PREDICTIVE CLOCK SKEW REDISTRIBUTION METHODOLOGY FOR IMPROVING TIMING QoR

M.Tech. Thesis

By

PRANSHU BISHT

DISCIPLINE OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JUNE, 2019

PREDICTIVE CLOCK SKEW REDISTRIBUTION METHODOLOGY FOR IMPROVING TIMING QoR

A THESIS

Submitted in partial fulfillment of the
requirements for the award of the degree
of
Master of Technology
in
Electrical Engineering
with specialization in
VLSI Design and Nanoelectronics

by
PRANSHU BISHT

DISCIPLINE OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
JUNE, 2019

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled "**PREDICTIVE CLOCK SKEW REDISTRIBUTION METHODOLOGY FOR IMPROVING TIMING QoR"** in the partial fulfillment of the requirements for the award of the degree of **MASTER OF TECHNOLOGY** and submitted in the **DISCIPLINE OF ELECTRICAL ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from JULY, 2017 to JUNE, 2019 under the supervision of Dr. Santosh Kumar Vishvakarma, Associate Professor, Indian Institute of Technology, Indore and Shrikrishna Nana Mehetre, Sr. Engineering Manager, Seagate Technologies HDD (India) Pvt Ltd.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Signature of the student with date
PRANSHU BISHT

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

Signature of the Supervisor of M.Tech. thesis #1 (with date)	Signature of the Supervisor of M.Tech. thesis #2 (with date)
Dr. SANTOSH KUMAR VISHVAKARMA	**SHRIKRISHNA NANA MEHETRE**

PRANSHU BISHT has successfully given his/her M.Tech. Oral Examination held on **24-Jun-2019**.

Signature(s) of Supervisor(s) of M.Tech. Thesis Date:	Signature of Convener, DPGC Date:
Signature of PSPC Member #1 Date:	Signature of PSPC Member #2 Date:

Dedicated To my family and friends

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my extreme gratitude to my research supervisor **Dr. Santosh Kumar Vishvakarma,** Associate Professor IIT Indore and **Shrikrishna Nana Mehetre** Sr. Engineering Manager at Seagate Technologies HDD (India) Pvt Ltd. At many stages in the course of this research project I benefited from their advice, particularly so when exploring new ideas. Their positive outlook and confidence in my research inspired me and gave me confidence.

A project of this nature, based on both experiment and theoretical work, is only possible with the help of many people. In particular, I would like to thank **Mahendra Singh Thakur** my mentor at Seagate for helping me with the experiments.

The many hours that I spent at the college have been very stimulating and enriching, and thanks to the wonderful that I have been privileged to interact with. Particularly, I would like to thank **Dr. Vishal Sharma** Sr. R&D Engineer at Synopsys, **Abhishek Dalal**, and **Pallab Nath** for the great discussions that we had in the lab and for useful comments that improved my work significantly.

I would like to thank the great people of the **Seagate** community specially **Jay Shah, Hemant Wadhavankar, Abhijit Jawkar** and **Gaurav Jadhav.** The divine inspiration from the Tech Events and study session gave me the confidence and power to stand up against the difficulties that I faced in the course of my Project.

At last, I would like to thank **IIT Indore** for providing the essential research facilities for my research work. I gratefully acknowledge the contribution of all the faculty members and staff of Electrical Engineering for their help and support. I would also like to express my sincere gratitude to Ministry of Human Resource and Development (MHRD), Government of India for providing the Teaching Assistantship during the course of M.Tech.

ABSTRACT

In modern technology with shrinking size and high speed requirement there is a need of an efficient design flow. The traditional method of synthesis doesn't take into account of clock skew scheduling that's going to be happening in Clock Tree building and optimization stage. In order to meet the timing specification synthesis, tries to optimize the data path. Although this method will improve setup violation but impact area and power on the overall design. On the other hand, positive slack distributed in our design is not utilized. To solve this issue many have proposed a Back-annotation [BA] flow. In BA, useful skew number is retrieve from CTS stage and feedback either to CTS or synthesis stage. This long feedback loop from the implementation tool to synthesis not only impact the design time cycle but also result in an inefficient result as the implementation is already done keeping data path optimization in mind.

In this project we will discussed a methodology on how this positive slack can be retrieved by adjusting the clock skew at synthesis stage and redistributed along the most critical path in our design. The proposed algorithm analyze slack across multiple depth of a timing path and redistribute the available slack by adjusting the clock skew of capture or launch registers to the most critical register pair. The clock skew adjustment script which will be our output will be passed down to physical implementation tool at placement and clock tree synthesis stage so that skew aware placement and clock tree build happen which will result into better timing correlation between the synthesis tool and implementation tool.

The proposed system leads to improve runtime and timing QoR of design. Power and area reduction over conventional method of zero skew synthesis is also observed. Moreover, the flow is generic which means that whenever there is a technology shift no reconfiguration is required as the flow is independent on the technology parameter.

LIST OF PUBLICATIONS

- **Peer Reviewed International Journal**
  - Vishal Sharma, Pranshu Bisht, Abhishek Dalal, Maisagalla Gopal, Santosh Kumar Vishvakarma and Shailesh Singh Chouchan "Half-select free bit-line sharing 12T SRAM with double-adjacent bits soft error correction and reconfigurable FPGA for low-power applications",International Journal of Electronics and Communication, Feb. 2019
- **Conferences**
  - Vishal Sharma, Pranshu Bisht, Abhishek Dalal, Shailesh Singh Chouhan, H. S. Jatana and S. K. Vishvakarma, "A Write-Improved Half-Select-Free Low-Power 11T Subthreshold SRAM with Double Adjacent Error Correction for FPGA- LUT Design," 22nd International Symposium on VLSI Design and Test (VDAT), 28th-30th June, 2018, Tamilnadu, India.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations

ASIC	Application Specific integrated circuit
BA	Back annotation
CTS	Clock Tree Synthesis
CSS	Clock Skew Scheduling
PT	Prime Time
SoC	System On Chip
Nm	Nanometer
WNS	Worst Negative Slack
TNS	Total Negative Slack
NVE	Number of Violating Endpoint
GhZ	gigahertz
RTL	Register Transfer Level
DC	Design Compiler
HDL	Hardware Descriptive Language
PnR	Placement and Routing
PNS	Power Network Synthesis
LVT	Low Threshold Voltage
HVT	High Threshold Voltage
SVT	Standard Threshold Voltage
MMWC	Maximum Mean Weight Cycle
QoR	Quality of Result
FFx	Flip flop
STA	Static Timing Analysis

Chapter 1

INTRODUCTION

In a sequential circuit, Timing is the main concern of a digital design engineer. As technology is moving into smaller node and demand for a high performance system keep pushing the clock frequency into GHz region, the conventional method of achieving the required timing target is no longer efficient. Apart from functional designing, timing closure is one of the important milestones on which heavy time is being spent which govern when a chip can be released to the semiconductor foundry for fabrication. So any effort which can help reduce the effort on timing closure will be of great importance.

The conventional method of synthesis, assume ideal clock network as clock tree in not build yet. This approach of ideal clock network is too optimistic which result in data path optimization for setup violation. As we go further into the implementation stage, this approach leads to penalty of area and power. Also, the positive slack available at multiple stages is not properly utilized which lead to underuse resources. Many researchers have started to address the need of clock skew at synthesis stage. They started to provide feedback from implementation stage or post synthesis stage back to synthesis stage. However, this method is inefficient as tool has already mapped the design and data path optimization is performed. This led to a classic chicken-egg problem. In further Chapter 3, we have tried to address this issue.

1.1 ASIC DESIGN FLOW

ASIC (Application Specific Integrated Circuit) are designed for specific applications. Examples of ASCI design include IoT devices, chip for satellite, mobile chip and interface chip etc. At Seagate ASIC is used for designing of flash controller. ASIC design cycle starts with a functional specification of a chip followed by RTL logic, synthesis, Implementation and STA. The typical design cycle can be shown by figure 1.1.

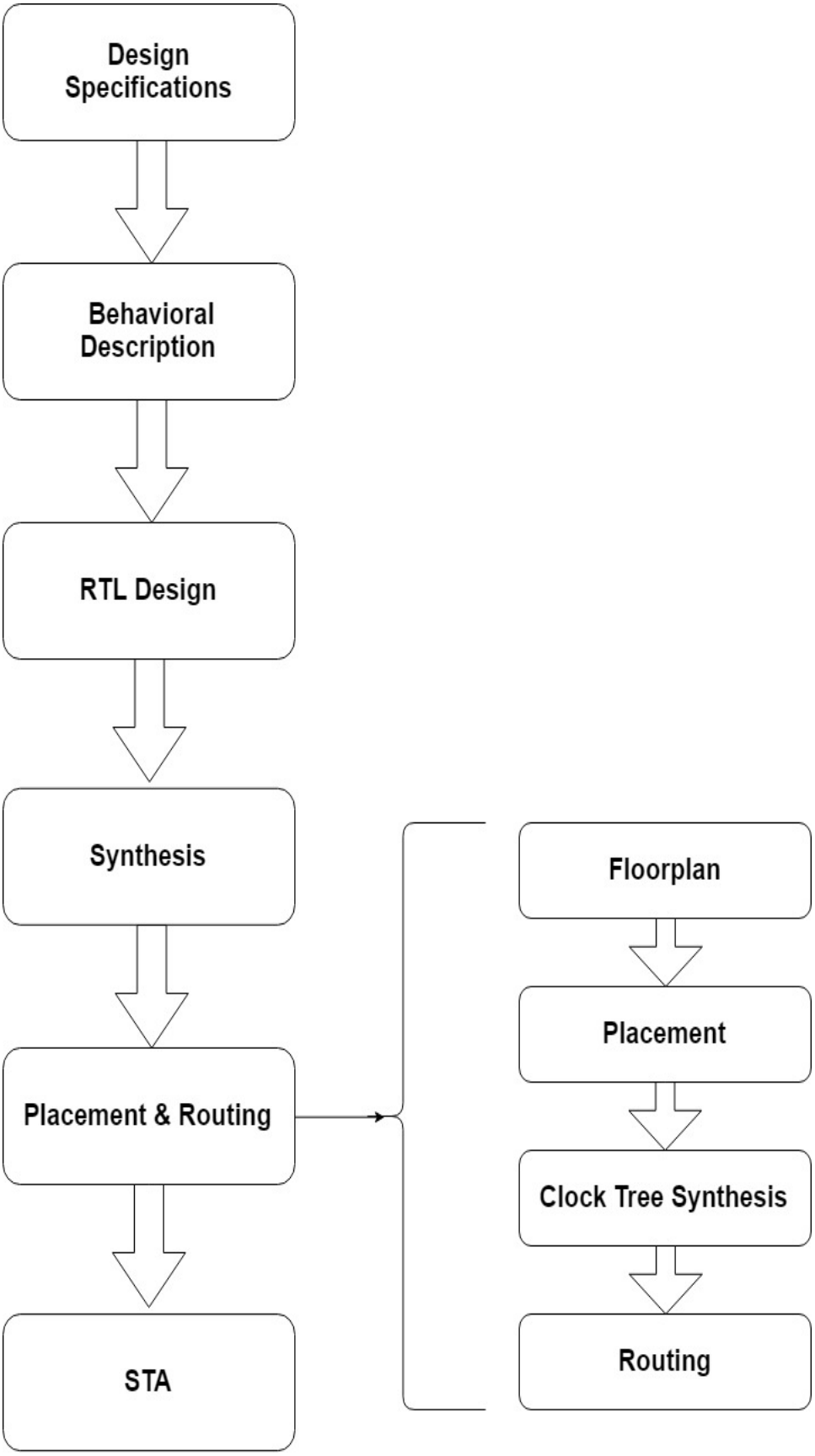


Figure 1.1 ASIC FLOW

1.1.1 Design Specification

To design a chip, one first lay down the specification of the system. Specifications are nothing but a high level representation of the system which include Functionality i.e. what will our designed chip do, Performance parameter like speed and power, Physical dimension, Technology constraint & Fabrication technology etc.

1.1.2 Behavioral or Functional Description

In this step, architecture of the design is decided. This includes whether to use RISC (Reduced Instruction Set Computer) versus CISC (Complex Instruction Set Computer), ALUs, pipelining, Floating Point units and size of caches etc. The functionality of the design is broken into small pieces with a clear understanding about different block implementation. The outcome of functional design is usually a timing or relationship between the various subsystems of a large block. This allows for a fast debugging of the full system. Behavioral design is largely a manual step and working environment is documentation.

1.1.3 Register Transfer Level

This step is basically the detailed logic implementation of the entire design. This is where the detailed description such as arithmetic operation, logic operation, controls flows and register allocation that represent a functional design are derived and tested. The subsystems which are defined at top level are implemented using logic representation, finite state machine, combinational and sequential logic etc. This description is called Logic Design or Register Transfer Level (RTL) description. RTL is usually expressed in a Hardware Description Language (HDL) such as VHDL or Verilog. This description can be used for simulation and verification. Nowadays another HDL language System Verilog is gaining popularity for verification.

1.1.4 Synthesis

Synthesis is a process of converting a design description written in a hardware description language such as Verilog or VHDL into an optimized gate level netlist mapped to a particular technology library. Implementation tool are not advance enough to derive the physical representation from the logical representation. To bridge the gap between logical and physical representation Synthesis is done. We have used Design Compiler (DC) tool from Synopsys for synthesis of our design. Constraint such as timing, area and power are considered for the optimization of the design. The synthesis tool try to meet required constraint criteria based Total cost of the design. At the end, we will also verify whether the gate level netlist obtained at the end of synthesis match the functional behavioral of design. For that we used the formality tool which does functional verification.

The next in ASIC flow is the physical implementation of the optimized gate level netlist. The Gate level Netlist is converted into geometric representation using an implementation tool such as Encounter, IC compiler etc.

1.1.5 FLOORPLAN

This is the first major step in PnR (Placement and Routing) to get our layout done. Goal of floorplan is to provide best possible seed to the placement stage. At floorplan stage we decide the area of the core, location of Input output pads, type of power distribution. Every subsequent stage like placement, Clock tree and routing performance depend on optimum floorplan. Maximum effort are put into floorplan to create a optimize seed for the placement stage. As good as our floorplan will be the subsequent stage result will be better. The core area of the floorplan can be decided by using the total area of cell available in the netlist.

- Goals of floor planning:
  - Place the Macros on a chip.
  - Decide the location of the I/O pins.
  - Decide the type of power distribution.
- Objectives of floor planning are:
  - To minimize the chip area
  - Minimize delay

- Minimize global congestion (routable)

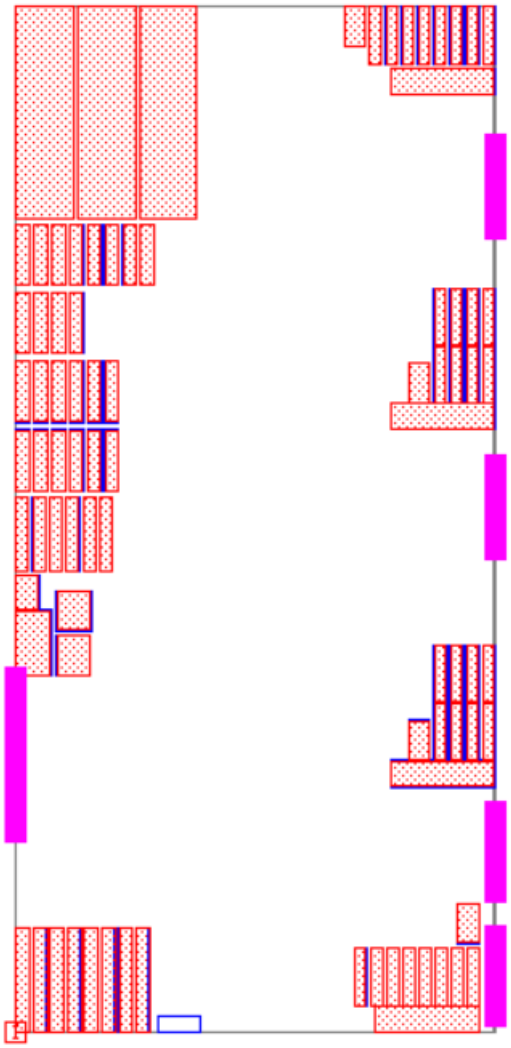


Figure 1.2 Floorplan

1.1.6 Power Network Synthesis

Power Network Synthesis (PNS) is done so that the power is distributed evenly in our design to each and every corner. To distribute power straps and rails are distribute in the design. Physical cells such as TAP, ENDCAP, DECAP cell are also included in the design. Power planning is done by introducing switches and creating power region to save power consumption. Power gating can also be implemented at this stage for further optimization.

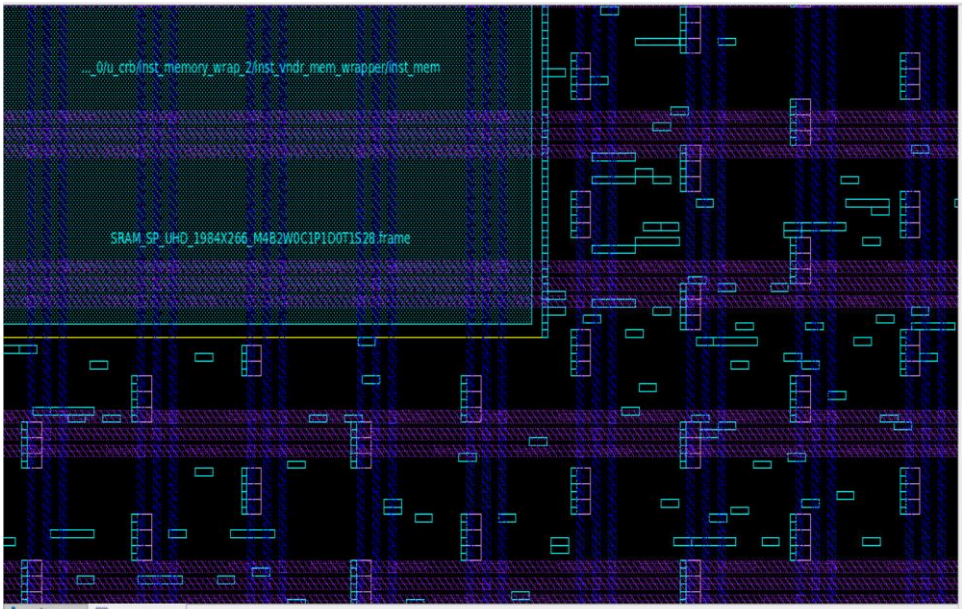


Figure 1.3 Power Network Synthesis

1.1.7 Placement

At placement we assign cells to a positions on the chip, which is basically knows as site rows. The main objective is to reduce area and interconnect cost. Placement stage also decides the routability of the design.

Goal of Placement

I. Timing, Power and Area optimizations
II. Routable design (minimal global & local congestion)
III. No/minimal cell density, pin density & congestion hot-spots

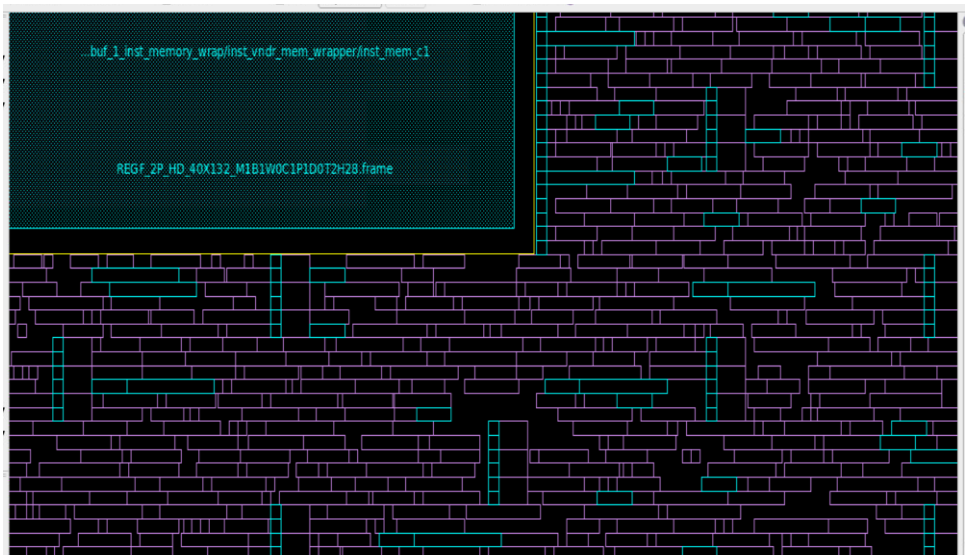


Figure 1.4 Placement

1.1.8 Clock Tree Synthesis

CTS is a process in which clock gets distributed throughout the design. The goal of CTS is to minimize the skew so that maximum frequency of the design can be properly utilized. Cell used in clock nets are LVT (Low Voltage Threshold) cell as these nets are most timing critical.

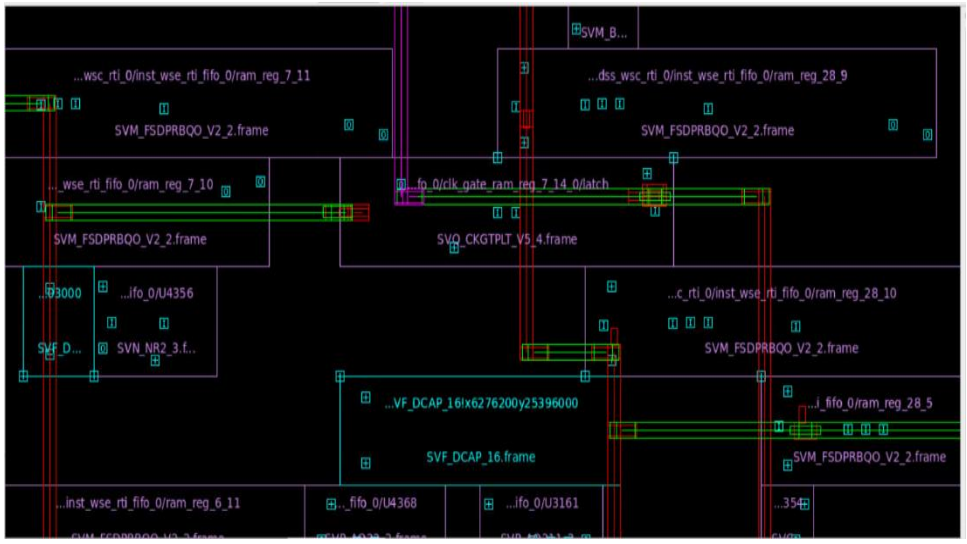


Figure 1.5 Clock Tree Synthesis

1.1.9 Routing

Routing is the process of physically connecting every pins and instance in our design. During routing we make sure that crosstalk effect doesn't degrade the system performance.

Figure 1.6 Routing

1.1.10 Static Timing Analysis

Timing analysis is the integral part of ASIC flow. Anything can be compromised but not timing. Timing should be met before sending a chip for fabrication. Static Timing Analysis is used to determine whether a design is meeting all timing constraint for that STA checks all possible path for timing violation. STA is much faster than timing-driven, gate-level simulation and doesn't check the functionality of the design. Static timing analysis tool work by dividing the design into different timing paths. Then propagation delay of each path is calculated. Then these delay is compared to the required maximum and minimum values.

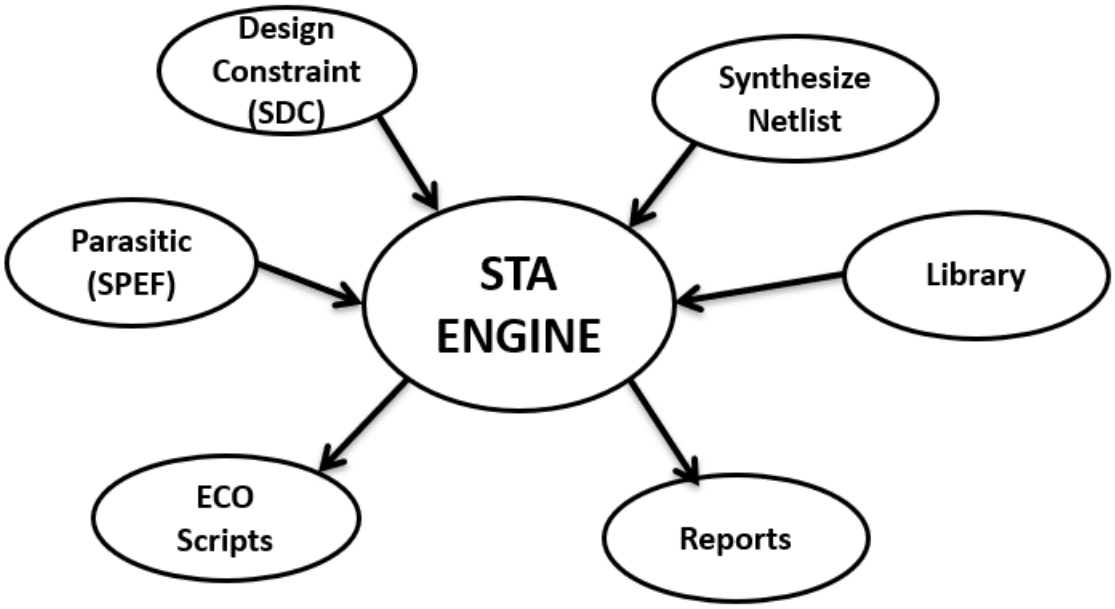


Figure 1.7 Static Timing Analysis.

In figure 1.7, we can see that STA Engine gives Reports and ECO scripts as an output. ECO script can further be used back in implementation tool to close timing and meet the design specification.

1.2 Definitions

1.2.1 Clock Skew

Clock Skew is defined as the time difference between the arrivals of clock signal at two different point is our design. Conventionally we try to minimize the clock skew in the design so that max utilization of frequency can be done.

1.2.2 Setup Time

Setup Time is defined as minimum amount of time for which data should be available at the capture edge of the clock so that the data can be perfectly latched. Any violation in setup time can cause an imperfect data to latch into the system and can impact the functionality of our design.

1.2.3 Hold Time

Hold Time is defined as the minimum amount of time after clock edge for which data should be held stable so it can be perfectly latched. Hold Time is the most critical of all timing violation as hold violation depend upon the data path delay.

1.2.4 Positive Slack

Slack is define as the difference between the data required time and the data arrival time of a signal. For a timing path slack determine the desired frequency. In Setup, Positive slack is when the data is arriving early than the required time.

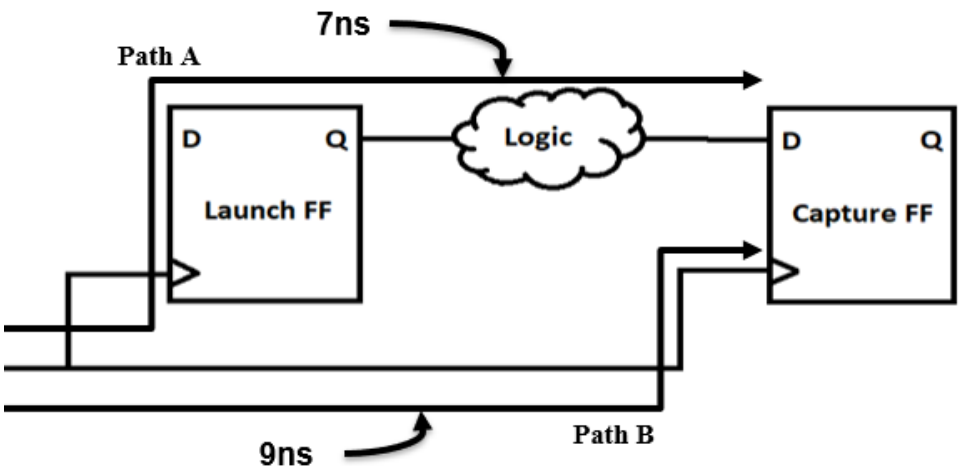


Figure 1.8 Positive Slack

In figure 1.8, Path A is the launch path and Path B is the capture path. Arrival time of a signal to reach capture flop from launch flop is 7ns and the required time is 9 ns. So the slack of our timing path is 2ns which is knows as positive slack. This positive slack can be used to the other critical path of the design.

1.2.5 Negative Slack

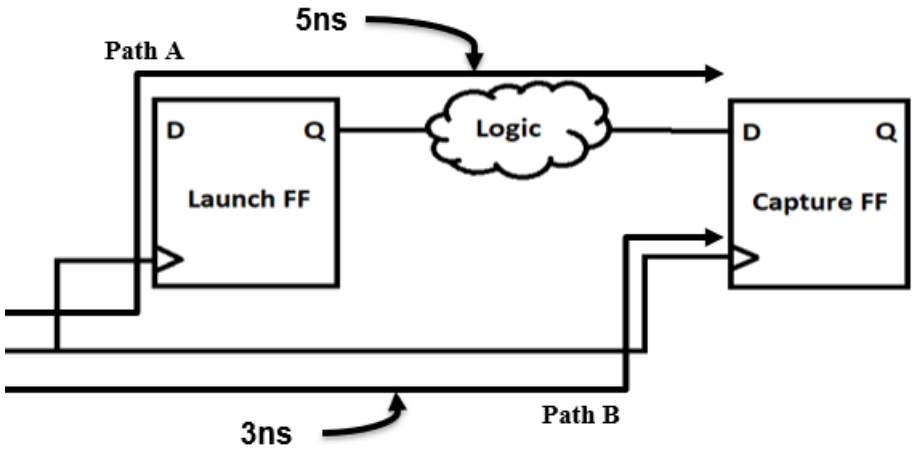


Figure 1.9 Negative Slack

In setup, negative slack is when the data arrival time is greater than the data required time. In figure 1.9 data arrival time is 5ns and data required time is 3 ns which create a negative slack of -2ns. In timing path, we aim to improve this negative slack. Chip will be fabricated only when there are no negative slack path.

1.2.6 Positive Skew

Positive skew is defined as when the clock edge on the capture flop come late than the launch flop than it is called positive skew. In positive skew clock and the data are traveling in the same direction. Positive skew can help in removing the setup violation but can degrade the hold violation. Clock and data are routed in the same direction than it is called +ve skew.

1.2.7 Negative Skew

Negative skew is defined as when the clock edge come early in capture flop than the launch flop than it is called negative skew. In negative skew clock and data are travelling in the opposite direction. Negative skew can help in removing the hold violation but can degrade the setup violation. Clock and data are routed in the opposite direction than it is called -ve skew.

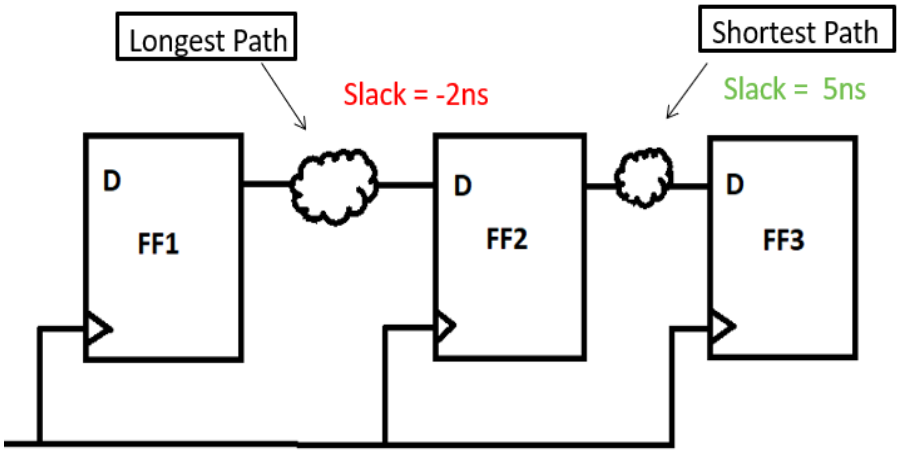


Figure 1.10 Timing Path before skew implementation

1.2.8 Useful skew

If we intentionally try to add skew in our design so that the violating timing path can be met is knows useful skew. This skew is borrowed from the adjacent stages and redistributed in the most critical path to meet timing of the design.

In figure 1.10, we have two flip flop pair FF1-FF2 and FF2-FF3. FFI-FF2 being the longest path has timing violation of 2ns. FF2-FF3 is the shortest path is meeting setup timing by 5ns. In useful skew we can borrow the positive 2ns from the FF2-FF3 pair and can distribute it to the FF1-FF2 pair to remove the timing violation. This can be done by adding a additional buffer for delay to the clock pin of FF2. By doing so we can reduce the timing violation is path 1 and path 2 will still be meeting by 3ns.

In figure 1.11 we have add the additional delay Td to clock pin of FF2 so that FF1-FF2 timing is met without effecting the timing of FF2-FF3.

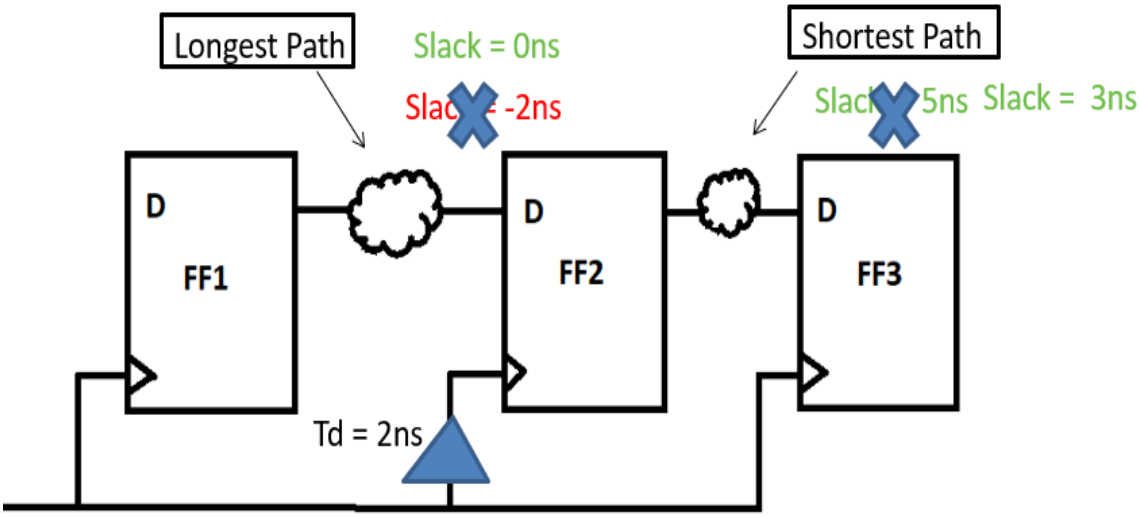


Figure 1.11 Timing path after Useful Skew.

1.3 Timing Path

In our design, timing path can be divided into four path mainly R2R, I2O, I2R and R2O.

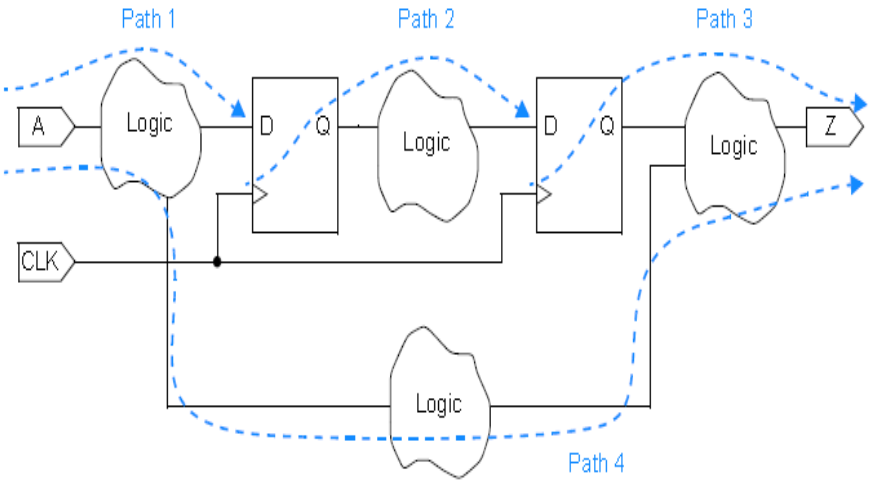


Figure 1.12 Timing Path[15]

I. R2R: R2gister to Register (Path 2)

II. I2O: Input to Output (Path 4)

III. I2R: Input to Register (Path 1)

IV. R2O: Register to Output (Path 3)

1.4 Motivation

In conventional ASIC flow, synthesis tool maximize his effort to reduce the setup time violation. To remove setup time violation tool used to optimize the data path. Data path optimization lead to increase in cell size or addition of buffer and invertor. The positive slack available in adjacent flop pair are not utilize. Also the synthesis tool is not aware of the clock skew scheduling that going to be happen in the clock build stage. Again in the placement stage, setup violation are removed by optimizing the data path based on the wirelength delay. This optimization lead to unnecessary addition of buffer and invertor. This lead to underutilize adjacent positive slack in flop pair. By doing data path optimization and not utilizing the positive slack, overall runtime of our design is also increased.

To make the synthesis tool aware of the clock skew scheduling there is a need of a new design flow. In this Thesis we have tried to address this issue by proposing a predictive clock skew redistribution technique which result in improving the timing of our design as well as the overall QoR. We are saying it as a predictive method because the clock tree is not available in the synthesis stage and the skew is done based on the data path optimization. The output skew adjustment of our script is feed to the placement stage for skew aware placement. The proposed method is unidirectional as no feedback is used in our design. This result in improving the runtime of our design as well.

1.5 Organization of Thesis

In Chapter 2 we have discussed in detailed about the different predictive skew methodology available and how a long feedback impact the runtime of our design. Chapter 3 present a proposed methodology that predict useful skew from synthesis stage and distribute the positive slack to the most critical path. Chapter 4 presents result and discussions section. In which we have presented result of proposed method and

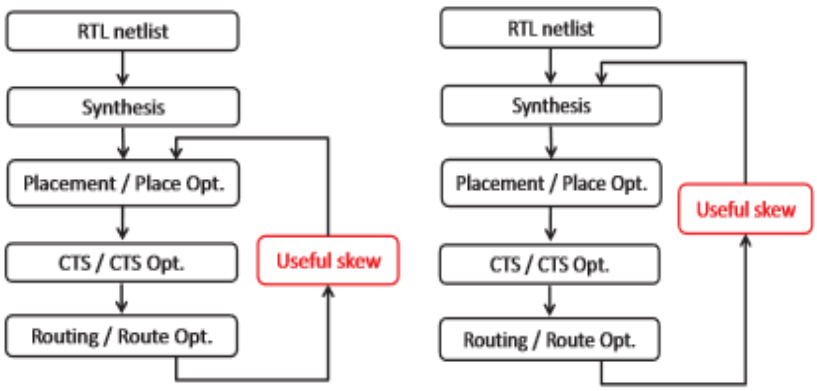
compared it to the conventional ASIC flow. Chapter 5 represent the conclusion of the whole work and future scope by which we can further refine our results.

Chapter 2

Literature Survey and Problem statement

2.1 Literature Survey

In traditional industrial ASIC, to remove timing violation back-annotation flow is used. The back-annotation flow can have many variant. In BA, feedback is used. This feedback is provided from Post-Clock tree synthesis stage either to the implementation tool or the synthesis tool. The long feedback result in increased run time. Not only the loop is ineffective but also the timing violation at the synthesis stage doesn't correlate with the violation at the end of implementation run. Synthesis tool and Placement stage invest a lot of effort in removing setup time. To remove setup time, data path optimization is done which lead to adding buffers or invertors, resizing of cell and VT(Threshold Voltage) group swapping. Now in BA flow, the useful skew values are retrieved when the data path optimization is already done. These skew values are not efficient and can only be taken for the most critical paths.



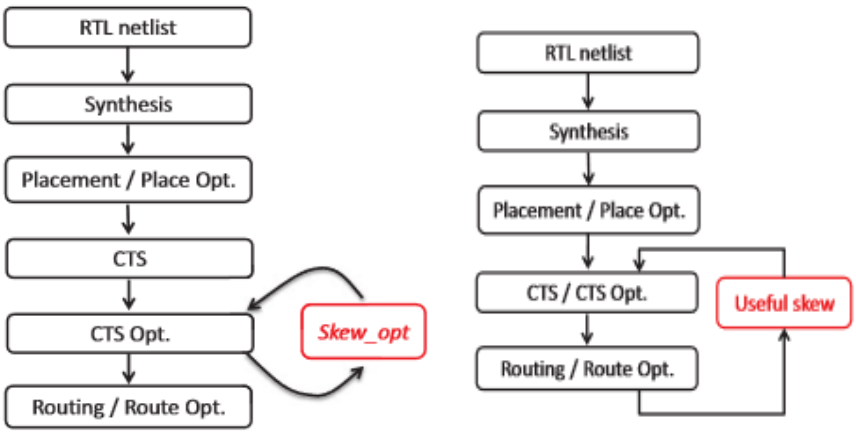


Figure 2.1 Back Annotation Flow[5]

Seungwon Kim[8] has proposed a unidirectional flow to overcome this long feedback loop. In which predictive useful skew Methodology is implemented with the incremental timing driven placement optimization stage. Timing driven placement(TDP) improves setup time violation and is not aware of the clock skew optimization next stage. To make placement aware of the clock skew optimization in the next stage author proposes to use a predictive useful skew methodology with hybrid legalization method for better placement result.

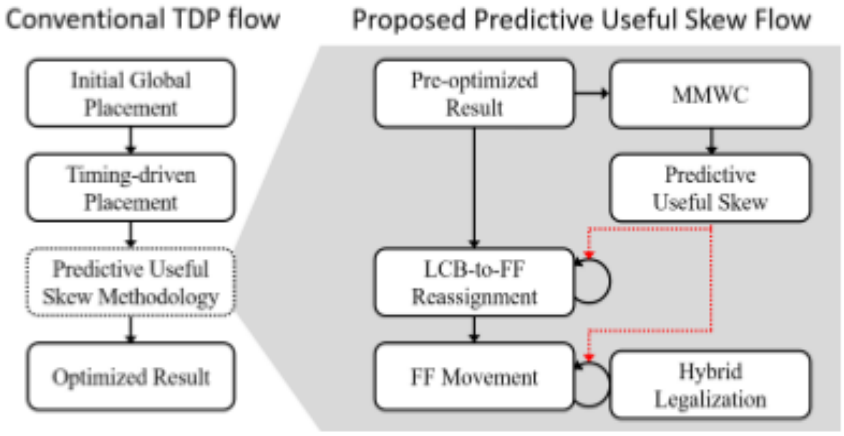


Figure 2.2 A predictive useful skew implementation in the conventional timing-driven placement flow.[8]

Using MMWC (maximum mean weight cycle)[13] optimal clock latency of each flip flop is obtained. Based on the clock latency value, Local clock buffer(LCB) are assign

to the flops and additionally flip flop and LCB are placed in a tangential position. The placement of LCB and Flipflop are changed depending on the wirelength and timing summary. The position is further optimized by using a proposed hybrid legalizer. After that STA is done to further verify and optimize the result. By doing so placement is aware of the clock skew optimization on next stage. The drawback is that the synthesis tool is not aware about these skew value and will try to optimize the result based on data path information which lead to underutilized resources.

Tuck-Boon Chan[5], proposed a method which applies useful skews at the post synthesis stage which improves the timing correlation between the post synthesis stage and the implementation stage. Author used the MMWC method to retrieve the optimum skew values and feed this value to the placement stage. Author also proposed an additional run of synthesis with LVT libraries only to predict the useful skew based on two synthesized netlist. The result shows that the single pass flow achieve better timing compared to the back annotation flow.

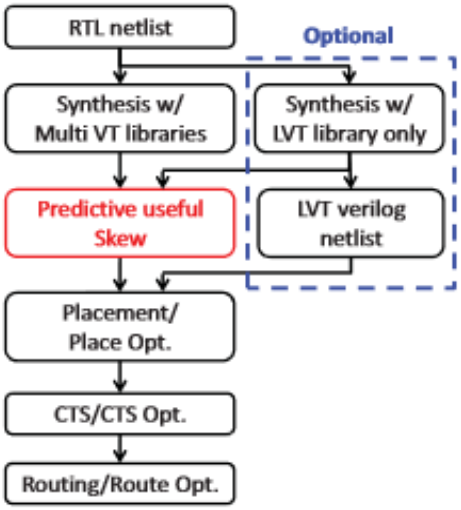


Figure 2.3 Predictive NOLO (“no-loop”) useful skew flow [5]

NOLO in figure 2.3 address the issue of making the implementation flow aware of clock skew optimization using the clock skew scheduling. The predictive skew methodology based on MMWC approach to find an optimum clock skew value for flop’s. This method will reduce the run time as the most critical path can easily be

recovered using the clock skew and implementation tool will have a lesser path for optimization. This indeed result in improving the QoR and better run time compare to the conventional industrial flow. Still the front end, synthesis tool is not aware about these optimization. In next paper author has address this issue.

In paper [1], author has address the issue of making the front end tool aware of the clock skew scheduling in implementation run. Author used the Synopsys Physical compiler to generate a physical aware placed-netlist. The useful skew is obtained using the Extensive slack balance algorithm. These skew value are feedback to synthesis tool using a SDC file and then again a re-synthesis is done.

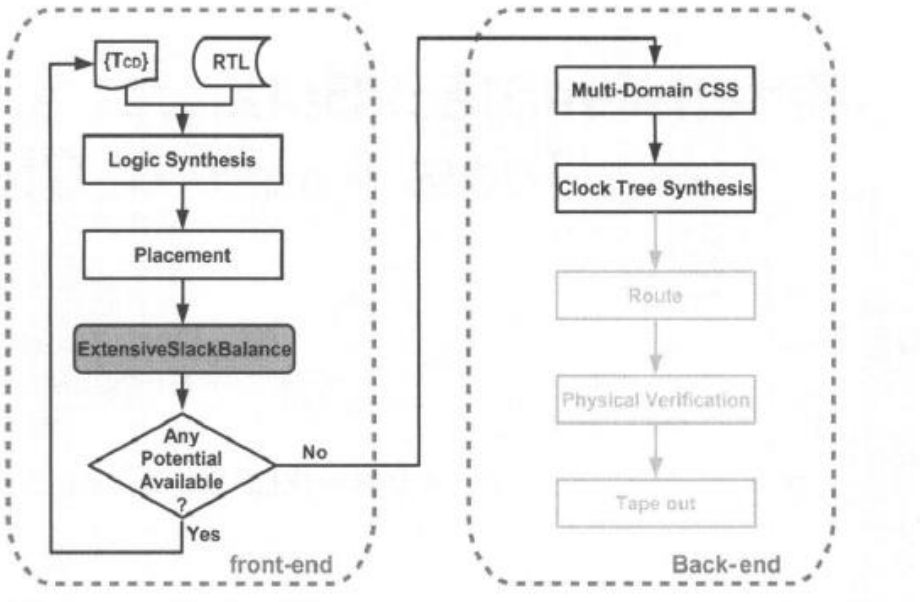


Figure 2.4 ExtensiveSlackBalance: an Approach to make Front-end tool aware of clock skew scheduling.[1]

To generate the placement author used the design compiler in topographical mode and based on that predictive useful skew values are extracted for all the critical paths. These skew value are again feed to the synthesis tool for another iteration. This process is repeated until desirable result are not obtained.

2.2 Problem Statement

Typical run time for a SoC at 28nm technology node is 5~7 days. Any effort to reduce the run time and timing closure cycle will be of great help. From previous section we can conclude that to remove timing violation either a long feedback is used or the useful skew is retrieved at the post synthesis stage. These long feedback increase the run time for chip implementation and taking feedback from the implementation tool result in uncorrelated data at the post-synthesis and post-route result as the feedback taken when the data path optimization is design has already taken place.

In synthesis, data path optimization is done to remove timing violation. The data path optimization lead to addition of buffers, invertors and re-sizing of cell. The excessive data path optimization at synthesis undermine the positive slack available at subsequent stage. This available positive slack can be utilize to reduce run time and effort level invest at synthesis stage.

To better utilize these positive slack from synthesis stage onwards we have proposed a method which retrieve this positive slack in synthesis stage and distribute them to the most critical path, accordingly develop the netlist without much effort invest in data path optimization. These useful skew value are again feed at placement stage so that skew aware placement take place. This method not only result in improving the setup timing violation but also result in better area and power consumption. Overall timing QoR of our design is improved and run time of our design is also improved comparing to the conventional flow.

Chapter 3

Proposed Method

In proposed method, useful_skew_proc is created using a TCL (tool command language) scripting language. The proc is used in the synthesis stage and the dumped skew adjustment file which is the output of our proc is feed to the Placement and Clock tree synthesis stage in implementation tool. There are various switches in the proc which can control the performance of the script and can impact the overall result of the design. Different switches are discussed in section 3.2.

Proposed proc make the synthesis tool aware of the clock skew optimization happen in the next stage. Zero skewing in synthesis stage is to conservative and doesn't take the account of clock optimization in subsequent stage. By using the proposed method we are synthesizing our netlist based on non-zero skewing approach such that the synthesis tool is more aware of the clock tree and can invest more effort in the critical path of our design which can effect our result in the implementation run.

The output skew adjustment script is feed forward to the implementation tool in placement stage. Sourcing these skew value will make the placement stage more skew aware and the placement of the cells is done accordingly. The run time for placement will also reduce as their will be less path for optimization and earlier critical path is converted into positive slack path by using useful skew. By doing so less number of buffer and invertor will be added and resizing of cell is also not required which will also result in saving area and power.

These value are also feed to the Clock tree build stage, such that we are forcing the CTS engine to implement these value in the clock tree. Further CTS can optimize the tree and data path for hold violation which will result in overall improvement of our result QoR.

3.1 Flow Chart

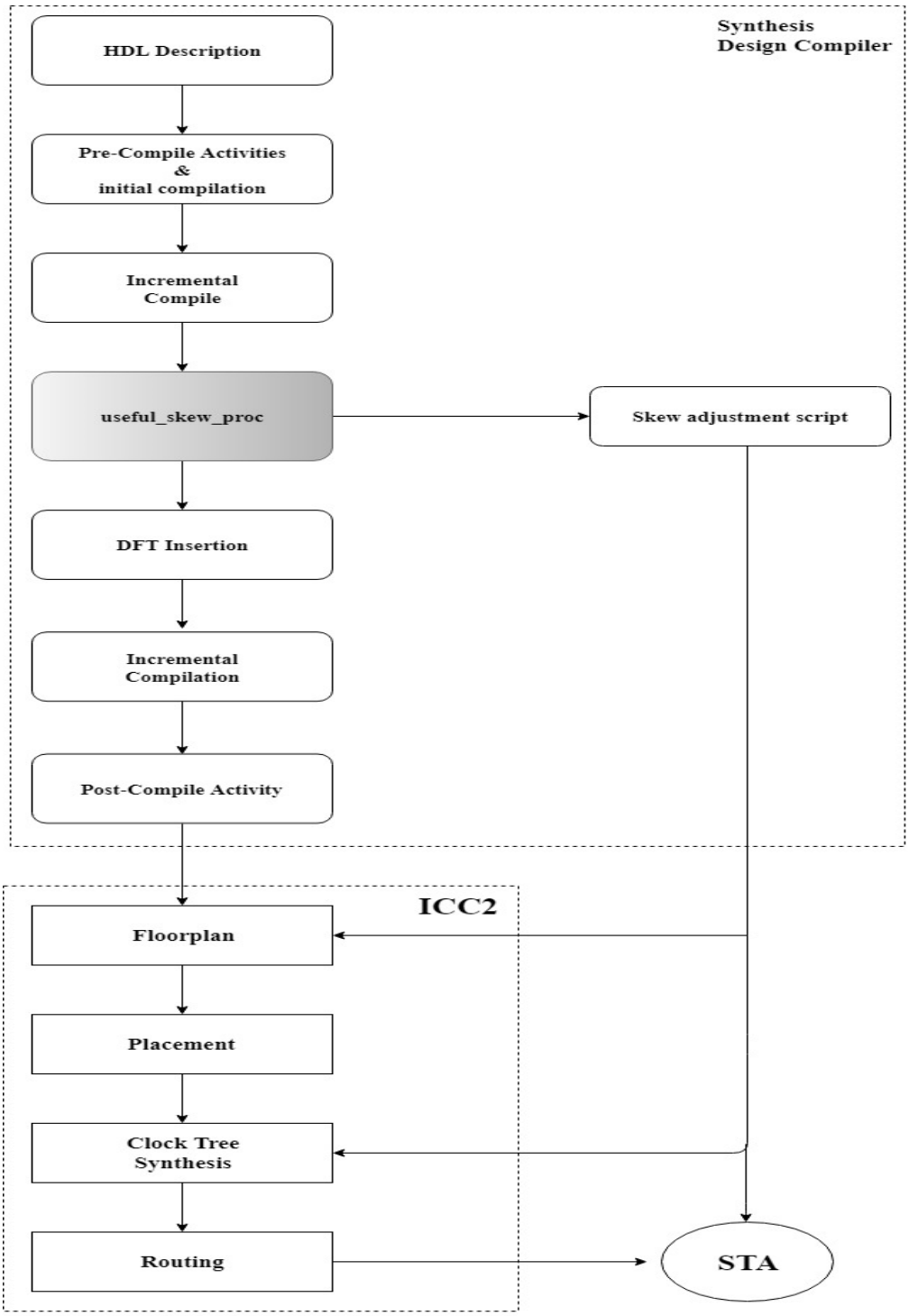


Figure 3.1 Proposed Flow

In the proposed flow chart, we can see that the useful skew value are retrieved at the synthesis stage after initial and incremental compilation. In the initial compilation our design is mapped to the cell and no optimization has taken place so far. Experimental result shows that the implementation of useful_skew_proc after initial compile lead to

degraded result as after the incremental compilation the drive strength of cell are changed and buffer-invertor are added or removed from the design. This lead to a uncorrelated result after proc implementation and post synthesis. To overcome this issue we started to implement the useful_skew_proc after the incremental compilation.

Proposed proc can also be implemented after the DFT insertion but experimental result show us that result are improved but run time is increased drastically. The output skew adjustment script contain information about all the clock pin of the flop on which skew is applied. This skew adjustment script is feed to the placement stage and cts stage for better timing correlation and make the implementation and synthesis tool more correlated.

Skew adjustment script is also feed to the STA engine for verification at placement and post synthesis result. We are using Design Compiler from Synopsys for the synthesis, IC Compiler II from Synopsys for Placement and Routing Stage and Prime Time from Synopsys for the Timing analysis of our design. StarRC extraction is used to generated SPEF file for more accurate result.

3.2 Script Argument Inputs

```
define_proc_attributes useful_skew_proc \
-info "PREDICTIVE CLOCK SKEW REDISTRIBUTION METHODOLOGY FOR IMPROVING TIMING QoR." \
-define_args {
{-depth "depth is used for 2nd order search. If required positive slack is not available at 1st order, 2nd order
        binary search will take place." AString string optional}

{-slack_margin "Minimum  +ve slack that should be available at Flop's" AString string optional}

{-max_borrowing "Maximum skew adjustment that can be done" AString string optional}

{-output_dir "Define where output skew adjustment script should be dumped" AString string optional}

{-max_path "Maximum number of path on which proposed proc should be applied" AString string optional}

{-scenario "Define the current/prefered scenario on which proc should run" AString string optional}

{-min_vio_slack "Minimum value of timing violation on which proc should be applied" AString string optional}

}
```

Figure 3.2 Script input arguments.

For maximum flexibility, user can control many switches of the script, which can impact the runtime and QoR. One of the switches is "-depth" which control how far the 2nd order search will look for the available positive slack values. A high depth value can improve the timing result but also result in an increased runtime. Another switch "-max_path" which control the maximum number of violating path on which our script will run. High value of "-max_path" can directly impact our runtime. All switches have their default values. Hence it is not compulsory for the user to provide all of them but the best possible QoR within the reasonable runtime can be achieved by providing the complete argument.

3.3 Proposed Method Flow.

I. Create a R2R path group for the entire worst case scenario. These path group should exclude path from don't touch.

II. Using "report_qor" command gets the total Number of Violating Endpoints (NVE) for each scenario. Otherwise user can also provide the NVE number using –max_path argument to save run time.

III. From the NVE number in (II) and by using "report_timing" command, gather all information on Startpoint, Endpoint, Startpoint clock, Endpoint Clock, violated slack and store them into a array.

IV. Now you can sort them from max violating path to the min. By experiment and iteration it have been observed that if script work from worst violating path toward minimum violating, we get better QoR and improved run time.

V. Next, we filter the data path based on the clock domain. Startpoint-Endpoint pair should be of same clock domain. Clock crossing is difficult to handle without the full design knowledge.

VI. Additional filtering required for the startpoint endpoint pair

   a. Endpoint should be unique in the collection.
   b. Startpoint Endpoint should be constrained.
   c. Remove any self-loop.
   d. For ease off calculation loop can be removed from design.
   e. Start/End points should not be a in/out ports.

VII. With the left over endpoint pair after filtering we will apply the clock skew adjustment for removing timing violation.

VIII. To obtain the value of slack for a given endpoint-startpoint pair we use a combination of "get_attribute" and "get_timing_path" command.

IX. The next step is to update latency of startpoint-endpoint if they are already applied in earlier stage.

X. With the updated slack, we can provide a clock skew adjustment if positive slack is available in the adjacent stage. If positive slack is available then the clock latency is written out to a file with "set_clock_latency".

Example :- "set_clock_latency 0.35 a/b/c/CK"

Where 0.35 is the delay that we want to introduced at the a/b/c/CK clock pin of the flop.

XI. After retrieving all the skew value and updating them in an output file, Next step will be to find the common skew point so that skew number can be grouped together for a hierarchy.

As shown in figure 3.2, the user input argument such as depth, slack margin & max_borrowing etc. can further improve the QoR but can result in increased run time for the script. By iteration and adjusting the depth and slack margin value, design QoR and run time can be balanced for optimum performance.

3.4 1st order search

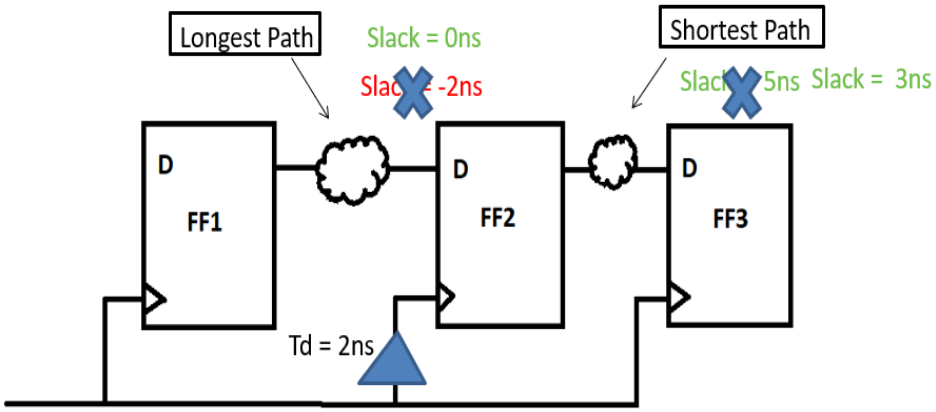
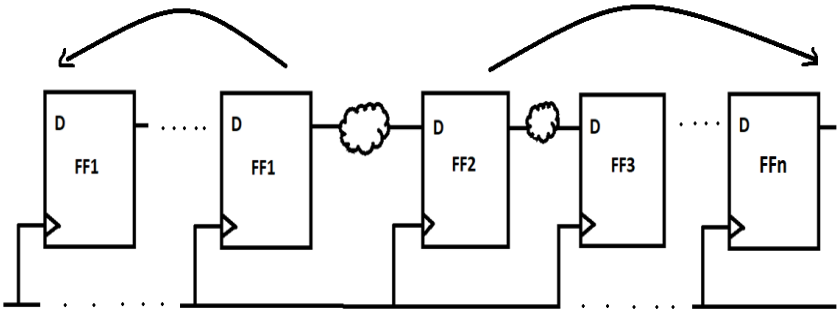


Figure 3.4 1st order Search

In 1st order, available skew is retrieved from the adjacent register. Positive slack is check in the adjacent register and distributed to the critical path. In Figure 3.4, consider flop pair FF1 and FF2 having a timing violation of 2ns. The adjacent flop FF3 to FF2 is meeting time by 5ns. This positive slack of 5ns can be distributed back to FF1-FF2 to removing timing violation. By giving a skew value of 2ns to FF2 the timing violation can be removed without effecting the adjacent register timing.

3.5 2nd order search



In 2nd order search, positive slack is not only retrieved from the adjacent register but also from the next register down the line. By adjusting the "-depth" switch we can control how far the 2nd order will search the positive skew down the line.

3.6 Proposed flow sequence

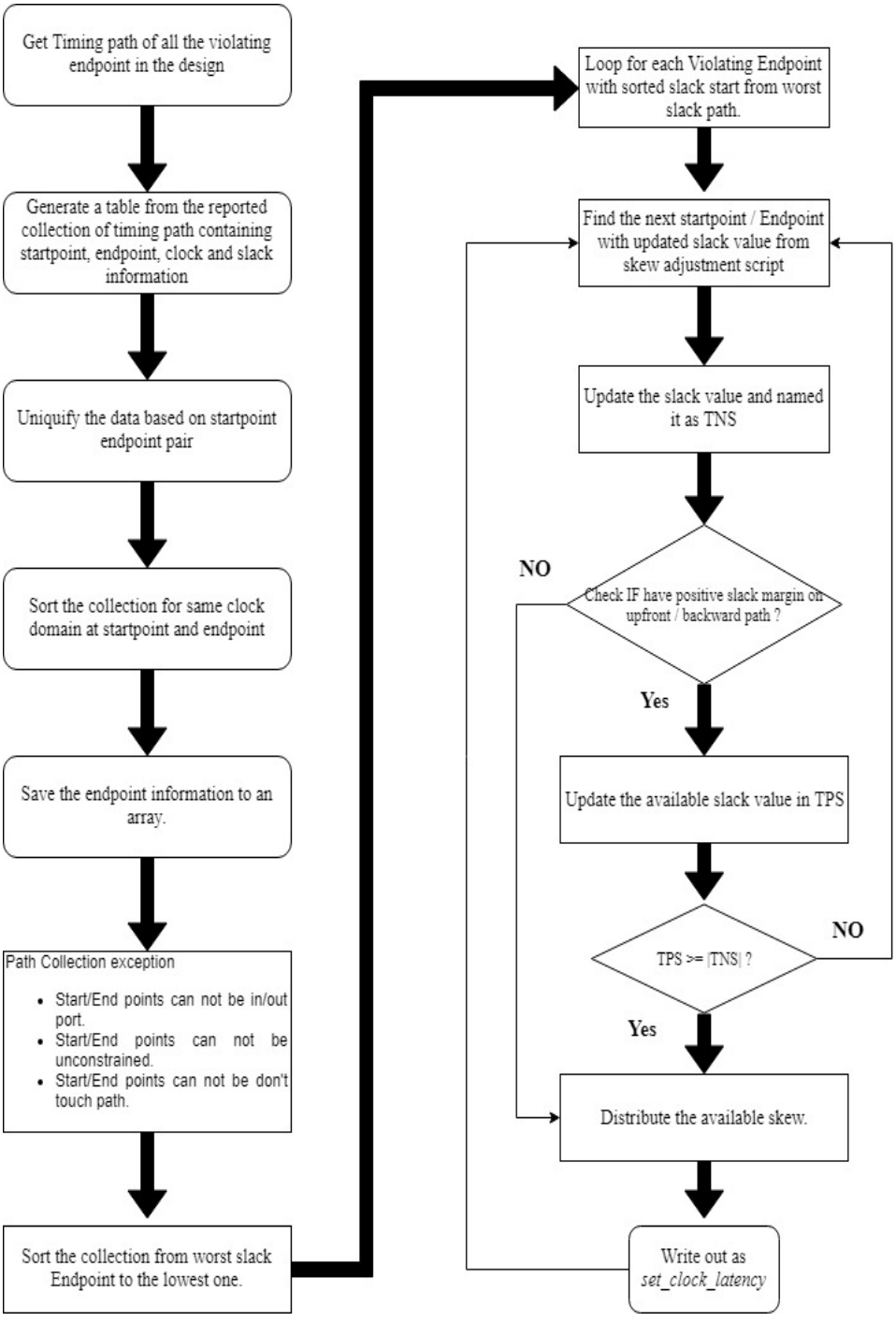


Figure 3.3 Algorithm Flow

3.7 Hierarchical Common Skew Points

In a design cells belonging to the common hierarchy used to sit together during placement stage. By taking advantage of this placement strategy, we can group the common skew point in an hierarchy based on the skew values. This grouping allows us to use lesser clock buffers/invertors cell required during balancing of the clock tree build stage. This not only allows us to save area but power of our design is saved as the cell required to build clock tree are generally LVT cell. In figure xxx it can be seen that a given hierarchy is sitting together after the placement stage.

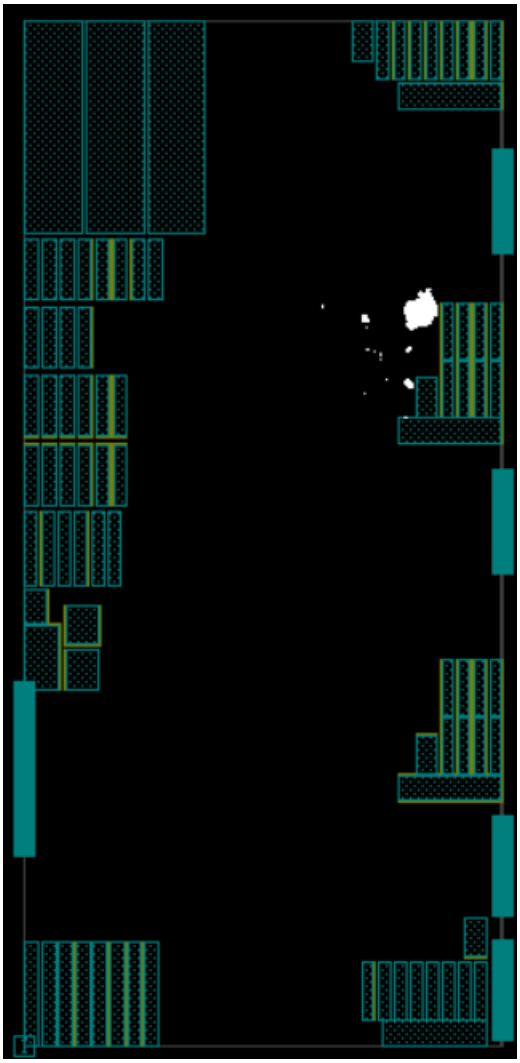


Figure 3.4 Hierarchy Placement

Chapter 4

RESULT

The predictive clock skew redistribution scheme was tested on a design with target frequency of 725 MHz. Design also consist of multiple clock with frequency ranges from 450 MHz to 725 MHz. The SoC design tested was on 28nm technology node.

At synthesis, timing comparison is done for setup violation at the slow corner as shown in Table 4.1. The timing analysis is done with Prime Time tool for accurate result. All the result are analyzed for the Reg2Reg path.

SETUP (0_Cworst)			
DESIGN	**WNS(ns)**	**TNS(ns)**	**NVE**
Classic	0.164	50.362	2287
Proposed Method	0.042	0.152	9

Table 4.1 Timing Report at post-synthesis.

Table 4.1 shows the setup violation summary at the post synthesis stage. We have used to 0_Cworst corner to analyzed the setup violation as it is the worst slow corner in our design. From the table we can say that the same design show better result with the skew adjustment script with respect to the Conventional flow. Also the number of violating endpoint has been reduced drastically. The proposed method not only reduce the Total negative slack but worst negative slack of our design is also reduced.

Scenario's	SETUP					
	CLASSIC			**Proposed Method**		
	WNS(ns)	**TNS(ns)**	**NVE**	**WNS(ns)**	**TNS(ns)**	**NVE**
0_RCworstT_SSG	0.381	4447.724	37523	0.429	4005.497	35888
125_RcworstT_SSG	0.261	448.052	4231	0.306	399.347	2683
0_CworstT_SSG	0.419	6267.667	44178	0.442	5878.204	42250
125_CworstT_SSG	0.278	605.168	8664	0.311	465.748	5318

Table 4.2 Setup timing summary after implementation run.

Result in Table 4.2 shows the setup timing summary across all different corner i.e. slow corner's, after design went through the complete implementation run. The result are obtained using Prime time tool and RC-STAR extraction is done for generating SPEF file which provide us with the more accurate result.

From Table 4.2, it can be observed that setup violation have improved by 6.21% at the worst corner. There is a small increase in the worst negative slack in our design but the number of violation above 350ps were few. Overall the setup violation is improved and number of violation is also reduced. This can be inferred from figure 4.1, which shows the comparison between the classing TNS and Proposed method TNS. By using Bar graph it can be clearly seen that the setup violation have been improved.

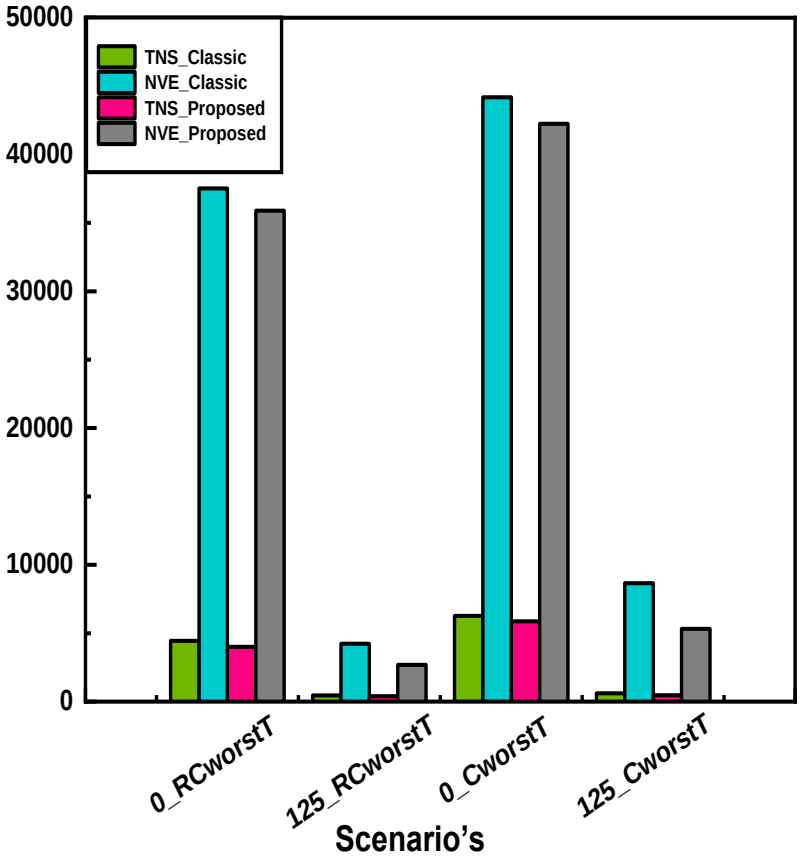


Figure 4.1 Setup timing Summary

Table 4.3, which shows the hold timing summary across different corner i.e. fast corner. The result shows that there is a degradation of number of hold violation. 125_RCbest is the worst corner for hold analysis. The degradation is observed in hold analysis because after the proc implementation and an iteration of incremental compile, the synthesis tool optimize only setup timing violation without taking the hold violation into account. This optimization lead to the removing of buffer or sizing of cell which ultimately lead to hold degradation. To remove the hold violation we have taken the fast corner into consideration to address this issue.

SCENARIO'S	HOLD					
	CLASSIC			Proposed Method		
	WNS	**TNS**	**NVE**	**WNS**	**TNS**	**NVE**
SSG						
0_Cworst	0.117	15.254	867	0.174	23.379	1182
125_Cworst	0.112	17.537	1003	0.17	28.239	1448
0_RCworst	0.116	13.186	801	0.176	22.566	1163
125_RCworst	0.108	14.517	901	0.171	28.153	1449
FFG						
0_Cbest	0.094	73.55	5699	0.143	109.992	5977
125_Cbest	0.107	120.733	8103	0.161	166.562	8182
0_RCbest	0.092	82.174	6048	0.139	114.369	6339
125_RCbest	0.1	129.223	8444	0.155	164.762	8571
0_Cworst	0.099	103.521	7073	0.151	141.639	7302
125_Cworst	0.108	158.003	9659	0.17	204.411	9812
0_RCworst	0.101	93.033	6678	0.154	135.944	6902
125_RCworst	0.116	151.624	9718	0.174	203.123	9445

Table 4.3 Hold Timing summary after implementation run.

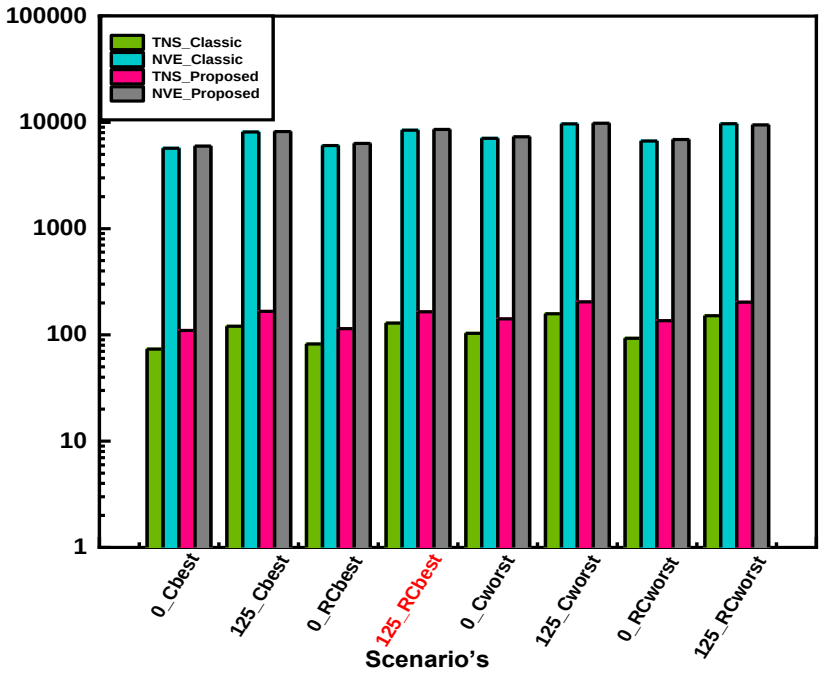


Figure 4.2 Hold Violation Summary

In Figure 4.2, we can see that the number of hold violation in all the scenario are comparatively same and there is a small degradation in the total negative slack. The difference in hold violation can be better understood by considering the histogram of hold violation of proposed method in below graph.

In histogram, obtained from table 4.4, as shown in figure 4.3, shows that there are few violation above 100 ps and maximum number of violation is concentrated around 65 ps to 30 ps. These violation can easily be removed in timing closure cycle without effecting much of the effort level and run time. Overall we can say that there is an improvement of 5.53% in timing violation.

Bin	CLASSIC	Proposed Method
-0.135	0	4
-0.105	0	15
-0.09	5	19
-0.075	17	40
-0.065	35	120
-0.045	242	700
-0.03	742	1089
-0.015	2694	2164
0	4709	4420
Total Vio	8444	8571

Table 4.4 Histogram for Hold Violation

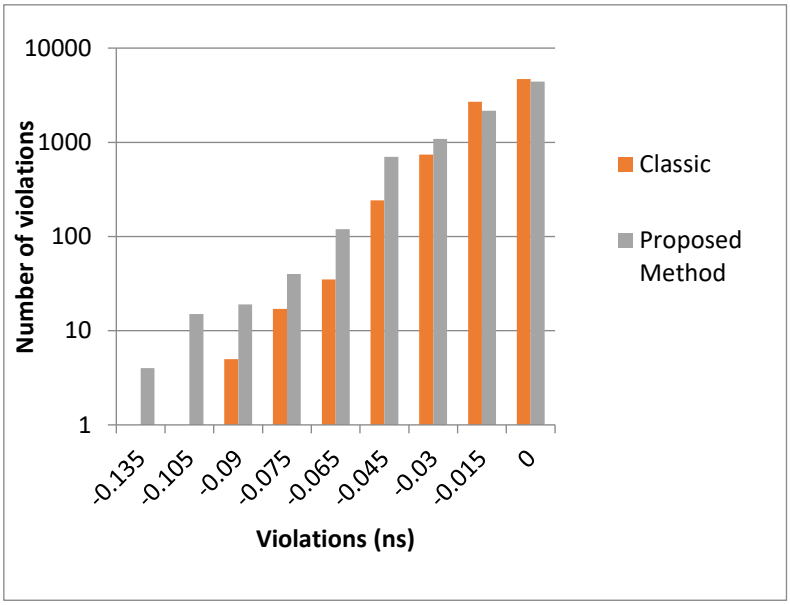


Figure 4.3 Graph comparing Hold violation.

Next we look at table 4.5, the area comparison with and without the predictive skew algorithm. From the table 4.5, it can be observer that there is an improvement in total cell count which lead to a lesser area in skew adjustment script design.

	CLASSIC		Proposed Method	
VT	**No. of Cells**	**Area(um2)**	**No. of Cells**	**Area(um2)**
30uhd_hvt	32466	30103.15	32666	30412.93
35uhd_hvt	121460	114333.17	130607	128885.39
40uhd_hvt	1212023	865393.31	1234762	882128.67
30uhd_svt	319399	436814.42	285473	373675.08
35uhd_svt	314847	529690.39	332819	541693.73
40uhd_svt	762230	634294.51	725596	609105.97
30uhd_lvt	0	0	0	
35uhd_lvt	21187	28858.75	22849	30776.21
40uhd_lvt	0	0	0	0
undefined	85	1116482.18	85	1116482.18
TOTAL	2783697	3755969.88	2764857	3713160.15

Table 4.5 Comparison of Area Summary and Cell Count.

The Table 4.5 also show that with more number of HVT, SVT cells there will be reduction in power consumption. This can be verified from Table 4.6.

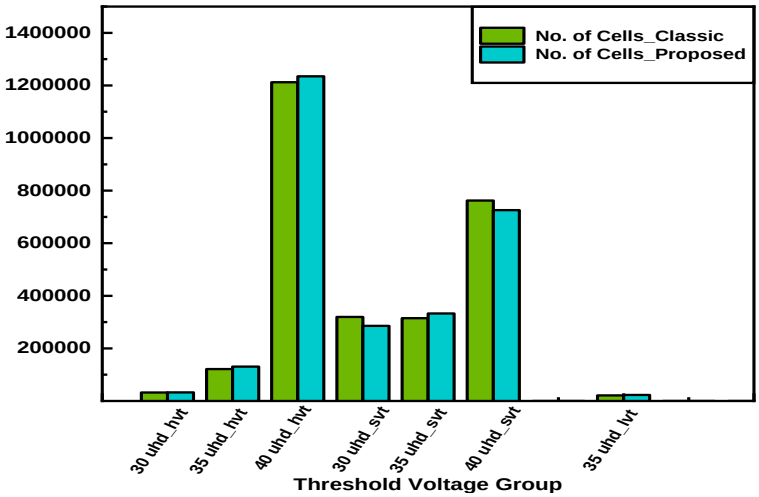


Figure 4.4 No of cell Comparison

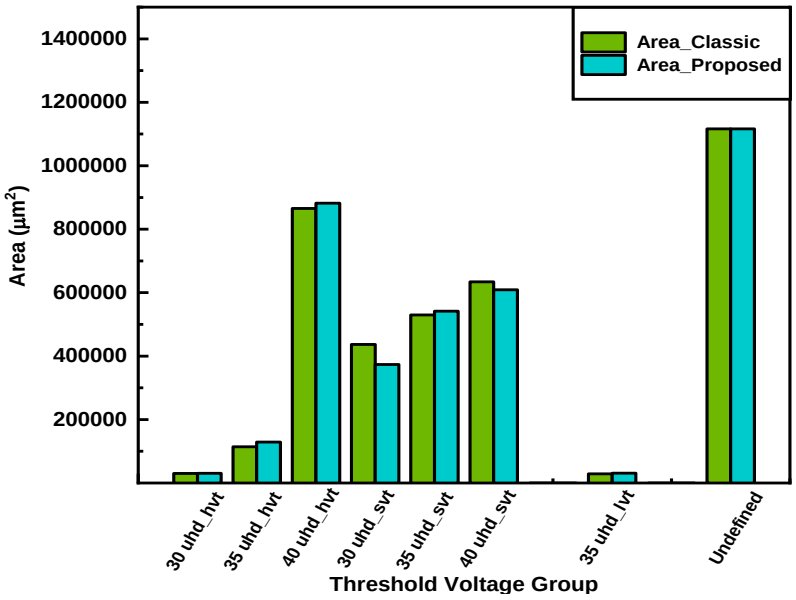


Figure 4.5 Area Comparison

VT	Classic	PROC
Power (nW)	4.06E+08	3.81E+08

Table 4.6 Power Comparison

VT	With_skew_only	With_skew_with_hier
35uhd_lvt	26574	22849

Table 4.7 CTS cell count comparison for common skew points method.

With lesser count of LVT cell in Table 4.7, we can say that the hierarchical common skew point method come out to be as success as the LVT cell are only used during the clock tree synthesis stage. A lesser value of LVT directly relate to lesser cell count at clock build stage.

Finally, run time of both the design i.e. with skew adjustment and without skew adjustment is presented in table 4.8. This further proved that the skew adjustment effectively help with the QoR of the design and result in lesser run time comparing to the conventional flow.

STAGE		Classic	Proposed Flow
Synthesis		43:49:00	55:36:00
Placement		32:00:00	16:00:00
CTS		35:25:00	30:30:00
Routing		53:20:00	55:00:00
Total (hrs) aprroax		164:34:00	157:06:00

Table 4.8 Runtime Comparison

Chapter 5

5.1 Conclusion

Predictive clock skew redistribution unidirectional flow methodology was successful implemented. From previous section, we have concluded that the utilization of positive slack from the early stage and redistributing it to most critical path not only improve the timing QoR but also result in a better run time. The search for positive slack not only limited to adjacent register but also can be utilize for n register down the line with the help of 2^{nd} order search. This might result in more number of violating points but result in decreasing the total negative slack significantly.

The proposed common hierarchy clock skew point also help in reducing the number of buffer required in clock tree synthesis stage which result in overall reduction in clock tree power consumption.

The overall runtime, power and area is also improved compare to the conventional flow which result in better utilization at timing closure. The degradation of hold violation is observed but these minute violation can easily be removed in timing closure without significant area and power penalty.

The flow is generic which mean it is technology independent and can be used at different technology node.

5.2 Future Scope

The proposed method can be used with MCMM (Multi corner multi-mode) analysis from the early stage to improve hold violation. Another upgradation will to use spg flow i.e. physical aware skew distribution at synthesis stage. SPG flow not only help in improving retrieving skew value but result will also be better correlated with the implementation engine. To remove hold violation, a high clock uncertainty value can also be used at synthesis stage so that the violation will be correlated to the implementation stage.

Chapter 6

REFERENCES

[1] K. Wang, L. Duan, and X. Cheng, “Extensiveslackbalance: an approach to make front-end tools aware of clock skew scheduling,” in Proceedings of the 43rd annual Design Automation Conference. ACM, 2006, pp. 951–954.

[2] J. P. Fishburn, “Clock skew optimization,” IEEE transactions on com- puters, vol. 39, no. 7, pp. 945–951, 1990.

[3] K. Ravindran, A. Kuehlmann, and E. Sentovich, “Multi-domain clock skew scheduling,” in Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design. IEEE Computer Society, 2003, p. 801.

[4] C. Albrecht, B. Korte, J. Schietke, and J. Vygen, “Cycle time and slack optimization for vlsi-chips,” in Proceedings of the 1999 IEEE/ACM inter- national conference on Computer-aided design. IEEE Press, 1999, pp. 232–238.

[5] T.-B. Chan, A. B. Kahng, and J. Li, “Nolo: A no-loop, predictive useful skew methodology for improved timing in ic implementation,” in Fifteenth International Symposium on Quality Electronic Design. IEEE, 2014, pp. 504–509.

[6] C. Albrecht, A. B. Kahng, B. Liu, I. I. Mandoiu, and A. Z. Zelikovsky, “On the skew-bounded minimum-buffer routing tree problem,” IEEE Trans- actions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 7, pp. 937–945, 2003.

[7] C. Albrecht, P. Witte, and A. Kuehlmann, “Performance and area opti- mization using sequential flexibility,” in Proc. International Workshop on Logic and Synthesis. Citeseer, 2004.

[8] S. Kim, S. Do, and S. Kang, “Fast predictive useful skew methodology for timing-driven placement optimization,” in Proceedings of the 54th Annual Design Automation Conference 2017. ACM, 2017, p. 55.

[9] C. Albrecht, “Efficient incremental clock latency scheduling for large cir-

cuits," in Proceedings of the Design Automation & Test in Europe Con- ference, vol. 1. IEEE, 2006, pp. 1–6.

[10] S. Roy, P. M. Mattheakis, L. Masse-Navette, and D. Z. Pan, "Clock tree resynthesis for multi-corner multi-mode timing closure," IEEE Trans- actions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 4, pp. 589–602, 2015.

[11] S.-H. Huang, Y.-H. Lin, and M.-L. Huang, "Utilizing clock skew for tim- ing reliability improvement," in TENCON 2007-2007 IEEE Region 10 Conference. IEEE, 2007, pp. 1–4.

[12] R. Chaturvedi and J. Hu, "A simple yet effective merging scheme for prescribed-skew clock routing," in Proceedings 21st International Confer- ence on Computer Design. IEEE, 2003, pp. 282–287.

[13] C. Albrecht, B. Korte, J. Schietke, and J. Vygen, "Maximum mean weight cycle in a digraph and minimizing cycle time of a logic chip," Discrete Applied Mathematics, vol. 123, no. 1-3, pp. 103–127, 2002.

[14] IC Compiler 2 User Guide.

[15] Design Compiler User guide.

[16] Prime Time User guide.

[17] Solvnet – Synopsys User Group.

[18] S.-H. Huang and Y.-T. Nieh, "Clock period minimization of non-zero clock skew circuits," in Proceedings of the 2003 IEEE/ACM international con- ference on Computer-aided design. IEEE Computer Society, 2003, p. 809.

[19] A. P. Hurst, P. Chong, and A. Kuehlmann, "Physical placement driven by sequential timing analysis," in Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design. IEEE Computer Society, 2004, pp. 379–386.

[20] M. C. Papaefthymiou, "Understanding retiming through maximum average-delay cycles," Mathematical Systems Theory, vol. 27, no. 1, pp. 65–84, 1994.

[21] D. Velenis, K. T. Tang, I. S. Kourtev, V. Adler, F. Baez, and E. G. Friedman, "Demonstration of speed enhancements on an industrial circuit through application of non-zero clock skew scheduling," in ICECS 2001. 8th IEEE International Conference on Electronics, Circuits and Systems (Cat. No. 01EX483), vol. 2. IEEE, 2001, pp. 1021–1025.

[22] J. L. Neves and E. G. Friedman, “Optimal clock skew scheduling tolerant to process variations,” in 33rd Design Automation Conference Proceed- ings, 1996. IEEE, 1996, pp. 623–628.

[23] “Design methodology for synthesizing clock distribution networks exploiting nonzero localized clock skew,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 4, no. 2, pp. 286–291, 1996.

[24] G. De Micheli, “Synchronous logic synthesis: Algorithms for cycle-time minimization,” IEEE Transactions on Computer-Aided Design of Inte- grated Circuits and Systems, vol. 10, no. 1, pp. 63–73, 1991.

[25] K. A. Sakallah, T. N. Mudge, and O. Olukotun, “checkt, and mintc: Timing verification and optimal clocking of synchronous digital circuits,” Ann Arbor, vol. 1001, pp. 48 109–2122, 1990.

[26] T. G. Szymanski, “Computing optimal clock schedules,” in [1992] Proceedings 29th ACM/IEEE Design Automation Conference. IEEE, 1992, pp. 399–404.

[27] S. Pullela, N. Menezes, J. Omar, and L. T. Pillage, “Skew and delay optimization for reliable buffered clock trees,” in Proceedings of 1993 International Conference on Computer Aided Design (ICCAD). IEEE, 1993, pp. 556–562.

[28] B. Wu and N. A. Sherwani, “Effective buffer insertion of clock tree for high-speed vlsi circuits,” Microelectronics journal, vol. 23, no. 4, pp. 291– 300, 1992.

[29] R.-S. Tsay, “An exact zero-skew clock routing algorithm,” IEEE Trans- actions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 2, pp. 242–249, 1993.

[30] K. D. Boese and A. B. Kahng, “Zero-skew clock routing trees with mini- mum wirelength,” in [1992] Proceedings. Fifth Annual IEEE International ASIC Conference and Exhibit. IEEE, 1992, pp. 17–21.

[31] J.-L. Tsai, T.-H. Chen, and C.-P. Chen, “Zero skew clock-tree optimiza- tion with buffer insertion/sizing and wire sizing,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, no. 4, pp. 565–572, 2004.

[32] R. B. Deokar and S. S. Sapatnekar, “A graph-theoretic approach to clock skew optimization,” in Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS’94, vol. 1. IEEE, 1994, pp. 407–410.

[33] L.-F. Chao and H.-M. Sha, “Retiming and clock skew for synchronous systems,” in Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS’94, vol. 1. IEEE, 1994, pp. 283–286.

[34] I. S. Kourtev and E. G. Friedman, “Clock skew scheduling for im- proved reliability via quadratic programming,” in Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design. IEEE Press, 1999, pp. 239–243.

[35] X. Liu, M. C. Papaefthymiou, and E. G. Friedman, “Maximizing per- formance by retiming and clock skew scheduling,” in Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361). IEEE, 1999, pp. 231–236.

[36] V. Nawale and T. W. Chen, “Optimal useful clock skew scheduling in the presence of variations using robust ilp formulations,” in Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design. ACM, 2006, pp. 27–32.

[37] Y. Taur, D. A. Buchanan, W. Chen, D. J. Frank, K. E. Ismail, S.-H. Lo,G. A. Sai-Halasz, R. G. Viswanathan, H.-J. Wann, S. J. Wind et al., “Cmos scaling into the nanometer regime,” Proceedings of the IEEE, vol. 85, no. 4, pp. 486–504, 1997.

[38] V. Mehrotra and D. Boning, “Technology scaling impact of variation on clock skew and interconnect delay,” in Proceedings of the IEEE 2001 Inter- national Interconnect Technology Conference (Cat. No. 01EX461). IEEE, 2001, pp. 122–124.

[39] A. Rajaram and D. Z. Pan, “Robust chip-level clock tree synthesis,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Sys- tems, vol. 30, no. 6, pp. 877–890, 2011.

[40] D.-J. Lee and I. L. Markov, “Obstacle-aware clock-tree shaping during placement,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 2, pp. 205–216, 2012.

[41] Y. Wang, Q. Zhou, X. Hong, and Y. Cai, “Clock-tree aware placement based on dynamic clock-tree building,” in 2007 IEEE International Sym- posium on Circuits and Systems. IEEE, 2007, pp. 2040–2043.

[42] K. Rajagopal, T. Shaked, Y. Parasuram, T. Cao, A. Chowdhary, and B. Halpin, “Timing driven force directed placement with physical net con- straints,” in

Proceedings of the 2003 international symposium on Physical design. ACM, 2003, pp. 60–66.

[43] Y. Liu, R. S. Shelar, and J. Hu, “Delay-optimal simultaneous technology mapping and placement with applications to timing optimization,” in Proceedings of the 2008 IEEE/ACM International Conference on Computer- Aided Design. IEEE Press, 2008, pp. 101–106.

[44] S.-W. Hur, A. Jagannathan, and J. Lillis, “Timing-driven maze routing,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 2, pp. 234–241, 2000.

[45] K. Sato, M. Kawarabayashi, H. Emura, and N. Maeda, “Post-layout optimization for deep submicron design,” in 33rd Design Automation Con- ference Proceedings, 1996. IEEE, 1996, pp. 740–745.

[46] Y.-P. Chen, J.-W. Fang, and Y.-W. Chang, “Eco timing optimization using spare cells,” in Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design. IEEE Press, 2007, pp. 530–535.

[47] M. Ni and S. O. Memik, “A revisit to the primal-dual based clock skew scheduling algorithm,” in 2010 11th International Symposium on Quality Electronic Design (ISQED). IEEE, 2010, pp. 755–764.

[48] S. M. Burns, “Performance analysis and optimization of asynchronous circuits,” 1991.

[49] J. Lu and B. Taskin, “Post-cts clock skew scheduling with limited de- lay buffering,” in 2009 52nd IEEE International Midwest Symposium on Circuits and Systems. IEEE, 2009, pp. 224–227.

[50] W. Shen, Y. Cai, W. Chen, Y. Lu, Q. Zhou, and J. Hu, “Useful clock skew optimization under a multi-corner multi-mode design framework,” in 2010 11th International Symposium on Quality Electronic Design (ISQED). IEEE, 2010, pp. 62–68.

[51] V. Ramachandran, “Functional skew aware clock tree synthesis,” in Proc. Int. Symp. Phys. Design, 2012.

[52] J. Bhasker and R. Chadha, Static timing analysis for nanometer designs: A practical approach. Springer Science & Business Media, 2009.

[53] V. G. Oklobdzija, V. M. Stojanovic, D. M. Markovic, and N. M. Nedovic, Digital system clocking: high-performance and low-power aspects. John Wiley & Sons, 2005.

[54] M. Ni and S. O. Memik, “A fast heuristic algorithm for multidomain clock skew scheduling,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 4, pp. 630–637, 2009.

[55] J. P. Fishburn, “Solving a system of difference constraints with variables restricted to a finite set,” Information processing letters, vol. 82, no. 3, pp. 143–144, 2002.

[56] S.-H. Huang, C.-M. Chang, and Y.-T. Nieh, “Fast multi-domain clock skew scheduling for peak current reduction,” in Asia and South Pacific Conference on Design Automation, 2006. IEEE, 2006, pp. 6–pp.

[57] H. Jiang, K. Wang, and M. Marek-Sadowska, “Clock skew bounds esti- mation under power supply and process variations,” in Proceedings of the 15th ACM Great Lakes symposium on VLSI. ACM, 2005, pp. 332–336.

[58] C. Lin and H. Zhou, “Clock skew scheduling with delay padding for pre- scribed skew domains,” in 2007 Asia and South Pacific Design Automa- tion Conference. IEEE, 2007, pp. 541–546.

[59] I.-M. Liu, T.-L. Chou, D. Wong, and A. Aziz, “Zero-skew clock tree con- struction by simultaneous routing, wire sizing and buffer insertion,” in Proceedings of the 2000 international conference on Computer-aided de- sign. IEEE Computer Society Press, 2001, pp. 33–38.

[60] S. Tam, R. D. Limaye, and U. Desai, “Clock generation and distribution for the 130-nm itanium/sup/spl reg//2 processor with 6-mb on-die l3 cache,” IEEE Journal of Solid-State Circuits, vol. 39, no. 4, pp. 636–642, 2004.

[61] S. Tam, S. Rusu, U. N. Desai, R. Kim, J. Zhang, and I. Young, “Clock gen- eration and distribution for the first ia-64 microprocessor,” IEEE Journal of Solid-State Circuits, vol. 35, no. 11, pp. 1545–1552, 2000.

[62] J.-L. Tsai, T.-H. Chen, and C. C.-P. Chen, “Optimal minimum-delay/area zero-skew clock tree wire-sizing in pseudo-polynomial time,” in Proceedings of the 2003 international symposium on Physical design. ACM, 2003, pp. 166–173.

[63] J.-L. Tsai, L. Zhang, and C. C.-P. Chen, “Statistical timing analysis driven post-silicon-tunable clock-tree synthesis,” in ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005. IEEE, 2005, pp. 575–581.

[64] A. Vittal, H. Ha, F. Brewer, and M. Marek-Sadowska, “Clock skew optimization for ground bounce control,” in Proceedings of the 1996 IEEE/ACM

international conference on Computer-aided design. IEEE Computer Society, 1997, pp. 395–399.

[65] Z. Xing and P. Banerjee, “A parallel algorithm for zero skew clock tree routing,” in Proceedings of the 1998 international symposium on Physical design. ACM, 1998, pp. 118–123.

[66] B. Taskin and J. Lu, “Post-cts delay insertion to fix timing violations,” in 2008 51st Midwest Symposium on Circuits and Systems. IEEE, 2008, pp. 81–84.

[67] W.-K. Chen, The VLSI handbook. CRC press, 2010.

[68] E. G. Friedman, “Clock distribution networks vlsi circuits and systems,” A Selected Reprint Volume, 1995.

[69] M. A. Jackson, A. Srinivasan, and E. S. Kuh, “Clock routing for high-performance ics,” in 27th ACM/IEEE Design Automation Conference. IEEE, 1990, pp. 573–579.

[70] N.-C. Chou and C.-K. Cheng, “On general zero-skew clock net construc- tion,” IEEE Transactions on Very Large Scale Integration (VLSI) Sys- tems, vol. 3, no. 1, pp. 141–146, 1995.

[71] N. Ito, H. Sugiyama, and T. Konno, “Chipprism- clock routing and tim- ing analysis for high-performance cmos vlsi chips,” Fujitsu Scientific & Technical Journal, vol. 31, no. 2, pp. 180–187, 1995.