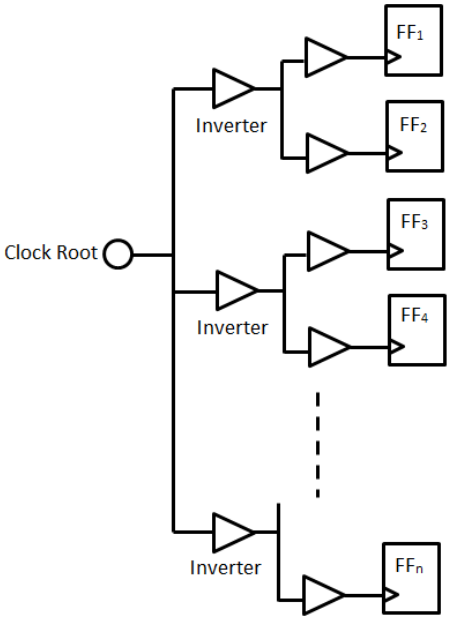
The standard cells and the macros are placed at a fixed optimized location, but the clock is still in the ideal mode. Since the data transfer between the various functional elements on the chip is carried out by the clock signals, the clock enters into the propagated mode. For a design to attain setup and hold requirements, the clock input of each sequential element must be in synchronization. A single clock net cannot drive the huge number of sequential elements present in the design. The clock distribution network distributes the clock signals from common point to all the clock pins of the elements. CTS is the process of balancing clock skew [11] and minimizing the uncertainty of arrival time. In Fig. 1.2a, a single clock source drives n number of FFs and the clock network is not balanced. In Fig. 1.2b, CTS balances the clock network and buffers are added to clock nets.

1.1.4 Routing

After CTS, the next stage is to determine the precise paths for interconnections. The tool has the netlist about the logical connectivity in the design. The aim of the routing process is to ensure that no design rule violations exist in completing the routes as well



(a) Clock net before Balancing



(b) Clock net after CTS

Figure 1.2: Balancing during CTS

as the timing constraints and clock skew are met [5]. Routing is carried out in two phases:

- Global Routing
  The design is partitioned in various routing regions and the route is analyzed by the number of available tracks in each region. The estimation of congestion corresponding to the available routing tracks is done. The congestion map is analyzed before detailed routing to evaluate routing issues.

- Detailed Routing
  In this phase, the tool tries to fix all design rule violations after the global route lays the actual interconnecting wires. The entire design is traversed by the tool until entire routing is pass with no major violations.

1.2 Challenges of Clock Network in Physical Design Flow

The main task for the SoC designers is to ensure that the clock reaches with no skew and almost no jitter in the entire chip. Multitude of clocks present in the modern designs has shifted the paradigm to multiple clock domains and thus, attainment of clock balancing and zero skew are difficult. Physical design flow starting from floorplanning make sure that at every stage optimization is performed. The design of the chip should not have severe setup and hold violations. There are factors which act as key issues related to CTS during physical design flow. Following subsections address the dominant hurdles in the design flow.

1.2.1 Process Variations

The variations due to the shrinking of devices are increasing with the newer technology nodes [12, 13]. A single chip consists of million of transistors packed on it, the variations arise during the manufacturing of these transistors. Wafer-to-wafer variations include variations due to dopants, oxide-growth rates, and the variation in the gate oxide stress levels. Due to scaling, the number of dopant atoms has decreased and concentration of the dopants has become a pronounced factor. Width and length of the transistor are the key factors of a device. Variability due to width and length affect the switching speed and the amount of leakage current in a device. The direct impact of process variations is on the yield and the performance of the design. The variations in the standard cell result in the mismatch across the various clock trees in the clock network.

1.2.2 Power Supply Variations

Supply voltage scaling causes variations in the switching activity across the die. The uneven power dissipation across the die is the result of fluctuations in the demand of current over a short interval of time. Eq. 1.1 gives the self-induced electromotive force,

the amplitude of the voltage drop due to sudden current variations.

$$V = L \times \frac{dI}{dt} \tag{1.1}$$

where I is the current and L is the self-inductance through the supply line.

Ripples or noise voltage is induced in the supply lines due to the presence of parasitic inductance. The current flows in the chip via interconnect which has a finite resistance. The variation in the resistance leads to IR drop. Referring to Fig. 1.3b, the propagation delay improves at high voltage and the cell becomes fast.

1.2.3 Temperature Variations

Temperature varies continuously on the chip while it is operating due to the power dissipation on the chip. Eq. 1.2 gives the dynamic power dissipation, where α is the activity factor, C is the load capacitance and V is the supply voltage. With the increase in temperature the drain current decreases. Both device and interconnect depend on the temperature and hence, are affected by the variations in the temperature. As shown in Fig. 1.3c, the propagation delay of the cell increases with the increase in temperature as mobility of electrons decreases. The threshold voltage of a transistor also depends on temperature. With the increase in temperature, the threshold voltage increases.

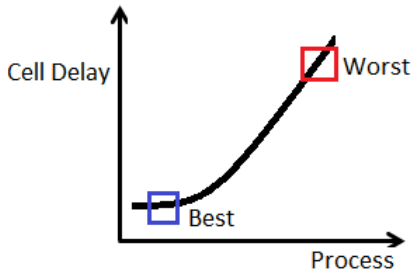
$$P = \alpha \times C \times V \times (f)^2 \tag{1.2}$$

1.2.4 Signal Integrity Issue

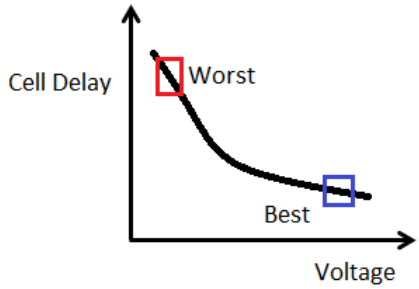
Signal integrity (SI) issues include crosstalk, IR drop, and electromigration. Referring to fig 1.4, relative switching of wires on account of the capacitive coupling results in crosstalk noise. Depending on the amplitude of the crosstalk noise, a delay uncertainty is superimposed on the victim net. With the increase in the clock frequency rates, the capacitive coupling dominates and results in significant delay in data paths.

IR drop in the wires is caused by the current from the supply voltage and the finite resistance of the wire. In case the wire resistance is very high or the current through the transistor is higher than estimated, there is an unwanted voltage drop. This unpredicted drop causes timing degradation in the signal and clock nets. It produces unwanted clock skew in the design and hampers the signal integrity of the design.

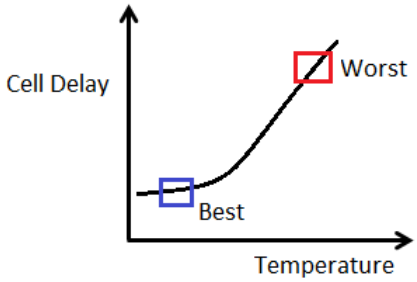
Electromigration depends on the current density and for high current density wires, it causes wear out of metal interconnects [14]. The metal ions migrate with the "electron wind" induced by the high current density.



(a) Cell Delay vs Process



(b) Cell Delay vs Voltage



(c) Cell Delay vs Voltage

Figure 1.3: Cell Delay with respect to Process-Voltage-Temperature Variations

In the SoC design flow Fig. 1.1, signal integrity analysis is performed after routing during the post-layout. But it can lead to costly iterations and many designs fail to close due to SI effects.

1.2.5 Timing Violations

A design consists of millions of gate and multiple clocks. There is multitude of timing paths for analysis and without the proper definition of clocks, the complete timing path becomes invisible to the timing analysis tool. Timing violations due to setup and hold, clock skew and due to the signal integrity issue cause hindrance towards the timing closure for a design.

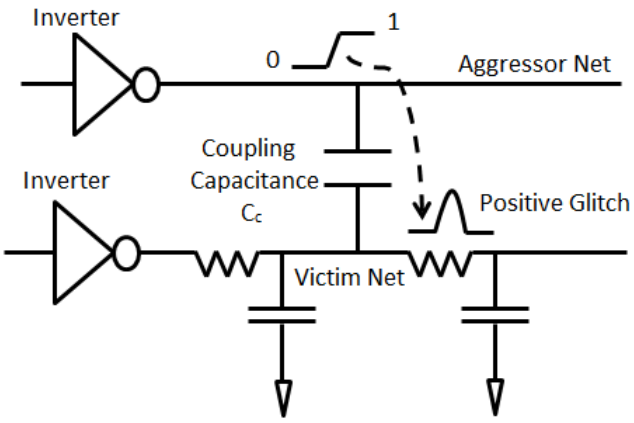


Figure 1.4: Crosstalk Noise due to Capacitive Coupling

1.2.6 Design Complexity

Modern designs have millions of cells and the single chip is broken down into a hierarchy of modules. The timing budgets are created for the whole design, which permits the engineers to use hierarchical design methodology and work on their modules for the timing closure. To ease the CTS for designs with multi-million flip-flops (FFs), multi-stage clock gating structures are incorporated. Complex clock structures due to the multitude of clocks present in the design impose a challenge for setting up the optimistic environment for the designers. Multi-mode and multi-corner scenarios increase with the technology proceeding towards few nanometers. Hence, the timing closure, as well as the clock network, have become a challenge to the SoC designers.

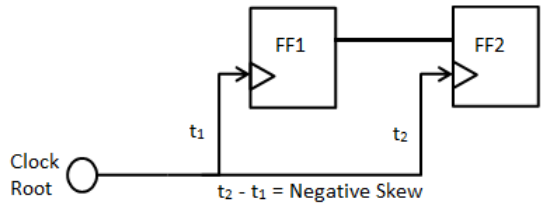
1.3 Conceptual Overview

It is imperative to understand few concepts for the problem statement of this thesis work. In this section, to develop the background for the proposed work useful skew methodology, multi-stage clock gating, clock insertion delay (CID) and multi-bit flip-flop (MBFF) are explained .

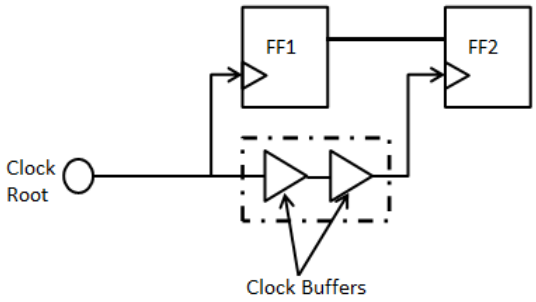
1.3.1 Useful Skew Optimization

In order to achieve target skew, useful skew optimization [15, 16] is incorporated. The violating path can be fixed not only with the data path changes but also with the clock path changes as shown in Fig. 1.5. In Fig. 1.5a, violating path exists between launch FF and the capture FF due to negative skew. Clock buffers are added to the capture clock path as show in Fig. 1.5b to balance the skew. The clock latencies of FFs are skewed intentionally to achieve the target skew. Further, in this manner the benefits are increased clock frequency and the timing margins of the design. Hence,

the skew transforms into a manageable resource and is useful to improvise the timing performance of the design.



(a) Negative Slack



(b) Useful Skew to achieve Target timing

Figure 1.5: Useful Skew Optimization

1.3.2 Multi-Stage Clock Gating

Multi-level Clock Gating [17], is implemented for dynamic power reduction of the inactive blocks or the sub-blocks. In this method of clock gating, the clock of first-level clock-gating is gated by the next-level clock gating as shown in Fig. 1.6. Similarly, multiple levels of clock gating can be employed in a design. In this manner, the power can be switched off for any module or modules, to save power. The modern VLSI designs consist of multiple levels of clock gating.

1.3.3 Clock Insertion Delay

The time taken by the clock signal to reach the clock tree sink from the clock source such as the phase-locked loop (PLL) is termed as the CID. It is classified into two categories:

- Clock Source Latency
  It is defined as the time interval the clock signal takes to reach from the clock generator source to clock root pins of the design.

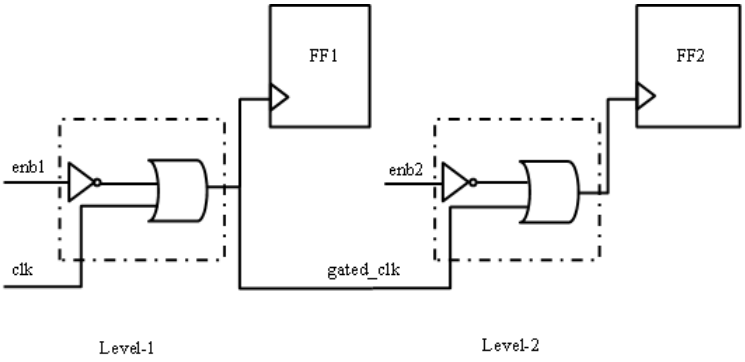


Figure 1.6: 2-Level Clock Gating

- Clock Network Latency
  The delay of the clock signal from the clock root pins of the design to clock tree sink pins of FFs in the design.

Fig. 4.1 shows the two components of CID in an example design. CID is the sum of clock source and clock network latency.

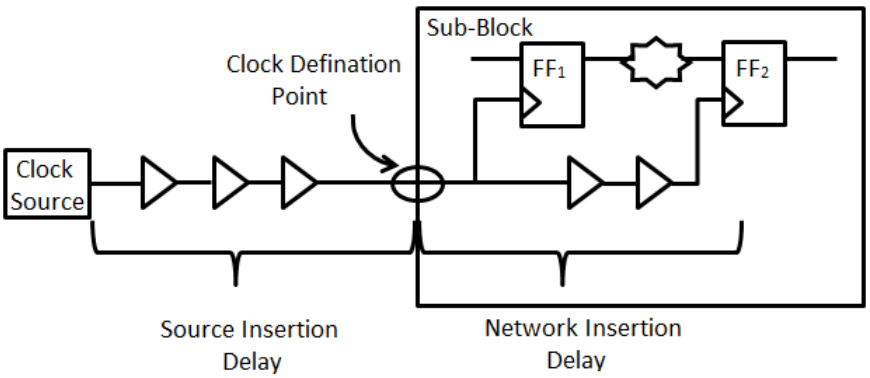


Figure 1.7: Components of CID

1.3.4 Multi-bit Flip-Flop

FFs are prominently used in the synchronous designs and for the storage of n-bit data, n independent single-bit FFs are incorporated. Referring to Fig. 1.8, each single-bit FF consists of two inverters to generate the clock signals which are opposite in phase, one master and a slave latch. With the advancement in technology nodes, multiple FFs can be driven by a minimum-sized clock driver. As a consequence, several redundant inverters are eliminated and power dissipation is reduced. Therefore, MBFFs have evolved as a solution to save power and area of a chip [18, 19].

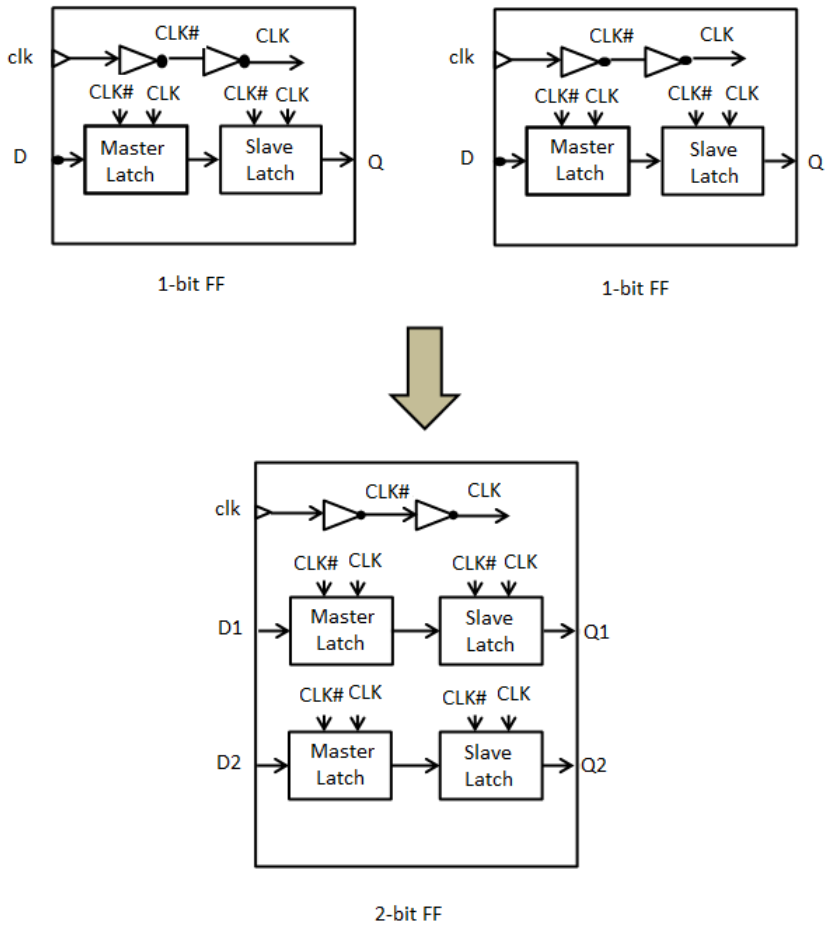


Figure 1.8: Merging two 1-bit FFs to one 2-bit FF

Application of MBFFs results in the following benefits:

- Improvement in core utilization
  Due to fewer sequential cells in the design and the reduction of clock buffers and clock nets, core utilization improves.

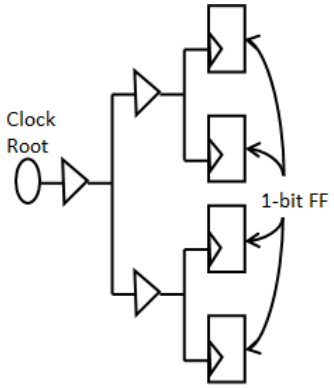
- Power Reduction
  Due to fewer clock tree sinks, the power of the clock network reduces.

- Easy Timing Closure
  As the number of FF reduces and the MBFFs share the common clock signal, the depth of clock tree reduces and hence, the timing violation due to skew an insertion delay also decrease.

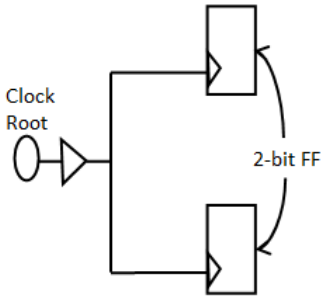
- Reduction in FF area
  Due to the shared clock drivers and the reduction in the hold buffers due to

internal scan chains, total FF area reduces.

Fig. 1.9a shows an example of a simple clock tree network with 1-bit FFs and Fig 1.9b with 2-bit FFs, after replacement the clock tree has fewer sinks.



(a) Clock tree with 1-bit FFs



(b) Same Clock tree with 2-bit MBFFs

Figure 1.9: Clock tree structure after MBFF generation

1.4 Motivation

For the nanometer technology nodes, the yield of the high-performance designs is of prime importance to achieve the major objectives of timing, SI, power, and area. With the growing complexity of the design, the designers look for alternate techniques to enhance the performance of the design with no extra burden for the designer. To reduce the design effort and to cope up with the shorter product cycles, ASIC design incorporates reuse of IP components. This integration of the various components increases the product complexity and hence, affects the PPA targets. In order to stabilize the cost function of a chip which is dependent on the PPA, the focus has shifted for improvement in current design methodologies and the SoC design flow. The physical

design flow used in the industry incorporates single-bit FFs using EDA tools. In order to reduce the power consumption and improve the timing closure of the design, there is a need for an efficient methodology during physical design flow. The power saving methods and components can be incorporated in the modern VLSI designs. Since, the clock is considered as the major power-dissipating component of the chip, thus CTS becomes the target of optimization for the semiconductor industry.

Chapter 2

State of the Art

This chapter highlights the work related to this study, in use by the industry and the academia, followed by explaining the details of the issues existing in the current methodology.

2.1 Related Work

A robust tool such as Cadence Innovus [20] is capable of timing driven placement and routing. The latest version incorporates clock concurrent optimization for timing optimization. It merges the clock tree synthesis with the physical optimization and simultaneously achieves timing convergence. Several design methodologies such as clock gating [21], [22], power gating [23], buffer sizing [24], and MBFFs [18, 25, 26] have been introduced to minimize the power consumption of the design due to clocking. MBFFs proved as the promising solution for low-power designs. MBFFs have been introduced at different stages of the design flow i.e. during RTL Mapping, during physical synthesis [27] and post-placement [28].

However, it is complicated to introduce MBFFs without floorplan and placement information as it can degrade the timing and congestion budgets. The practical industrial flow infers to MBFFs at the synthesis stage with the floorplan information added to it. This methodology benefits the designer as it reduces the chances of congestion and timing violations in the backend ow. For more accurate timing analysis, recent work [29, 30] applies MBFFs during the backend flow after the placement stage. However, due to the fixed combination logic cells in the design after their placement, the number of mergeable FFs is less. Also, as the size of MBFF is more, it may cause difficulty in placement legalization and hence congestion in the design.

2.2 Current Methodology

In the recent work [31, 32], placement optimization with clock tree aware MBFF generation has been introduced to minimize power consumption and latency of clock network.

The merging of 1-bit FFs is performed according to the clock tree topology. Through the placement iterations, the FFs are progressively merged by referring to the original clock tree topology. This approach offers power reduction without degrading the performance of the design but it has limitations to address in the real designs for the merging of the FFs. This include:

- A design can have millions of instances due to its complex functionality. If the merging of the FFs is improper, then the complete design may suffer from congestion and timing violations. As the number of FFs increases dramatically, the need for full scenario exploration coupled with fast turn-around time is a must.
- As the designs today are vast and complex, the single chip is broken down into a hierarchy of modules. The timing budgets are created for the whole design, which permits the engineers to use hierarchical design methodology and work on their modules for the timing closure. The merging algorithm has to follow the hierarchical approach to target the FFs at the same hierarchy level.
- With gated clocks widely used in the design for dynamic power reduction, the grouping of FFs should combine similar toggling of the clock pulse. If the merging of a gated and non-gated driven FF takes place due to any reason, then the logic will get affected.
- Complex clock structures due to the multitude of clocks present in the design impose a challenge for setting up the optimistic environment for grouping the FFs. This limits the number of FFs for merging.
- Further, SoC designers use benchmark CAD tools for the physical design flow which target for the minimization of cost function PPA (power, performance, and area). The integration of the algorithm with these benchmark tools will result in huge benefits, especially when the technology is advancing to the nanometer.

To achieve proper and reliable flop merging, it is important to consider all the above-mentioned points and modify the merging algorithms for the real designs. In this work, MBFF generation algorithm is formulated to tackle the above-mentioned challenges of real designs.

Chapter 3

Proposed Timing and Clock Tree Aware Placement Flow with Multibit Flip-Flop Generation

This chapter introduces a novel placement methodology with clock-tree aware MBFF generation for power optimization and the corresponding proposed algorithm for FF merging. The methodology offers an additional feature to the EDA tools to integrate MBFFs during placement iterations with the knowledge of the clock-tree structure of the design. Timing is considered as a strict constraint, therefore at each stage timing is checked and hence the methodology is also timing-aware. Merging of the FFs results in the reduction in power consumption of the design and improved routability of the design. The proposed methodology consists of four major steps: 1) timing-driven global placement; 2) fast CTS; 3) MBFF ECO generation and 4) timing-driven incremental placement.

3.1 Methodology Overview

Referring to Fig. 3.1, the first step of the methodology is timing-driven global placement. It generates the locations of the cells based on the minimum placement density and minimum interconnecting wire-length (WL) criteria. Further, CTS is performed with the initial cell locations and clock tree is generated. In order to minimize the run-time, it is performed in the fast mode without the trial route. The merging of FFs is done after this step when MBFF ECO is generated based on the merging conditions. FFs are progressively merged and are replaced with MBFFs; they are placed during timing-driven incremental placement. The process of merging is iterative and repetitive so as to avoid congestion and timing violations. Until the largest bit number of MBFF is achieved and no more MBFF can be applied, merging is repeated. Timing violations are checked after the MBFF are generated, if the timing is worse than the initial global

placement timing then the MBFF corresponding to the timing violations are iteratively disintegrated into smaller bit number FFs. At the final stage, legalized placement is obtained with the MBFFs in the design. It is to be noted that, Cadence Innovus does not automatically merge the FFs. Hence, the FF merging algorithm has been proposed to find the best candidates of FFs to be merged and replaced with MBFF. Hence, the MBFF ECO generation is integrated with the FF merging algorithm. Also, various reports are generated at every step for the understanding of the user.

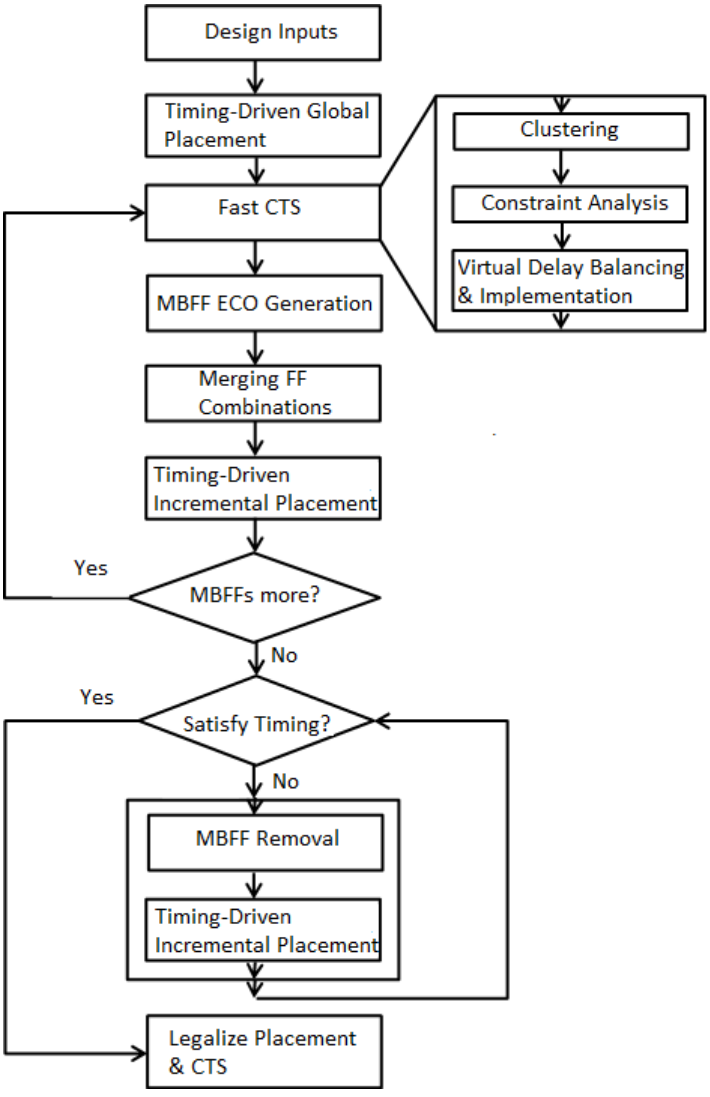


Figure 3.1: Proposed Clock Tree Aware Placement Flow with MBFFs

3.2 Steps of Methodology

In this section, the steps involved in the placement flow are explained. The placement and CTS optimizations are done by EDA tool. However, the mode of operation of the integrated steps such as placement and CTS is varied according to the proposed work. The following sections elaborate the steps involved in the proposed placement flow.

3.2.1 Timing-driven global placement

In the first stage, timing-aware global placement is performed with the design netlist, MBFF library, floorplan and timing constraints as inputs as shown in Fig. 3.2. Placement is done in the timing-driven mode so that the timing violations can be tested at an early stage. Tool efficiently fulfills the objective of minimized WL and placement density during the placement of logic cells. The placement is done with the EDA tool to find the best locations of the cell and to consider the two objectives of general global placement for best trade-off.

$$minW(x, y) \tag{3.1}$$

$$minD(x, y) \tag{3.2}$$

where W(x,y) is the WL function of all the signal nets and D(x,y) is the placement density function for each bin in the core.

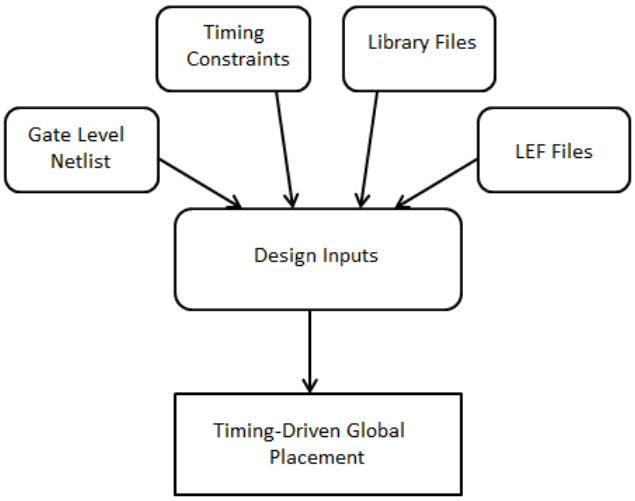


Figure 3.2: Inputs to Timing-driven global placement stage

These are the two basic objectives, but the EDA tools use complex algorithms to minimize the total collaborative function of these two equations, Eq. 3.1 and Eq. 3.2 so as to obtain the optimum placement with millions of cells in the real design.

3.2.2 Fast CTS

After the cells are placed and the timing is satisfied, CTS is performed. In order to ensure that it does not incur huge run-times, it is done without trial routing. CTS involve the following three steps:

1. Clustering
   In this stage, the logic elements operated in the same clock domain or with similar input timing registers are grouped into clusters. In the cluster based clock tree approach, the clusters are optimized individually to meet the DRC constraints such as maximum capacitance, maximum fan-out and maximum transition are checked. Fig. 3.3 depicts an example of cluster formation.

(a) FFs Before Clustering **(b)** Formation of clusters after Clustering

Figure 3.3: Clustering during CTS

   The major advantage of clustering is that the insertion delay and skew can be planned by designers to achieve the minimum timing requirements. The register clustering algorithms combine the logic elements after the placement is done and continuously shrink it with respect to the target skew and the target insertion delay. As the chip consists of both clock and the signal nets, there could be a negative impact of clustering on the signal nets due to the movement of the registers. Moreover, it may lead to congestion or routing failure in the design.

   Henceforth, due to the process variations and the requirement of timing closure, new design methodologies are being incorporated by the EDA companies. The algorithm for clustering is a customized cluster-based CTS for Cadence Innovus and hence, it utilizes the best solution to improve the performance of the design and meet the timing margins.

2. Constraint Analysis
   After the clusters are formed, the timing constraints included in the SDC are

identified and their inter-relation is modeled. The skew and insertion delay targets are mapped with an additional amount of delay to satisfy these constraints. In this step, the amount of delay to be added for balancing the clock tree nodes is calculated under all possible scenarios.

3. Virtual Delay Balancing & Implementation
   Virtual delay balancing is the methodology by which the additional delay is annotated on the clock nodes to achieve the solution found by constraints analysis. The clock enters the propagated mode and thus virtual delay is the extra delay to match the ideal mode timing and the timing due to its propagation. The virtual delay is implemented through real physical cells loaded in the input library files.

   Further, the refinement is done to achieve the best possible solution through the real physical cells. Thus, this is the timing optimization step where the objective is the target propagated clock timing. It maps the gap be the ideal and propagated timing of clocks. The complete clock architecture is ready at this stage and is passed as an input to the next stage to find the merge-able flops in the clock tree.

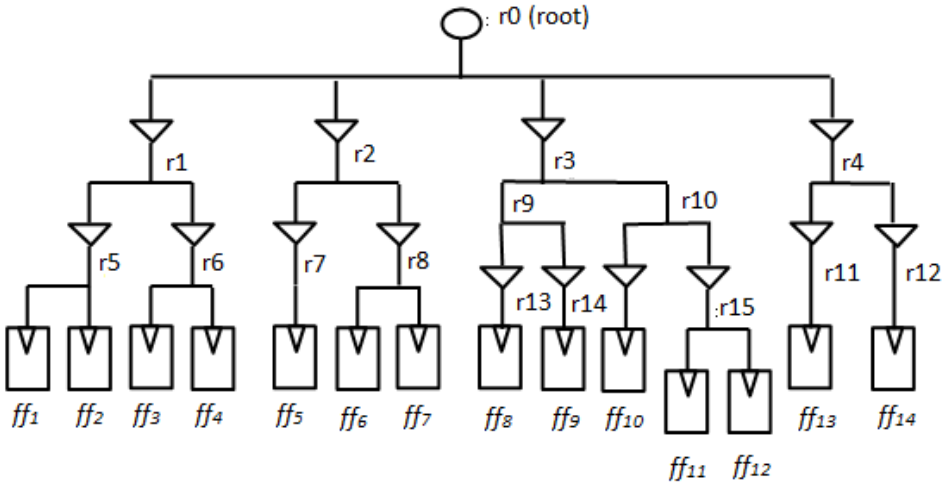
Fig. 3.4, shows an example of the clock tree built after CTS. In Fig. 3.4a, balanced clock routing tree is shown and in Fig. 3.4b, corresponding levels in the clock tree are shown. The clock tree level helps in determining the insertion delay for the particular FF present in the level. The next step works on the synthesized clock tree generated during this step to find the set of merge-able FFs.

3.2.3 MBFF ECO generation

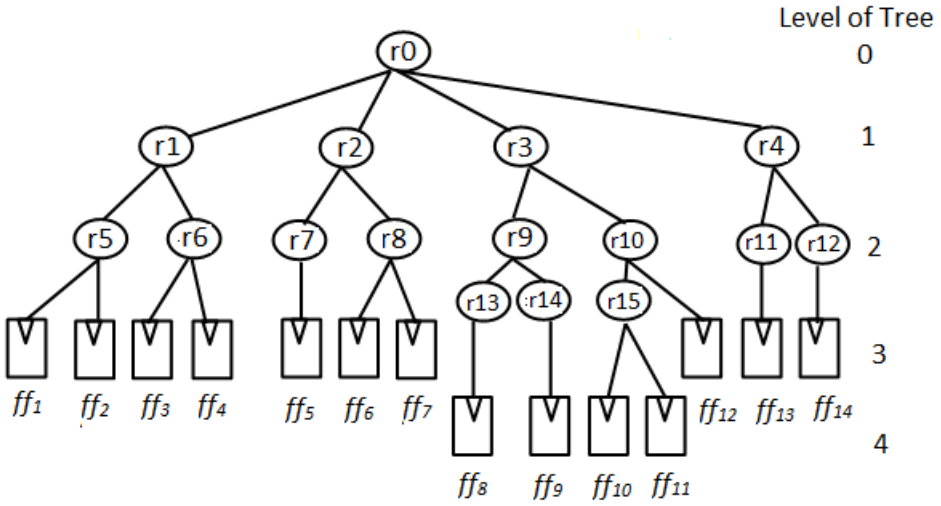
Before the MBFF generation, a look-up table is generated for FF merging with the consideration of timing, insertion delay, signal net and clock net Wl. Thus, to merge the FFs of a particular clock tree, merge-able conditions are defined. These conditions cater the design complexity for FF merging. Each clock tree is then checked for its capacity to merge the FFs into MBFF. The merge-able conditions between two flops *f(i)* and *f(j)*, are derived to ensure the correct and efficient replacement of FFs to MBFF. These are defined as:

- The merging of FFs far from each other affects the placement quality and hence, it is required to merge FFs placed near to each other. The distance, in terms of coordinates, between *fi* and *fj* should be less than a maximum set value, *dist(max)*. Based on Eq. 3.3, the violations due to timing and routing are constrained as the resultant MBFF will not be very far away from its parent single-bit FFs.

$$dist(fi, fj) \leq dist(max) \tag{3.3}$$



(a) Clock tree structure



(b) Corresponding tree with levels

Figure 3.4: Example clock tree structure

- The two FFs are merge-able if the difference between their respective levels, *level(fi,fj)*, from the root is equal or less than a constant, k. In order to preserve the latency of the clock-tree, FFs with close tree levels are considered to be merge-able. The value of k cannot be too small or too large as the number of FFs reduces after the MBFF generation. Also, the clock tree levels and the clock net WL decreases dramatically.

  Henceforth, the value of the constant k is set to be 1, to ensure maximum merging and FF power optimization.

$$|level(fi) - level(fj)| \leq k \tag{3.4}$$

- As the single chip is broken down into a hierarchy of modules. The timing budgets are created for the whole design, which permits the engineers to use hierarchical design methodology and work on their modules for the timing closure. The merging algorithm has to follow the hierarchical approach to target the FFs at the same hierarchy level. The candidates FFs, *fi* and *fj*, should be in the same hierarchy level, *Hi* and have same pin configuration. As the clock architecture of modern SoC designs is hierarchical, this checkpoint is compulsory to ensure the correct merging of FFs.

- With gated clocks widely used in the design for dynamic power reduction, the grouping of FFs should combine similar toggling of the clock pulse. If the merging of a gated and non-gated driven FF takes place due to any reason, then the logic will get affected. Either both or none of the candidates FFs should be gated in the same clock-path. This condition is of utmost importance because it is difficult to infer during placement if the resultant MBFF should be gated or not. This decision is critical since it tends to affect the logical functionality of the module.

The algorithm 1 depicts the procedure for FF merging. At first, a particular clock tree is selected and all its sinks are included in the database. On these sinks, the conditions for merging are applied. To save the memory and to prevent extra run-time, distance is given the first priority. Inside the circular area, of radius, *dist(max)*, around *fi*, all the clock tree sinks are taken for further analysis of merge-able conditions and are added to a list, *mergeflops*. Rest all clock tree sinks are eliminated for merging with *fi*. With the elimination, the analysis of the other merge-able conditions is broken down into a smaller number of sinks. The other conditions are applied on the *mergeflops*. If more than one FF satisfies all the merge-able conditions, the one with the minimum distance from *fi* is the best candidate for merging.

Similarly, the method is repeated for all the sinks of the clock tree. The same algorithm is replicated on other clock trees as well. After performing the merging algorithm, the optimum solution for all FFs is obtained in the look-up table. On the basis of the results, the merging of FFs to MBFFs is performed and the netlist with MBFFs is generated.

3.2.4 Merging FF Combinations

Once the look-up table is generated, the weight of each FF with its candidates is calculated and it is denoted by w_{fij}. In the FF look-up table, the weight function from

Algorithm 1 FF Merging

```
clock_trees ← get_clock_trees()
for tree in clock_trees do
    clock_tree_sinks ← collection of all sinks of clock tree
    for sink in clock_tree_sinks do
        calculate position of sink or fi, x and y co-ordinates
        sink_pos ← calculate_position()
        sink_x_pos ← extract x coordinate from sink_pos
        sink_y_pos ← extract y coordinate from sink_pos
        calculate level of sink in clock tree
        level(sink) ← level_clock_tree()
        create a circle of radii, dist (max) around sink with
        center as (sink_x_pos, sink_y_pos)
        candidate_merge_list ← collection of all sinks in that region
        Apply conditions one by one on list of candidates
        for candidate in candidate_merge_list do
            level(candidate) ← level_clock_tree()
            if (|level(fi) − level(fj)| ≤ k) && hierarchy && clock gating then
                add the candidate to the look-up table
                mergeflops ← candidate
            end if
        end for
        calculate weight of all candidates in look-up table
        for mergeableflops in mergeflops do
            weight_sink ← calculate_weight(mergeableflops)
            weight is proportional to distance of the candidate from fi
        end for
        the less weight one is put in the combination table with fi
        best_to_merge ← min(weight_sink)
        fi can merge with best_to_merge
    end for
end for
```

Eq. 3.5 finds the best match such that the resultant signal net WL, clock latency, and the power consumption is minimum during MBFF generation.

$$w_{fij} \propto dist(fi, fj) \tag{3.5}$$

Fig. 3.5 gives an example to illustrate the formation of MBFF based on the FF merging algorithm. Fig. 3.5a, corresponds to the initial clock routing tree after initial timing-driven global placement. After performing the merging algorithm, the resultant 1-bit FF pairs are merged and are replaced by 2-bit FFs. Fig. 3.5b shows newly generated updated clock routing tree with 2-bit FFs. The 1-bit FFs are not merged to largest bit number FF available in the library but progressively to elude sudden change in placement profile.

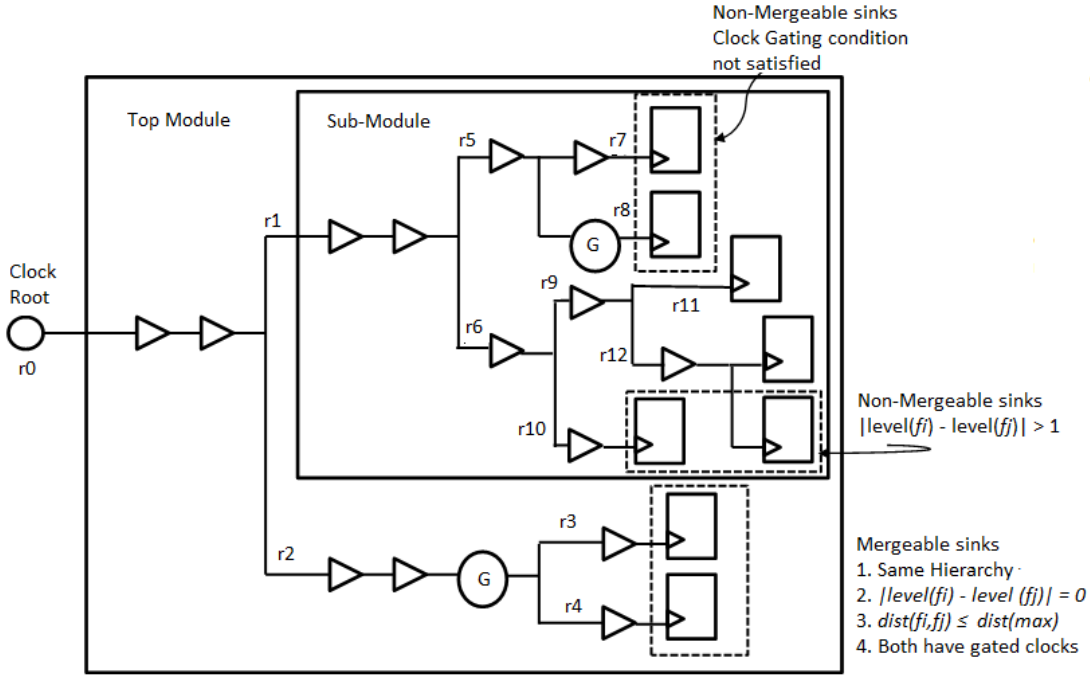
3.2.5 Timing-driven Incremental placement

After the netlist consisting of MBFFs is generated, incremental placement is done to find the legal location of the MBFF. As the size of MBFF is larger than the single bit FF, congestion and timing constraints can be violated hence timing-driven incremental placement performed. To ensure that the placement is legalizable and the timing constraints are met, timing-driven incremental placement refines the locations of combinational cells, sequential logic cells, and the newly generated MBFFs.

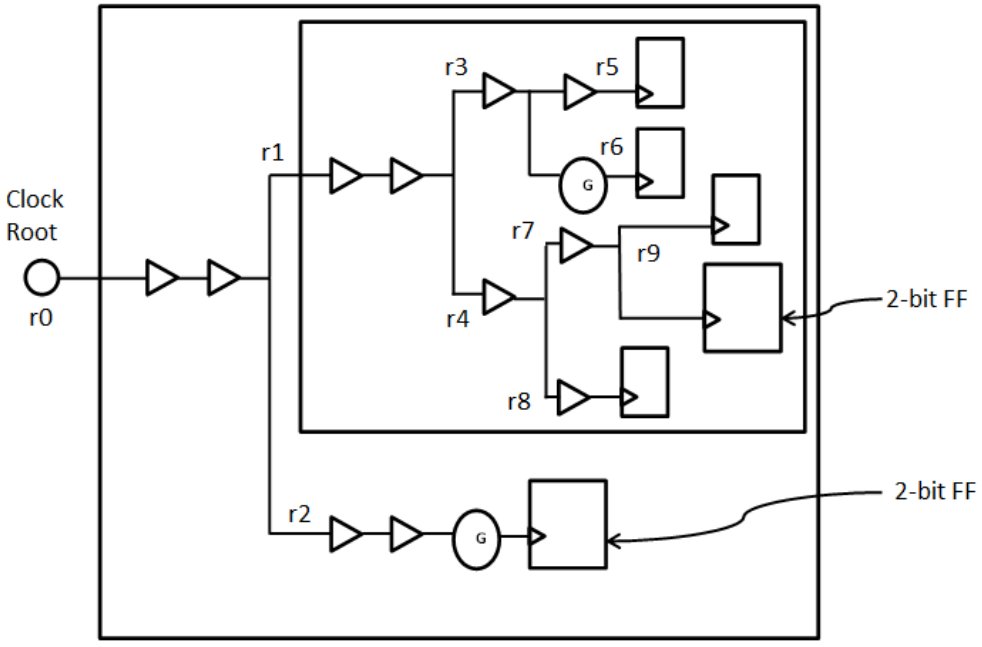
It is difficult to locate a legal location for a MBFF, and could be hazardous for the placement density constraints. Hence, to ensure a legalizable placement, Innovus implementation system incorporates new GigaPlace which accounts for slack and skew optimization.

In case after the MBFF generation, the timing results obtained from the incremental placement are worse than those in the initial placement stage, the MBFFs are iteratively disassembled to single-bit FFs. Once the timing slacks are improves, legalized placement with MBFFs is obtained after placement legalization. Further, CTS is performed with MBFFs and the clock routing tree is obtained. Later, the entire physical design with MBFFs is accomplished upto route stage.

In this section, an overview of MBFF generation based on the clock tree during the placement iterations has been described. The intent of using MBFF instead of single-bit FF is to reduce the overall power and the net WL of the design. As explained in the previous sections, the candidates for the merging are picked from all the scenarios and then filtering based on the weight function as given in Eq. 3.5 takes place with the iterative placement. The iterations in placement continue until no more MBFFs can be generated in the clock routing tree. Clock tree is built after every MBFF formation as it lowers the risk in skew due to the movement of FFs during iterative placement.



(a) Mergeable Conditions applied to clock tree with single-bit FFs



(b) Final clock tree with newly generated MBFFs

Figure 3.5: FF merging and MBFF generation

Chapter 4

Proposed Clock Insertion Delay Reduction Algorithm

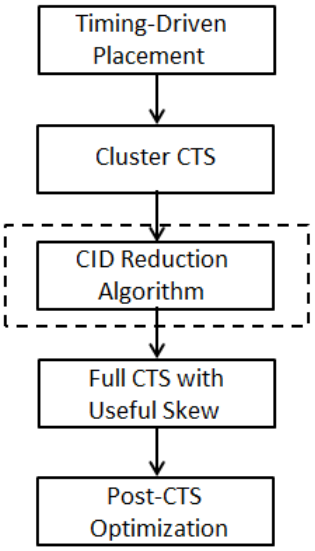
To overcome the wire delay variations, crosstalk penalty on signal and clock nets, extra pessimism to model the on-chip variations, a specific method to mitigate such challenges has been identified. CID reduction is pointed out as a solution for all the above-listed challenges. The current flow used in the industry does not target the violating insertion delay clock tree sinks. However, performing the check on the violating clock nets and the clock tree sinks can significantly improve the performance in terms of insertion delay of the clock tree structure.

This chapter introduces a novel way to reduce CID of all or selected clocks using intuitive and step by step approach. The methodology works on extracting the dominant clock for a design and the violating clock tree sinks at the early CTS stage. It also explains the significance of the proposed algorithm.

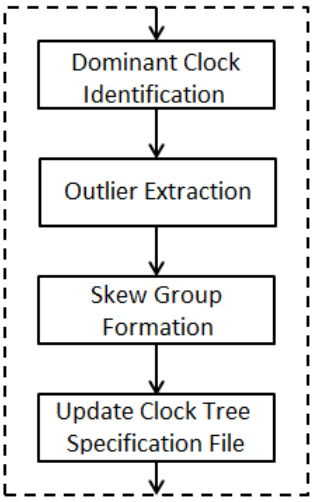
4.1 Algorithm Overview

Referring to the Fig. 4.1a, the flow of the proposed algorithm is illustrated and as shown, the CID reduction algorithm is integrated with the CTS stage. Fig. 4.1b shows the integral steps of the algorithm. The first step is to find the dominant clock in the design after the cluster CTS. The clock termed as the dominant clock is the one with either or all of the characteristics, which are, the one with maximum frequency or the one with maximum number of sinks spread in the clock tree structure or the one with the most critical path. After the analysis of clock tree and its balancing requirements, dominant clock is identified.

The next step involves the extraction of the sinks of the dominant clock which pull the CID to a maximum. The clock tree sinks which constrain the balancing requirements during CTS and increase the CID to the maximum are considered as outliers in this algorithm. After the placement is completed, a clock tree specification file is generated that captures all the design constraints. As the clock is in the propagated mode during



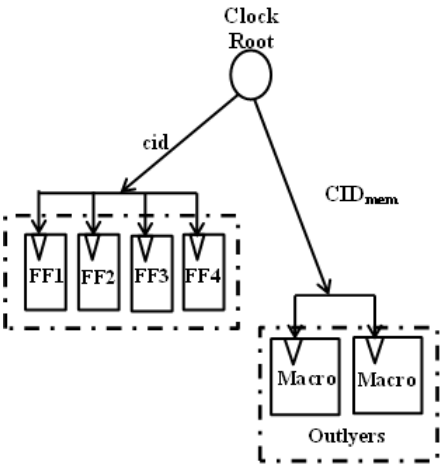
(a) Physical Design Flow with CID Reduction Algorithm



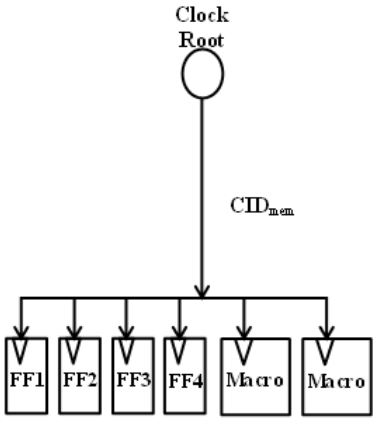
(b) Integrated Steps of Algorithm

Figure 4.1: CID Reduction Algorithm Overview

CTS, thus the constraints can be infeasible. In the Fig. 4.2, it can be seen that to balance the CID of FFs with the hard macro (memory), the CID of the FFs is increased. In Fig. 4.2a, the insertion delay of FF1-FF4 is less than that of macro blocks and in Fig. 4.2b, the insertion delay of entire block is maximum after CTS.



(a) Before CTS Balancing cid ¡ CID of FF1-FF4



(b) After CTS balancing

Figure 4.2: Outliers in Clock Tree

Innovus does not automatically perform the analysis for the reduction of insertion delay for clock nets. Hence, the algorithm filters the outliers from the clock routing tree and fixes the outliers clock tree sinks for improvement in average CID.

4.2 Steps Involved in Algorithm

In this section, the integral steps of the algorithm are explained. The sub-sections describe the method of extraction of outliers and the formation of dedicated skew-groups of outliers.

4.2.1 Extraction of Outliers for Dominant Clock

After the Timing driven placement, the next step is the clock tree prototyping using transition fixing only. This is similar to the cluster CTS as explained in the previous chapter. The purpose of transition fixing in CTS is to theoretically extract the lowest CID and to understand which clock tree sink is pulling it to maximum.

Next step is to analyze the clock tree structure and identify the dominant clock. The approach is therefore, to find the outliers that are the root cause of increment in CID. By the what-if analysis of the placement pin and the macro placement changes on the sinks with maximum CID, a group of outliers can be filtered out. In the later stage, the chip is divided into four quadrants and the outliers present in each quadrant are extracted and grouped separately. This process is the filtration process wherein, firstly a database of outliers is maintained and later on the basis of their positions and the architecture of clock tree, groups are created.

4.2.2 Skew Group formation

Once the outliers groups are determined, the outliers are grouped in separate skew groups based on their grouping in the last step. In this way, parent clock skew group's insertion delay targets are unblocked. With the help of a handful of outliers, each and every clock tree sink is satisfied with the ease of its timing targets. The previously generated clock specification file is modified automatically with the added skew groups. The modified specification file is the input for the next stage of the algorithm. Referring to the Fig. 4.2a, the new skew grouping is shown in the Fig. 4.3.

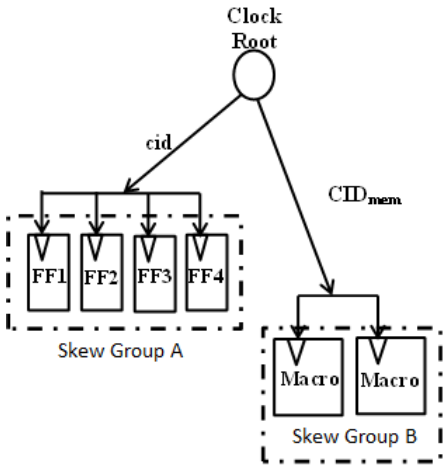


Figure 4.3: Skew Group Formation

4.2.3 Full CTS with Useful Skew Optimization

Outlier grouping leads to change in the timing paths of the outlier sinks to the other sinks. One suspected result of path grouping is the timing violations either setup or hold violations. To fix the timing violations, useful skew with CTS is conducted. This includes violating path fixing not only by the data path changes but also with launch or capture path or both clock path and the data path changes. This clock path change can happen across multiple clock paths. Hence, the CTS in this stage is with useful skew optimization.

The CTS is followed by the post-CTS optimization to further improve the setup and hold violating paths. Post-CTS fix the remaining violations of design such as DRVs and optimizes the timing paths as now the clocks are in propagated mode. The timing constraints vary for post-CTS, it includes the clock uncertainty to model jitter. Further, the routing is performed after removing the critical timing violations.

In this section, the process of filtering out the insertion delay increasing clock tree sinks is shown. The objective of outlier filtering process is to reduce the clock insertion delay for a design. As explained in the sections above the outliers are extracted from all possible cases, and then the clock tree specification file is modified. The complete CTS with useful skew optimization is performed with the updated specification file. The run-time for picking the outliers and re-run of CTS is a rewarding trade-off. Thus, it does not affect the run-time to a greater extent and improves the performance of clock tree in terms of the insertion delay.

Chapter 5

Results And Discussions

In this chapter, the results obtained in the experiment have been presented and discussed. Firstly, it provides the experimental setup of the system on which the trials have been performed. Next, specifications of design under study during the experiment are depicted. This is followed by results obtained for the proposed placement flow and the CID reduction algorithm at each step of the methodology.

5.1 Simulation Setup

The placement flow algorithm has been implemented on Red Hat Enterprise Linux v5.9 operating system using Cadence Innovus 15.2. Table 5.1 illustrates the specifications of the simulation setup and Table 5.2 design specifications on which the tests are performed.

Table 5.1: Setup for Experiment

Operating System	Red Hat Enterprise Linux Server release 6.7 (Santiago)
Number of Processing Cores	14
Vendor ID	Intel(R) Xeon(R) CPU E5-2697
Processor Speed (GHz)	2.6
Tool	Cadence Innovus(TM) v15.20-p005_1
Scripting Language	TCL

Table 5.2: Specifications of Design at 28nm Technology Node

	Design I	Design II
Total Standard Cell Number	186920	990416
Total Hard Macros	18	33
Total Chip Area	0.65 mm^2	2.07 mm^2

5.2 Comparison of Traditional placement flow without MBFFs and Proposed Timing and Clock Tree aware Placement Flow with MBFFs

The baseline flow and the proposed placement flow are shown in Fig. 5.1a and Fig. 5.1b respectively. All the inputs are provided at the beginning of the backend design flow.

Table 5.3 shows the results of the comparative analysis of various attributes performed on the two designs. Due to the applied merging algorithm and the MBFFs in the design, the number of clock tree sinks decreases in both designs by 1.5%. The number of merge-able flops for a design depends on its complexity, logic and size. As the undertaken designs are that of secured IP and complex, the merge-ability is limited.

According to the Table 5.3 for the design I and II, the number of timing violating paths reduces by 17.6% and 68% respectively which enables the designer for easy timing closure. Fig. 5.2 shows the improvement in the violating paths for design I and design II. Core density improves by 9.8% and clock power reduces by 11.8% for design I. Due to the fewer clock tree sinks, reduction in clock inverters and clock nets, the chip utilization improves. For design II, core utilization increases by 12.8% and clock power optimize by 9%.

Table 5.4 depicts the analysis of power consumption by the two approaches. Fig. 5.4 shows the power distribution and it is observed that reduction in power is by 50.46% and 37.7% respectively for two designs. The results for worst negative slack (WNS) is almost similar from the both the flows. From Fig. 5.3, the total negative slack (TNS) for setup has improved by a considerable amount for both the designs. The performance of the design is not degraded after the application of MBFFs because the CTS is performed with useful skew optimization. Similarly is for the TNS for hold mode.

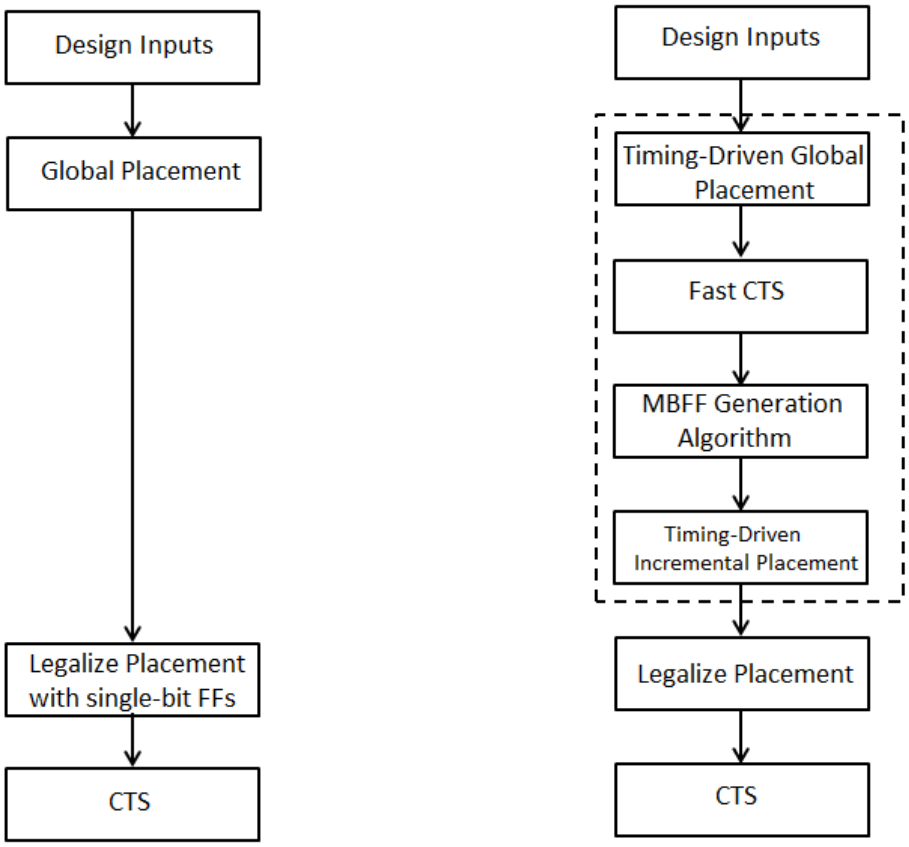
Consequently, the proposed timing and clock tree aware placement flow with MBFF generation leads to improved performance of the designs in terms of chip power by 44%, core density by 11.3% and clock power by 10.4%. Also, the corresponding FF merging algorithm proves effective by considering the real-time design scenarios.

Table 5.3: Comparison between the Non-MBFF Flow and the Proposed MBFF Flow for Design I and Design II

	Design I		Design II	
	Baseline Flow	Proposed Flow	Baseline Flow	Proposed Flow
# of 1-bit FFs	135623	72682	23586	17333
# of 2-bit FFs	0	1714	0	3125
# of Clock Tree Sinks	135623	133548	23586	20458
WNS (Setup) (ns)	-0.018	-0.02	-9.637	-9.548
WNS (Hold) (ns)	-0.001	0	-0.706	-0.447
TNS (Setup) (ns)	-0.278	-0.183	-1007.2	-566.438
TNS (Hold) (ns)	-0.001	0	-73.863	-22.183
Violating Paths	91	75	1701	546
# of Standard Cell	990416	990383	186920	182342
Total Standard Cell Area (mmˆ2)	0.854	0.853	0.17	0.16
Core Density (%)	71.415	78.402	73.314	82.702
Total WL (m)	26.22	26.23	4.2	4.1
Clock Power (mW)	37.56	33.21	7.45	6.78

Table 5.4: Comparison of Power Consumption of the designs

Power (mW)	Design I		Design II	
	Baseline Flow	Proposed Flow	Baseline Flow	Proposed Flow
Internal	130.11	119.2	31.53	26.62
Switching	342.13	114.88	54.475	31.23
Leakage	0.024	0.023	0.0126	0.0114
Total	472.27	234.108	86.0165	57.88

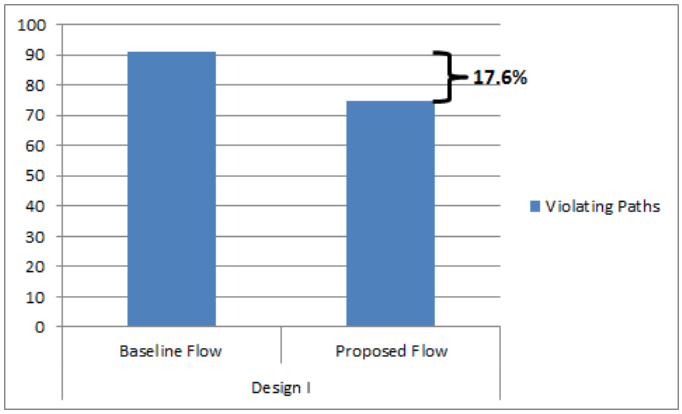


(a) Baseline Flow without MBFFs **(b)** Placement Flow with MBFFs

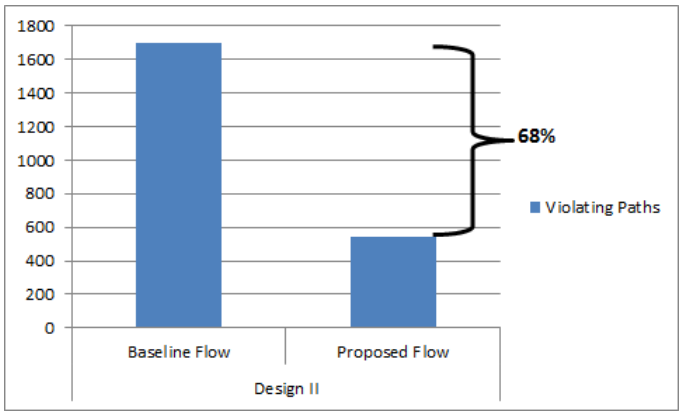
Figure 5.1: Comparison of Flows without and with MBFFs

5.3 Comparison of Proposed CID Reduction Algorithm with traditional approach of CTS

According to Table 5.5, the results of the CID reduction algorithm, average CID cuts by 19.5.%. It can be attributed to the outlier group which was excluded from the dominant clock group. It translates effectively into less CID of parent clock and all associated benefits such as controlled clock tree structure and reduced WL. Thus, with a handful of outliers, we are able to improve the rest of clock tree sinks in terms of CID. Fig. 5.5, which shows that the for the entire design after applying the proposed algorithm, the average CID reduces by 9.2%.



(a) Violating paths for Design I

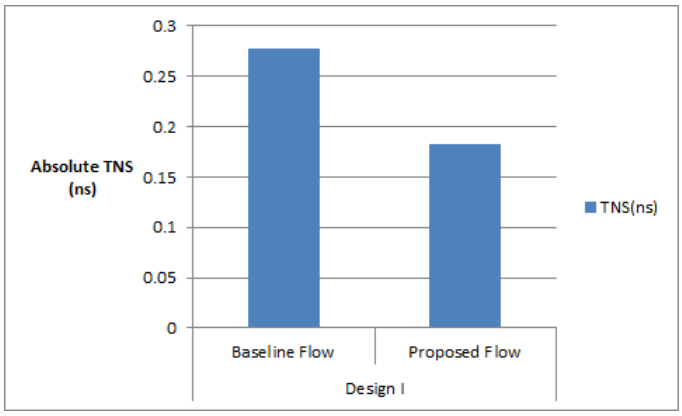


(b) Violating paths for Design II

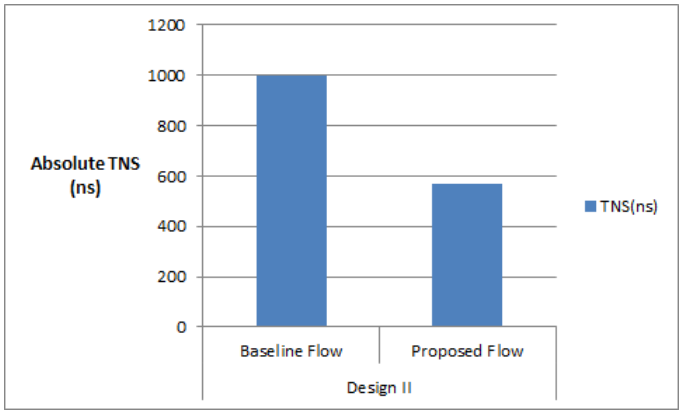
Figure 5.2: Comparison of the number of Violating Paths

Table 5.5: Experimental Results for Dominant Clock by CID Reduction Algorithm

Dominant Clock Attributes	Base Algorithm	Proposed CID Algorithm
Minimum CID (ns)	0.954	0.740
Maximum CID (ns)	1.116	0.902
Average CID (ns)	1.078	0.868
# of Inverters	1125	1078
WL after CTS (um)	201323.3	198488.836

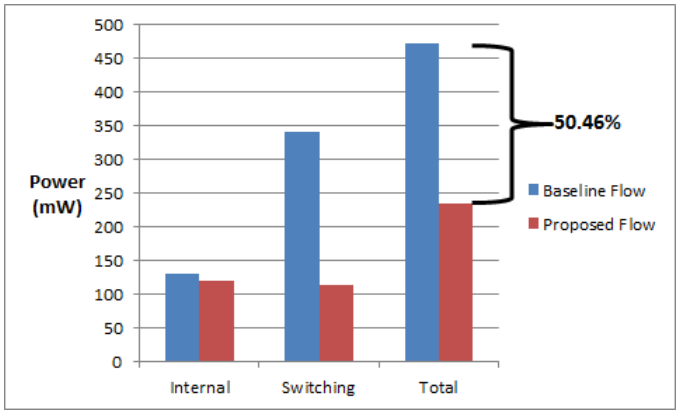


(a) TNS for Design I

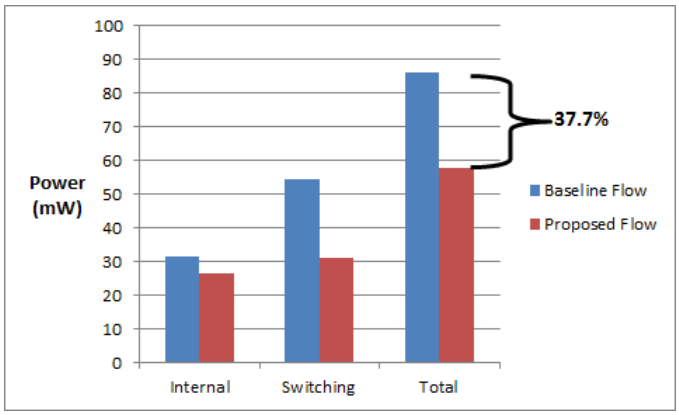


(b) TNS for Design II

Figure 5.3: Comparison of the TNS of the designs



(a) Power distribution for Design I



(b) Power distribution for Design II

Figure 5.4: Distribution of Power Consumption

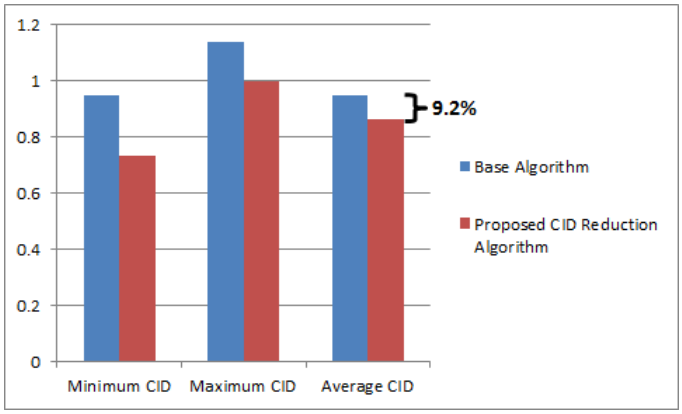


Figure 5.5: Results for Clock Network CID of design after CTS

Chapter 6

Conclusion And Future Work

6.1 Conclusion

This thesis presents an effective timing and clock tree aware placement strategy with MBFF generation and the corresponding algorithm for FF merging. It is focused on the analysis of real-time designs and tackles all complex scenarios efciently. The proposed strategy is unified with the industry EDA tool, Cadence Innovus, as a result, it allows the designer to use benchmark platform for the physical design flow. It also accelerates the timing closure of the design with the useful skew optimization technique incorporated during CTS. The thesis demonstrates the impact of MBFFs on the power, performance and area (PPA) on a SoC design.

Experimental results for the presented placement ow prove its efficiency from the non-MBFFs traditional placement flow. This approach shows the reduction in power, improvement in the utilization of the core for two designs without the degradation in the design performance. For the design I, core density increases by 9.8%, clock power decreases by 11.8% and overall total power reduces by 50.46%. For design II, core utilization increases by 12.8%, clock power optimizes by 9% and total power for design reduces by 37.7%. In the end, the algorithm for the optimization of CID is also presented which results in minimization of average insertion delay of the design. The results show 9.2% reduction in the average CID of entire design after applying the proposed algorithm. Reduction in CID infers to controlled clock structure due to the reduction in the amount of buffer or inverter in the clock path. Controlled clock tree architecture implies less crosstalk penalty and hence, the CRPR is in control.

Hence, this placement strategy seamlessly integrates with the EDA tools for the physical design process and significantly optimizes the PPA.

6.2 Future Work

6.2.1 Higher-bit MBFF

The MBFF cell library in this work includes 2-bit FFs only. With the higher bit MBFFs such as 4-bit in the MBFF library, further merging of the 2-bit FFs can be done. Integrating 4-bit FFs in the library would make the power consumption even less. The number of clock tree sinks will eventually reduce and hence the power utilization of the chip. With higher bit MBFFs than 4-bit MBFF there could be an issue with congestion in the design.

So the analysis of the proposed algorithms with higher bit MBFFs will lead to improved core density and reduced power.

6.2.2 Clock Mesh Framework

Clock meshes are robust to the process variations and hence, are used by the designers for tighter skew control, lower insertion delay and high tolerance towards OCV [33, 34]. Due to their resistance towards the variations, many EDA tools incorporated clock mesh synthesis with the physical design flow. In some designs, mesh architectures consume less power than the clock trees but in general, they can consume more power. It is preferred for high fanout clock networks as they reduce the adverse effects of OCV and improve the performance of the chip.

In this work, placement flow involves CTS methodology, but the physical design flow and the proposed FF merging algorithm can also be performed with clock mesh synthesis. The work can be extended to clock mesh framework and their power consumption can also be reduced by the proposed methodology.

The clock meshes provide lower insertion delay, and it can be further reduced with the application of the CID reduction algorithm. The results and the analysis can be done in future for the performance of design with clock meshes.

References

[1] B. H. Lorincz, Y. , X. Li, K. Mai, L. T. Pileggi, R. A. Rutenbar, and K. L. Shepard, "Digital circuit design challenges and opportunities in the era of nanoscale cmos," *Proceedings of the IEEE*, pp. 343365, Feb. 2008.

[2] H. -L. Roh, "Driving forces the technological challenges for soc development of tomorrow," *IEEE Workshop on Signal Processing Systems*, pp. 2-, Aug. 2003.

[3] S. Roy, P. M. Mattheakis, L. Masse-Navette, and D. Z. Pan, "Evolving challenges and techniques for nanometer soc clock network synthesis," *12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pp. 1-4, Oct. 2014.

[4] R. A. Bergamaschi and J. Cohn, "The a to z of socs," *IEEE/ACM International Conference on Computer Aided Design*, pp. 791798, Nov. 2002.

[5] L. Xiu, "VLSI Circuit Design Methodology Demystified:A Conceptual Taxonomy, Springer, 2008.

[6] J. Lu and B. Taskin, "From RTL to GDSII: An ASIC design course development using Synopsys #x00AE University Program," *IEEE International Conference on Microelectronic Systems Education (MSE)*, pp. 7275, June 2009.

[7] S. N. Adya and I. L. Markov, "Power distribution network design for VLSI," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 11201135, Dec. 2003.

[8] Q. K. Zhu, "Fixed-outline floorplanning: enabling hierarchical design, John Wiley & Sons, Dec. 2003.

[9] B. Halpin and N. Sehgal and C. Y. R. Chen, "Detailed placement with net length constraints," *The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications*, pp. 22-27, June 2003.

[10] L. N. Kannan and P. R. Suaris and Hong-Gee Fang, "A Methodology and Algorithms for Post-Placement Delay Optimization," *31st Conference on Design Automation*, pp. 327-332, June 1994.

[11] A. Balboni, C. Costi, M. Pellencin, A. Quadrini, and D. Sciuto, "Clock skew reduction in asic logic design: a methodology for clock tree management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 4, pp. 344356, Apr. 1998.

[12] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 2329, Jul. 1999.

[13] M. Dietrich and J. Haase, "Process Variations and Probabilistic Integrated Circuit Design, Springer Science & Business Media, 2011.

[14] P. B. Ghate, "Electromigration-induced failures in vlsi interconnects," *20th Annual Reliability Physics Symposium*, pp. 292299, March 1982.

[15] T. B. Chan, A. B. Kahng, and J. Li, "Nolo: A no-loop, predictive useful skew methodology for improved timing in ic implementation," *Fifteenth International Symposium on Quality Electronic Design*, pp. 504509, March 2014.

[16] H. M. Chou, H. Yu, and S. C. Chang, "Useful-skew clock optimization for multipower mode designs," *IEEE/ACM International Conference on Computer Aided Design (ICCAD)* , pp. 647650, Nov. 2011.

[17] X. Man and S. Kimura, "Comparison of optimized multi-stage clock gating with structural gating approach," *IEEE Region 10 Conference in TENCON*, pp. 651656, Nov. 2011.

[18] M. P. H. Lin, C. C. Hsu, and Y. T. Chang, "Recent research in clock power saving with multi-bit flip-flops," *IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 14, Aug. 2011.

[19] G. Prakash, K. Sathishkumar, B. Sakthibharathi, S. Saravanan, and R. Vijaysai, "Achieveing reduced area by multi-bit flip flop design," *International Conference on Computer Communication and Informatics (ICCCI)*, pp. 14, Jan. 2013.

[20] Innovus User Guide.

[21] S. K. Teng and N. Soin, "Low power clock gates optimization for clock tree distribution," *11th International Symposium on Quality Electronic Design (ISQED)*, pp. 488492, March 2010.

[22] M. P. Dev, D. Baghel, B. Pandey, M. Pattanaik, and A. Shukla, "Clock gated low power sequential circuit design," *IEEE Conference on Information Communication Technologies (ICT)*, pp. 440444, April 2013.

[23] S. Y. Chen, R. B. Lin, H. H. Tung, and K. W. Lin, "Power gating design for standard-cell-like structured asics," *Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 514519, March 2010.

[24] W. M. D. J. G. Xi, "Buffer insertion and sizing under process variations for low power clock distribution," *32nd Conference on Design Automation (DAC)*, pp. 491496, 1995.

[25] J. T. Yan and Z. W. Chen, "Construction of constrained multi-bit flip-flops for clock power reduction," *International Conference on Green Circuits and Systems (ICGCS)*, pp. 675678, June 2010.

[26] Y. T. Shyu, J. M. Lin, C. P. Huang, C. W. Lin, Y. Z. Lin, and S. J. Chang, "Effective and efficient approach for power reduction by using multi-bit flip-flops," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 4, pp. 624635, April 2013.

[27] C. Santos, R. Reis, G. Godoi, M. Barros, and F. Duarte, "Multi-bit flip-flop usage impact on physical synthesis," *25th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pp. 16, Aug. 2012.

[28] M. P. H. Lin, C. C. Hsu, and Y. T. Chang, "Post-placement power optimization with multi-bit flip-flops," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 12, pp. 18701882, Dec. 2011.

[29] H. Moon and T. Kim, "Design and allocation of loosely coupled multi-bit flip-flops for power reduction in post-placement optimization," *21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 268273, Jan. 2016.

[30] W. Chen and J. T. Yan, "Routability-driven flip-flop merging process for clock power reduction," *IEEE International Conference on Computer Design (ICCD)*, pp. 203208, Oct 2010.

[31] C. C. Hsu, Y. C. Chen, and M. P. H. Lin, "In-placement clock-tree aware multibit flip-flop generation for power optimization," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 592598, Nov. 2013.

[32] M. P. H. Lin, C. C. Hsu, and Y. C. Chen, "Clock-tree aware multibit flip-flop generation during placement for power optimization," *IEEE Transactions on ComputerAided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 280292, Feb. 2015.

[33] P. Chakrabarti, V. Bhatt, D. Hill, and A. Cao, "Clock mesh framework," *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, pp. 424431, March 2012.

[34] A. Rajaram and D. Z. Pan, "Meshworks: An efficient framework for planning, synthesis and optimization of clock mesh networks," *Asia and South Pacific Design Automation Conference*, pp. 250257, March 2008.