A NOVEL APPROACH TO **POWER OPTIMIZATION** IN HIGH FREQUENCY MICROPROCESSOR DESIGN USING AN INTELLIGENT **SLACK MANIPULATION** ALGORITHM

Major Project Report

Submitted in partial fulfillment of the requirements

For the Degree of

Master of Technology

In

Electronics & Communication Engineering
(VLSI Design)

By

Kruti Trivedi
(10MECV08)

Department of Electronics & Communication Engineering
Institute of Technology

A novel approach to **Power Optimization** in High Frequency Microprocessor Design using an intelligent **slack manipulation** algorithm

Major Project Report

Submitted in partial fulfillment of the requirements

For the Degree of

Master of Technology

In

Electronics & Communication Engineering
(VLSI Design)

By

Kruti Trivedi
(10MECV08)

Under Guidance of

Dr. N.M Devashrayee

Anoop V

Department of Electronics & Communication Engineering
Institute of Technology

Certificate

This is to certify that the Major Project Report entitled "**A novel approach to Power Optimization in High Frequency Microprocessor design using an intelligent slack manipulation Algorithm** " submitted by Kruti Trivedi (10MECV08), towards the partial fulfillment of the requirements for the Degree of Master of Technology in Electronics & Communication Engineering (VLSI Design) of Nirma University, Ahmedabad is the record of work carried out by her under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree.

Date: Place: **Bangalore**

Anoop V,
Design Automation Team,
Intel Corporation,
Bangalore,

Dr. N. M. Devashrayee
PG-Coordinator(VLSI Design),
Institute of Technology,
Nirma University,
Ahmedabad.

Prof. A. S. Ranade
Head of Department,
Dept. of Electrical Engineering,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. K Kotecha
Director,
Institute of Technology,
Nirma University,
Ahmedabad.

Declaration

This is to certify that

i) The thesis comprises my original work towards the Degree of Master of Technology in Electronics & Communication Engineering (VLSI Design) at Nirma University, Ahmedabad and has not been submitted elsewhere for a degree.

ii) Due acknowledgement has been made in the text to all other material used.

-Kruti Trivedi

Acknowledgment

I am greatly indebted to all our faculties at **Institute of technology , Nirma University.** Especially **Dr. N.M. Devashrayee,** who has been a constant source of inspiration and support. It is indeed fortunate of me to have known a person with such positive energy which has always been a guiding light for me.

Innumerable Teachings by my faculties **Prof. N. P. Gajjar, Prof. Usha Mehta** and **Prof. Amisha P. Naik** have helped me in innumerable ways since the commencement of the Thesis work and have also helped me to easily get absorbed in the company for the internship.

I am extremely thankful to **Anoop V**, my mentor at Intel® Corporation who has been guiding me since day one and has been instrumental in every facet of my work at at Intel®. Also **Srivatsa Srinath** , my manager has been very helpful as he always brought out the best from within me.

I would also like to thank my all my team members in especially **Amit Dounde**, **Pradeep Bharadwaj** and **Milind Mahajan** who have helped me throughout the duration of my internship at intel.

Lastly I am thankful to all friends and family as without their love and support it would have been impossible for me to be where I am today.

Abstract

Versatile form factor and battery backup is one of the major focus in electronics industry. This leads to exploration more power reduction opportunities in all products . Tradeoffs in area , power and delay are the order of the day in VLSI field. However in some cases we can actually exploit some resources and manipulate them in such a way to optimize other resources rather than mere tradeoff. One such way is to ***squeeze extra margin*** from positive slack paths to gain power reduction.

This Thesis is an outcome of efforts directed to develop and test an algorithm which can utilize the extra slack margin available for the cell by intentionally engineering delays in the cells in the design and hence leading to a more power efficient design.

This Algorithm also physically implements addition of delays to cells by **downsizing** the cells and then **swapping** suitable cells in the design by the corresponding lower leakage cell instances from the available design library. This is done in manner such that the original timing on lower slack margin paths are not affected and the critical path slack remains essentially the same.

This Thesis also discusses various aspects related to Static Timing Analysis , Slack calculation and Slack Budgeting and Power Dissipation which were the prerequisites in development of this algorithm

This Thesis also discusses various aspects related to Static Timing Analysis , Slack calculation and Slack Budgeting and Power Dissipation which were the prerequisites in development of this algorithm.

Table Of Contents

Chapter 6: Results

Chapter 7: Conclusion and Future direction

List of Figures and Tables

Figures

Tables

List of Abbreviations

STA	**Static Timing Analysis**
DFT	**Design For Testability**
CAD	**Computer Aided Design**
IC	**Integrated Circuit**
CMOS	**Complementary Metal Oxide Semiconductor**
ASIC	**Application Specific Integrated Circuit**
RTL	**Register Transfer Level**
VHDL	**Very High Speed IC Hardware Description Language**
CTS	**Clock Tree Synthesis**
FP	**Floorplan**
SDC	**Synopsys™ Design Constraints**
TAT	**Turn Around Time**
RC	**Resistance and Capacitance**
ZSA	**Zero Slack Algorithm**
ECO	**Engineering Change Order**
BGA	**Ball Grid Array**
PCB	**Printed Circuit Board**
DFM	**Design For Manufacturability**
TCL	**Tool Command Language**
PERL	**Practical Report Extraction Language**
DUA	**Design Under Analysis**
FPGA	**Field Programmable Gate Array**
DAG	**Directed Acyclic Graph**
QoR	**Quality of Results**
GUI	**Graphical User Interface**
DRC	**Design Rule Check**

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

The main Motivation behind the conceptualization of the Thesis work was to study and implement various strategies to reduce the the electronic industry's most sought after resource – Power. It is the virtue which always enchants the one and all. No one will buy a high end tablet or an ultra smart phone if battery backup is not sufficient. There are innumerable strategies tried and tested every now and then for increasing Power savings.One such strategy around which the Thesis revolves is exploitation of extra timing available in the design.

Traditionally, power dissipation of VLSI chips was a neglected subject. In the past, the device density and operating frequency were low enough that it was not a constraining factor in the chips. As the scale of integration improves, more transistors, faster and smaller than their predecessors, are being packed into a chip. This leads to the steady growth of the operating frequency and processing capacity per chip, resulting in increased power dissipation. For state-of-the art systems, the trade-off solutions between the conflicting design criteria (i.e., delay, area, and power) should be considered.

In general, low-power optimizations without compromising performance are dependent on the **time slack calculation** and the **surplus slack** (slack budget) distribution. The time slack means the difference between the signal required time and the signal arrival time at the primary output of each module.

Due to design complexity, optimization techniques need to be applied in multiple stages starting from high level of abstraction down to gate level and physical design. In order to abstract the complexity, each design is decomposed into a set of sub-designs. The essential constraint during the design optimization flow is the timing constraint.

The sub-designs along the critical paths are the most constrained components during the optimization process in CAD flow. However, timing constraint is loose on the other sub-designs. Hence the allowable delay allocated on each sub-design can be greater than in the graph such that their physical size can be reduced by mapping to smaller cell instances with larger delay from a target library. In general, delay budgeting can be applied during library mapping stage.

1.2 RTL TO LAYOUT FLOW

As the thesis work and internship work at intel® required a clear understanding of the basic backend design flow, a brief introduction is provided with some basics and scope of optimization at various stages. Thus the rest of the chapter is an answer to the question:

What Is The BACKEND DESIGN FLOW?

In today's world of VLSI circuit design and manufacture, as system architects integrate more and more system functions into one chip, IC design engineers must confront the difficult challenge of building giant entities of hundreds of millions of transistors.

Two crucial requirements in this chip creation process are **build it correctly** and **build it fast**. Nowadays, it is commonly agreed that it will cost millions of dollars to develop a multimillion-gate ASIC, from concept to silicon. The financial penalty of building something incorrectly and remaking it all over again is intolerable.

Furthermore, in addition to being built correctly (performing as specified), it must also be built in a timely fashion. Otherwise, the market window for the product may be missed. There are too many examples of technically great products not being able to earn a dollar for their investors, simply due to the delay in their chip-creation execution. Fortunately, there is something called ***design flow***, which IC implementation engineers can rely on in this very demanding business.

A well-tuned design flow can help designers go through the chip-creation process relatively smoothly and with a decent chance of error-free implementation. And, a skillful IC implementation engineer can use the design flow creatively to shorten the design cycle, resulting in a higher likelihood that the product will catch the market window.

In principle, a design flow is a sequence of operations that transform the IC designers' intention (usually represented in RTL format) into layout GDSII data . In practice, a design flow is a sequence of executions that perform each individual task described previously. In composition, a design flow is a suite of software programs; they are either commercial CAD point tools or programs or scripts developed in-house.

The detailed tasks are:

- Logic synthesis
- DFT insertion
- Floorplan
- Design partition
- Macro placement
- Power distribution structure
- Clocks distribution structure
- Preliminary check
- Place and route
- Parasitic extraction and reduction
- Final layout generation Manufacturing

1.3 MAJOR STEPS

The major steps in chip construction are RTL coding, function verification, logic synthesis, place and route, final logic verification, timing verification, physical verification, and tapeout.

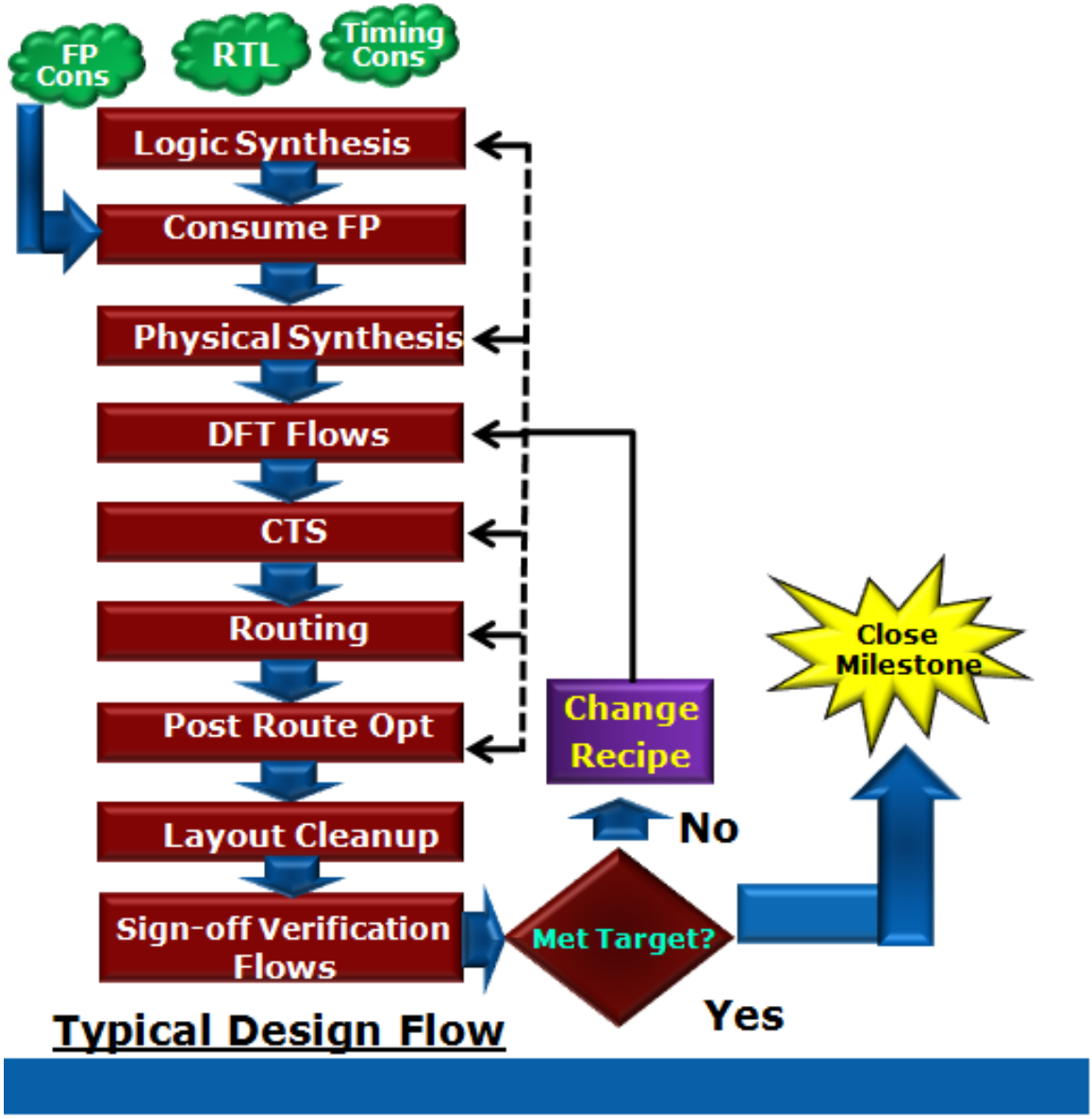


Figure 1.1 Typical Design Flow

RTL coding is the step of translating the design intention, which is described in plain language such as English or Chinese, into a simulative computer language. In this way, simulation can be performed to verify design intention. Additionally, it must be possible for the RTL code to be turned into hardware successfully in later stages. In other words, the RTL code must be synthesizable. The two major languages used in this field are Verilog and VHDL.

Function verification is the process of verifying that the RTL code, created in either Verilog or VHDL, can perform the intended functions defined in the product specification. The main approach used in this process is simulation, with the aid of test benches that describe the system's operations .A large number of test benches must be created in a typical project to cover as many chip operation scenarios as possible.

The aim of function verification is to find design problems in RTL code or to find *bugs* as they are commonly called. Ideally, the number of bugs should be zero before the flow progresses to the next step.

Logic synthesis turns the verified RTL code into real circuit hardware by using sophisticated algorithms. The main task involved in this process is mapping the logic functions defined in the RTL code to standard cells in a selected ASIC library, where each standard cell has a predefined function. The result of this step is a netlist, which has the instantiations ofthe cells used and a description of their interconnections.

Place and route is the process of laying out those cells and interconnecting wires in an automatic way. It is impossible to accomplish the layout process manually in any design of sufficient complexity.

After place and route, it must be functionally verified that the resultant physical entity can still achieve the design functionality defined in the RTL code. This physical entity must also be checked against design rules for manufacturability. The timing aspect of the design is examined at this stage as well. The final step is tapeout, in which the design data is generated in a certain format and sent to a manufacturing facility for fabrication.

Without question,a design flow is a complicated system that includes many commercial CAD tools, as well as many tools or scripts developed in-house. However, no matter how many tools or scripts are integrated in the flow, an ASIC design flow is yet characterized by two key purposes: **create** and **check**.

The process of creation refers to the activities of creating hardware, such as RTL coding, logic synthesis, and place and route. IC implementation is a process of creating hardware. The fundamental elements of an IC chip are CMOS transistors, bipolar transistors, capacitors, and resistors. In the world of integration, the basic building blocks are logic gates, memories, and special function macros. They are composed of those fundamental elements but at one abstraction level higher.

The creation process has two facets: creating the chip **logically** and creating the chip **physically**.

Logic creation must be realized by real logic gates, memories, and special modules. Then this logic representation of the design is constructed physically so that a chip is produced to perform its intended functions.

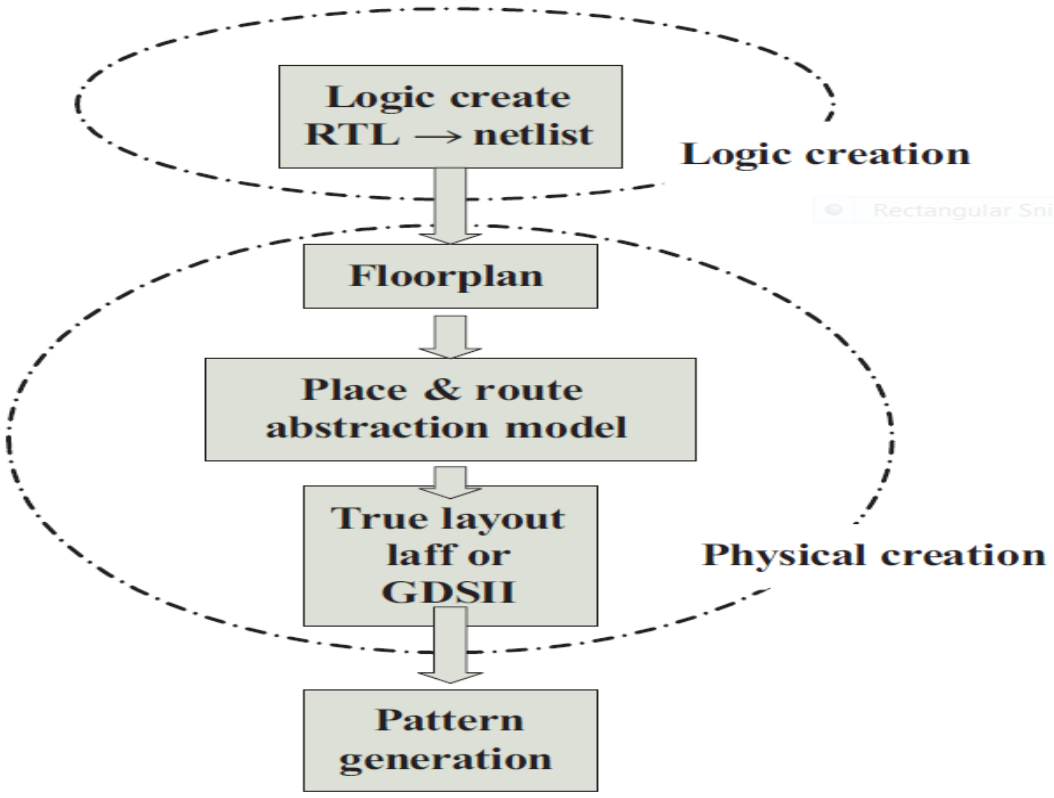


Figure 1.2 Backend Design Flow

The requirements for this creation process are correctness and promptness. Structure correctness must be guaranteed by the construction. This is the most important issue in implementation.

This structure correctness depends heavily on the quality of the CAD tools. Promptness can be achieved by employing the hierarchical design methodology. In other words, the divide- and-conquer approach can simplify the complicated task so that the amount of design data handled at one time is significantly reduced. Promptness can also be improved by using more powerful CPUs and larger memories.

Sometimes, promptness can also benefit from allocating more disk space to the project. In this way, different implementation ideas are tested in parallel, thus using less time than testing them in sequence. This approach can be characterized as "trade space with time (schedule)."

During the hardware creation process, extra effort must be taken to ensure that the resultant chip can be tested for manufacturing defects so that no bad parts are accidentally delivered to customers. This is design for testability (DFT). It is different than functional testing, which checks for functional or logic bugs that exist in all of the dies, if they exist. DFT looks for manufacturing defects that are introduced in the manufacturing process, not in the IC implementation process.

This type of work is very algorithm-intensive and requires a huge amount of computation and a huge amount of memory storage (up to 128 G today). Using abstraction models can significantly reduce the amount of information handled by the tools and thus speed up the process and increase the manageable design size.

CHAPTER 2

LITERATURE REVIEW

The literature review consisted of a range of topics varying from **Static Timing Analysis**, understanding various aspects of timing constraints especially **slack**, **Slack Budgeting** and **Slack Management** and **Power Dissipation** trends .This chapter is a tour through the topics which were relevant and instrumental in every aspect of the work.

2.1 STATIC TIMING ANALYSIS

"Do I meet timing?" or more specifically put "Will the silicon work at the targeted speed with high timing yield?" Static Timing Analysis (STA) promises a fast, exhaustive, safe and simple answer to this question.

STA is based on a lot of simplifications to model a dynamic behavior in a static way. It is also working with models being derived from other models, which may be derived from measurements or even a spec. Measurements may be outdated at the time the designs hits the fab, the spec may be not kept. Each of these layers introduces an error which should be bounded, so the designer is on the safe side and final timing yield is acceptable.

The STA is ***static*** **s**ince the analysis of the design is carried out statically and does not depend upon the data values being applied at the input pins. This is in contrast to simulation based timing analysis where a stimulus is applied on input signals, resulting behavior is observed and verified, then time is advanced with new input stimulus applied, and the new behavior is observed and verified and so on.

The more important aspect of static timing analysis is that the entire design is analyzed once and the required timing checks are performed for all possible paths and scenarios of the design. Thus, STA is a complete and exhaustive method for verifying the timing of a design.

2.2 STA FLOW

Given a design along with a set of input clock definitions and the definition of the external environment of the design, the purpose of static timing analysis is to validate if the design can operate at the rated speed. That is, the design can operate safely at the specified frequency of the clocks without any timing violations.

Figure 2.1 shows the basic functionality of static timing analysis. The DUA is the design under analysis. Some examples of timing checks are setup and hold checks. A setup check ensures that the data can arrive at a flip-flop within the given clock period. A hold check ensures that the data is held for at least a minimum time so that there is no unexpected pass-through of data through a flip-flop: that is, it ensures that a flip-flop captures the intended data correctly. These checks ensure that the proper data is ready and available for capture and latched in for the new state.

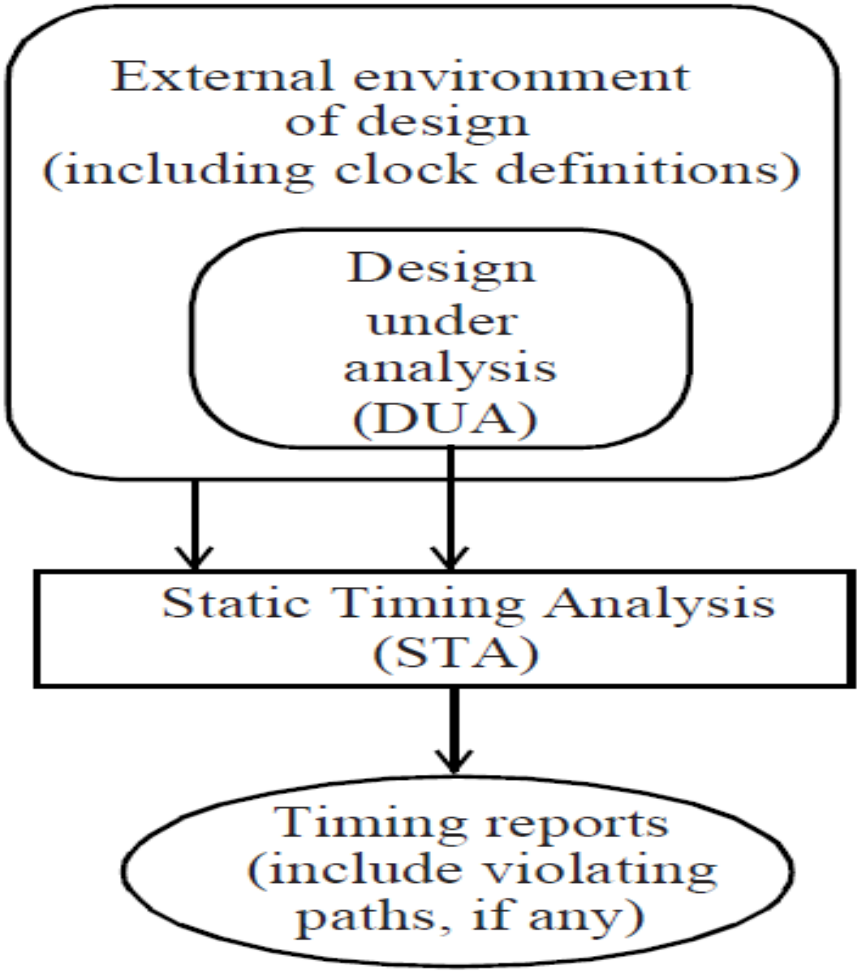


Figure 2.1 STA design flow.

2.3 STA IN CMOS DIGITAL DESIGNS

In a CMOS digital design flow, the static timing analysis can be performed at many different stages of the implementation. Figure 2.2 shows a typical flow.

STA is rarely done at the RTL level as, at this point, it is more important to verify the functionality of the design as opposed to timing. Also not all timing information is available since the descriptions of the blocks are at the gate level, the STA is used to verify the timing of the design. STA can also be run prior to performing logic optimization - the goal is to identify the worst or critical timing paths

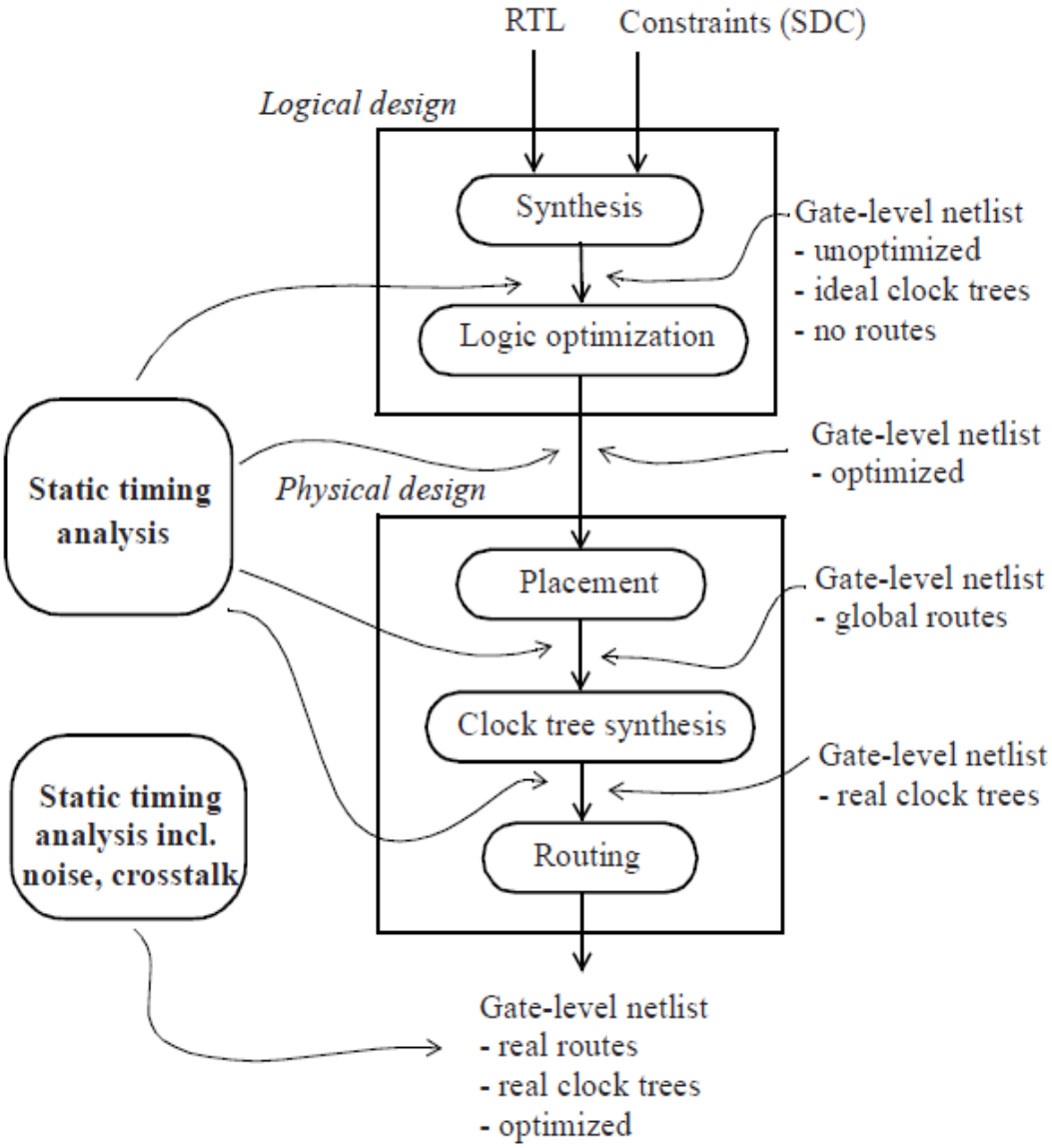


Figure 2.2 STA in CMOS digital design flow.

.

STA can be rerun after logic optimization to behavioral level. Once a design at the RTL level has been synthesized to the gate level, the STA is used to verify the timing of the design. STA can also be run prior to performing logic optimization - the goal is to identify the

worst or critical timing paths. STA can be rerun after logic optimization to see whether there are failing paths still remaining that need to be optimized, or to identify the critical paths.

At the start of the physical design, clock trees are considered as ideal, that is, they have zero delay. Once the physical design starts and after clock trees are built, STA can be performed to check the timing again. In fact, during physical design, STA can be performed at each and every step to identify the worst paths.

In physical implementation, the logic cells are connected by interconnect metal traces. The parasitic RC (**R**esistance and **C**apacitance) of the metal traces impact the signal path delay through these traces. In a typical nanometer design, the parasitics of the interconnect can account for the majority of the delay and power dissipation in the design. Thus, any analysis of the design should evaluate the impact of the interconnect on the performance characteristics (speed, power, etc.). As mentioned previously, coupling between signal traces contributes to noise, and the design verification must include the impact of the noise on the performance.

At the logical design phase, ideal interconnect may be assumed since there is no physical information related to the placement; there may be more interest in viewing the logic that contributes to the worst paths. Another technique used at this stage is to estimate the length of the interconnect using a wireload model. The wireload model provides estimated RC based on the fanouts of a cell.

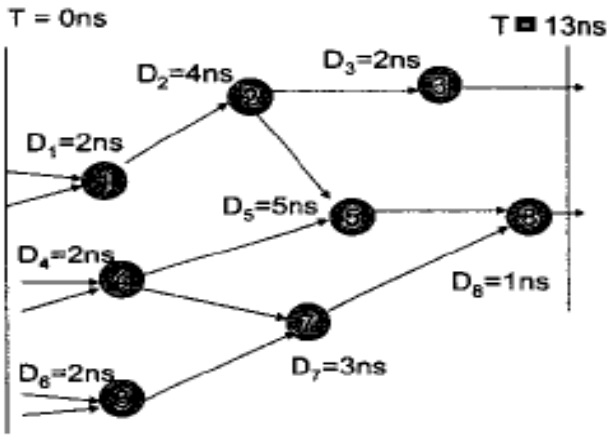
Before the routing of traces are finalized, the implementation tools use an estimate of the routing distance to obtain RC parasitics for the route. Since the routing is not finalized, this phase is called the ***global route*** phase to distinguish it from the ***final route*** phase. In the global route phase of the physical design, simplified routes are used to estimate routing lengths, and the routing estimates are used to determine resistance and capacitance that are needed to compute wire delays. During this phase, one can not include the effect of coupling. After the detailed routing is complete, actual RC values obtained from extraction are used and the effect of coupling can be analyzed. However, a physical design tool may still use approximations to help improve run times in computing RC values.

An extraction tool is used to extract the detailed parasitics (RC values) from a routed design. Such an extractor may have an option to obtain parasitic with small runtime and less accurate RC values during iterative optimization and another option for final verification during which very accurate RC values are extracted with a larger runtime.

2.4 SLACK BUDGETING

Delay budget is a delay each component of a design ,can tolerate under a given timing constraint. Delay budgeting has been widely exploited through the CAD flow to improve the quality.

For each application we go through synthesis and place and route stages in order to obtain accurate results. In some applications, optimal delay budgeting can speed up runtime of place-and-route up to 2 times.



Node	Delay (nsec)	Budget A	Budget B
1	2	-	-
2	4	-	1
3	2	5	4
4	2	5	-
5	5	-	-
6	2	5	-
7	3	2	7
8	1	-	-
Total Budget	--	17	12

Figure 2.3 A Delay Budgeting example.

2.5 OVERVIEW OF BUDGET MANAGEMENT

Given a directed acyclic-graph (DAG) with timing constraints,the budget management problem is to assign an incremental delay to each vertex such that the sum of delays is maximized without violating the given timing constraints. During budget allocation, each net should be assigned a non-zero budget and avoid an even budget distribution, to prevent over-constraining the circuit . The first approach to the budget management problem was zero-slack assignment (ZSA), which was used to generate performance constraints for layout. Slack would be assigned iteratively along a path identified as having minimum slack by adding an incremental delay to each vertex on the path

Timing optimization during placement has typically been performed in an iterative fashion. An existing placement is analyzed for timing-problems. This information is used to improve either the current placement or a subsequent placement run.

These methods generally fall into two categories:

- ➔ path-based methods
- ➔ net-based methods.

Path based methods directly optimize for reducing the lengths of critical paths during placement, whereas the net-based methods try to reduce the lengths of individual nets that belong to these critical paths.

Net based methods tend to be faster because they do not have to perform longest path analysis during placement.

However, path-based methods better capture overall timing problems and have better chances of convergence

2.6 POWER DISSIPATION TRENDS

To this point in time, Moore's Law has predicted with remarkable accuracy that engineers would be able to pack more transistors onto a single die. However, with each technology shrink, new challenges must be overcome to ensure that Moore's Law continues. Excessive chip power dissipation, a specific challenge in the late 80s, forced the industry to make the fundamental shift from bipolar to CMOS technology (Figure 2.4). Today, we are again at that junction in the process technology timeline, where as geometries shrink,

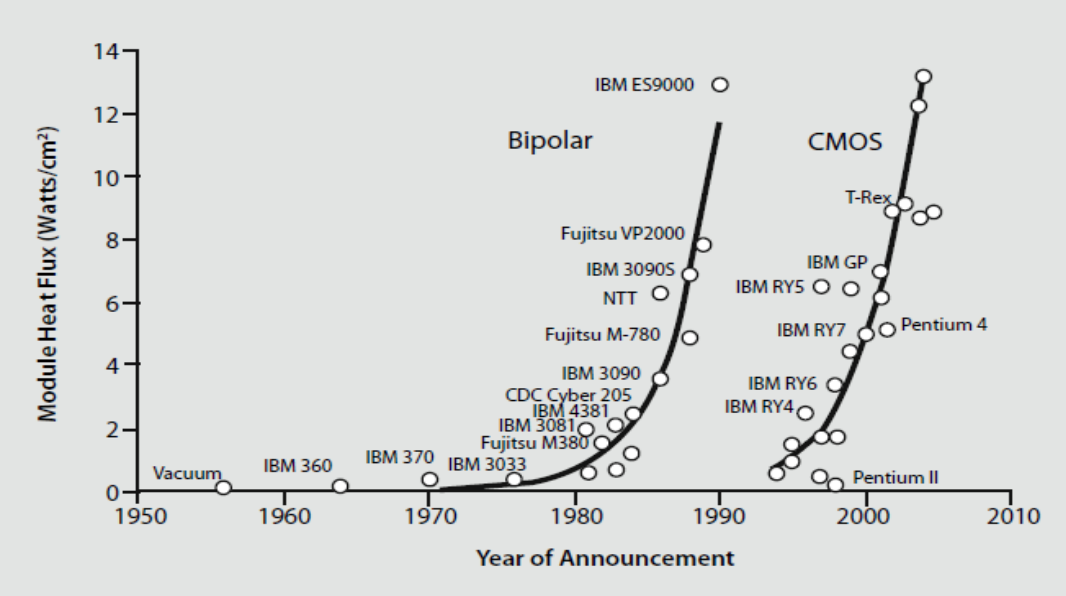


Figure 2.4: Popular processor power dissipation trends by silicon technology

power has again become the dominant challenge. Unfortunately, unlike in the 80s, there is no obvious new process technology on the horizon that can solve the power problem and keep Moore's Law alive. As a result, today's high-end processors are actually multiple lower-frequency processors being used in parallel. While this increases overall throughput, individual processor performance remains limited, additional area and power are consumed, and it dramatically complicates embedded and system-level software. Solving the power reduction challenge today is a matter of applying dedicated engineering and design techniques, which while providing compelling benefits, also incur substantial costs in terms of design complexity.

Moore's Law Boon or Curse?

For several decades Moore's law has served as a beacon to predict device density and its subsequent power dissipation. The law, named after Intel founder Dr. Gordon Moore, states that semiconductor transistor density, and hence performance, doubles roughly every 18 months. Dr. Moore made his prediction in the 1970's and history supports his conclusion, as shown in (Figure 2.4) . It is fascinating to observe how closely the microprocessor industry has followed Moore's law. Incidentally, it is useful to note that this trend also applies to components other than microprocessors. Although little industry data is available, the evolution of CMOS technology clearly suggests the same trend.

In the 70s and 80s, Moore's law served as a point of discussion - and much disbelief. Many people never expected to see such densities. Moreover, once the transistor densities were converted into power dissipation the disbelief in Moore's law was further amplified because the heat flux densities began to rival the space vehicle re-entry temperatures.

As a side note, the current lack of cooling technology and inaction on the thermal management industry to develop it further may be attributed directly to the disbelief of Moore's law. So, when viewed from the thermal management standpoint, it appears that Moore's law has been rather a curse.

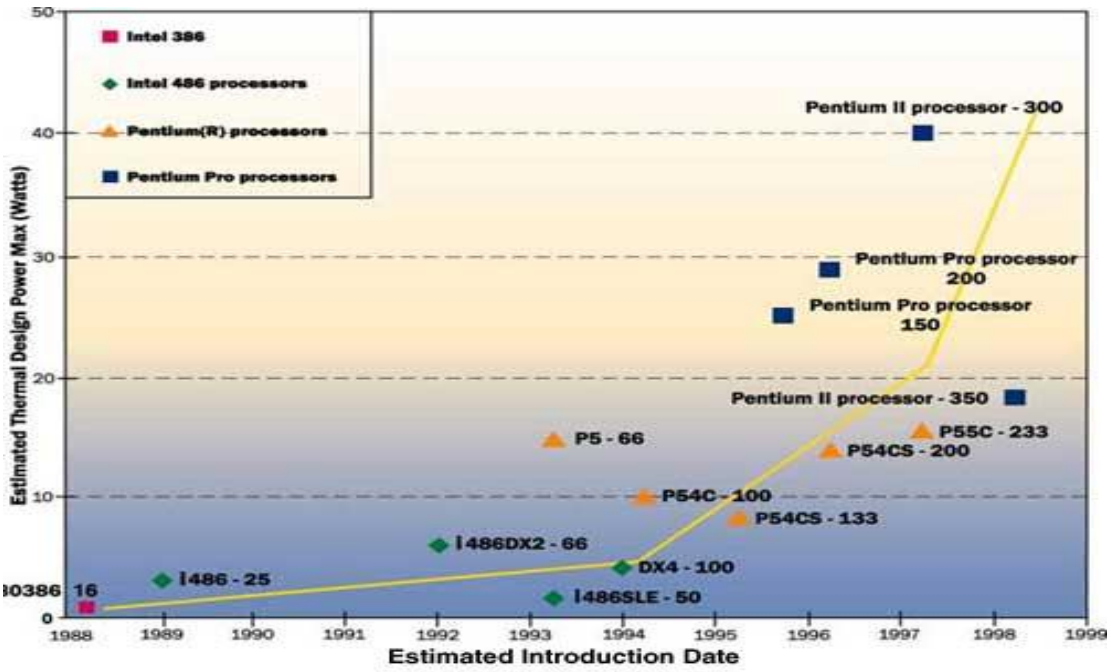


Figure 2.5: Time-line plot of Intel corporation microprocessor power dissipation

Gordon Moore's insightful observation on device advancement has had a direct correlation with power dissipation. Figure 2.5 shows the power dissipation of some representative Intel microprocessors as a function of time. The scatter in data clearly suggests technology advancements to reduce power. But the overall trend is the exponential increase in power dissipation with time.

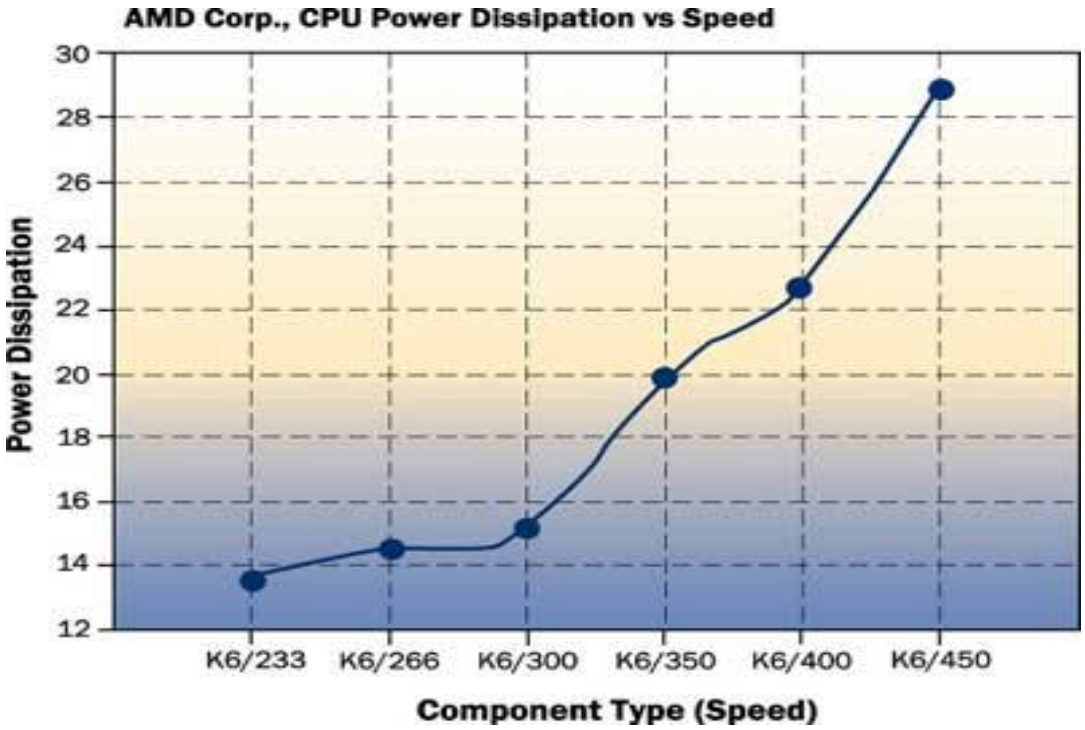


Figure 2.6: Power dissipation of the AMD Corp. K6 family of microprocessors as a function of time (clock speed reflects time history).

With the current device and manufacturing technologies, all industry beacons point to significant power increases with time. None of the plots displayed indicates that power dissipation at the chip level - and subsequently at the system level - is declining or even reaching a plateau.

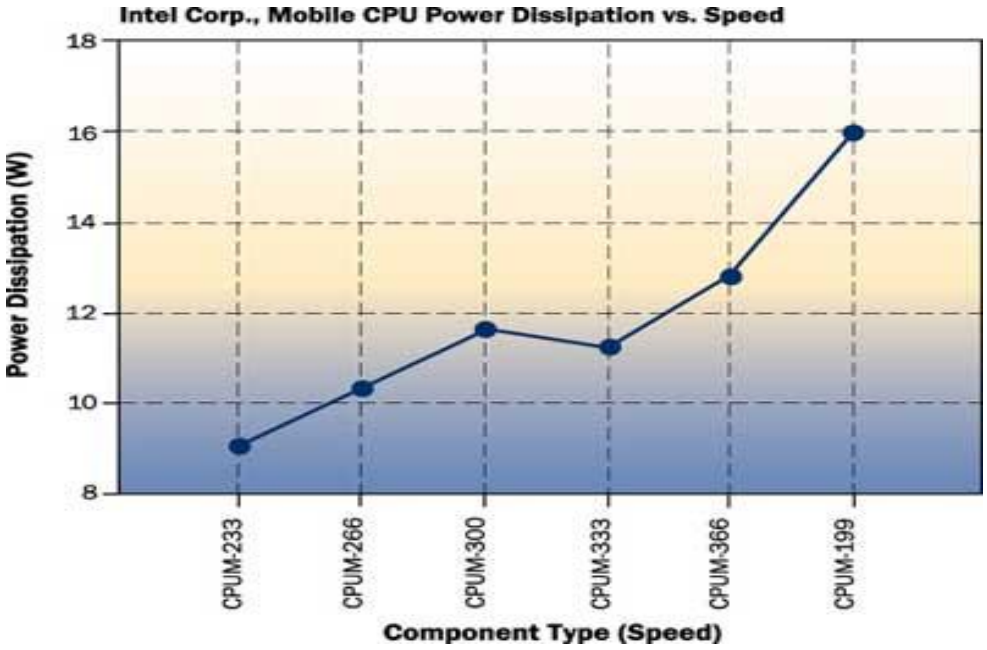


Figure 2.7: Power dissipation of the Intel Corp. family of mobile microprocessors as a function of time (clock speed reflects time history).

Almost from inception, power dissipation and its adverse impact on electronics has been a point of concern for the industry. Whether cooling vacuum tubes or sophisticated integrated circuits, an integral part of the design cycle has always been the consideration of power dissipation and consumption issues.

2.7 REASON FOR POWER DISSIPATION INCREASE

Two causal agents underlie this increase in power dissipation: speed and the number of gates on the silicon. Over the years, there has been much effort to reduce the size of the devices (transistors, diodes, etc.) on the silicon wafer. More than a decade ago, 1.5 micron appeared to be the limit and this caused much enthusiasm about the possibilities for this new technology.

Today, by contrast, geometries down to 45 nm are gaining in popularity and soon to become the standard. The 45 nm device size, like its predecessors has created much excitement as chip designers contemplate "system-on-chip" products. Figure 2.8, illustrates Intel device technology as a function of time.

The timeline chart clearly shows a rapid decline in part size as a function of time, and this trend shows no sign of reaching the asymptotic condition (plateau). If we define the lower size limit to be the atomic or molecular scale, it appears that the downward trend will continue accompanied by the challenge to develop production hardware capable of making parts that small (transistors, diodes, etc.).

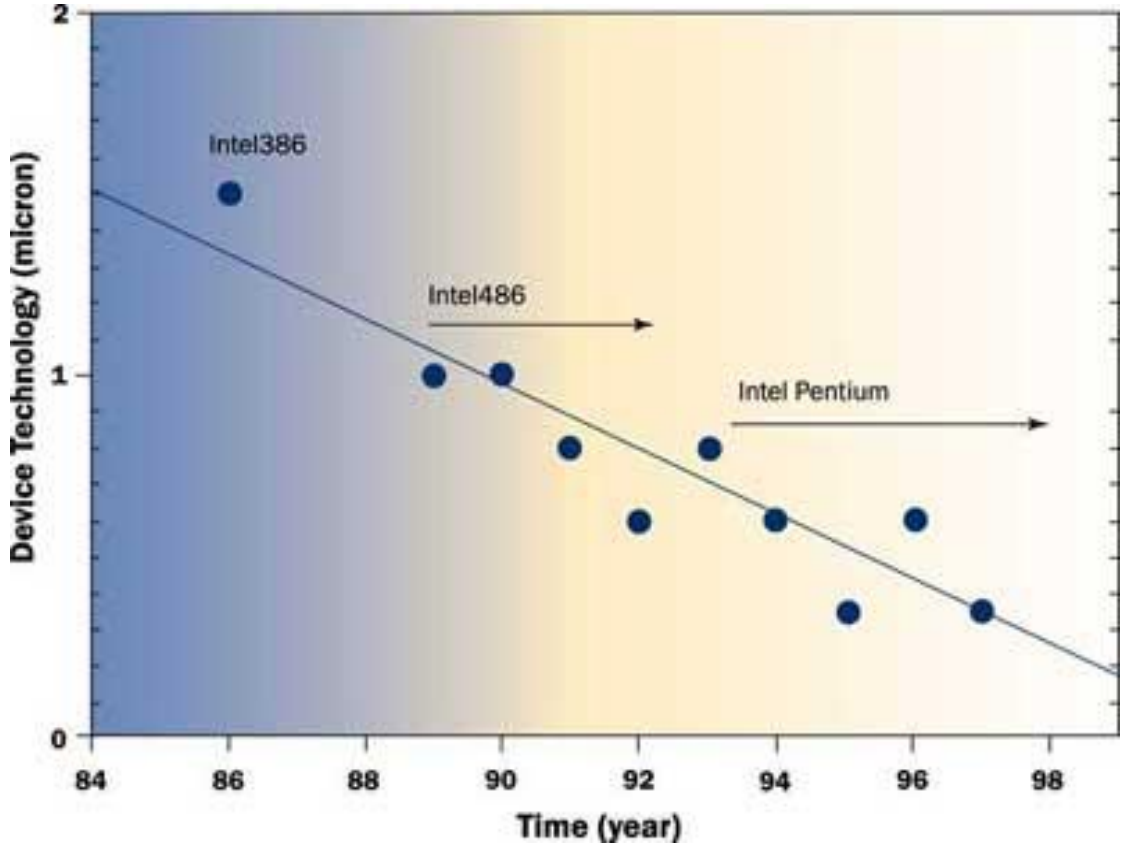


Figure 2.8: Intel device technology as a function of time.

It is interesting to consider what happens if we superimpose Figures 2.5 and 2.8. It is clear that as technology has shrunk part size, power dissipation - attributable to increases in speed - has continued to rise.

The most intriguing and perhaps confusing point involves power. Power consumption per gate has decreased dramatically from 0.64 to 0.15 W/Gate/MHz. Yet, overall power dissipation has risen dramatically for just about every component on the market. This phenomenon can be attributed to two factors: the increased number of devices on a die (gates/device) and speed. A simple calculation demonstrates the effect of speed on power dissipation. Suppose we have two devices, 500 and 600 MHz, respectively, with each containing five million parts (gates). If we use today's technology, we can determine that the power dissipation in such a device is 375 W and 450 W for the 500 and 600 MHz dies, respectively. Such power dissipation in a spatially constrained environment is, at times, beyond imagination.

To gain an appreciation for these power levels and their impact, let us compare a 100W light bulb with a ball grid array (BGA) device that dissipates 25 W

	Light Bulb	**BGA Package**
Power dissipation	100W	25W
Surface area	$106cm^2$ (bulb surface area)	$1.96cm^2$ (die area)
Heat flux	$0.9\ W/cm^2$	$12.75\ W/cm^2$

Figure 2.9: Comparison of Heat Flux between a Light Bulb and a BGA Package.

The heat flux of a light bulb is approximately 13 times less than that of the BGA. Furthermore, the light bulb can dissipate heat through an entire room, whereas the BGA is confined to the small space provided between two PCBs. Certainly, we do not want to touch a light bulb that has been on for a while. Thus you can imagine the potential temperature of a device residing on a PCB with such power densities. We can therefore conclude that the combination of higher speed, number of parts on the die and spatial constraints collectively result in power dissipation and a thermal management situation that is challenging, if not impossible, to meet with today's technology and design constraints, e.g., $T_{j,max} = 125\ ^oC$.

2.8 SOURCES OF POWER DISSIPATION IN CMOS

Across a variety of electronics industry segments, device power dissipation has become the important player in ASIC design. A successful chip design requires low power consideration, and determination of the locations (spots) on the die where power dissipation occurs. The design objective is to minimize device junction temperature because temperature directly impacts die performance (speed, sinking and sourcing capability). Therefore, understanding the source of the power may shed some light onto what the future may offer and may perhaps expose the limits that currently confronts the designer.

As device dimensions continue to shrink in this deep-submicron or nanometer realm, the physical characteristics of semiconductor devices become increasingly complex. Designers are faced with new challenges for accurately predicting circuit performance. In addition, as both the clock speed and the number of devices on the chip increase, power consumption is emerging as a major concern. For portable applications, long battery life is extremely important to enhance the competitiveness of the end product. For non-portable applications, excessive heat generated by high power consumption can lead to device reliability problems.

Irrespective of application, there are two common sources of power consumption that engineers must attend to: **dynamic** and **static power**.

Dynamic Power

Dynamic power, considered the "active" component of power consumption, is consumed when the device is changing state, and includes both **switching power** and **short circuit power**. Short-circuit power is dissipated in CMOS logic when both complementary transistors are briefly turned on as the switch changes state, and is generally negligible. Switching power is dissipated by the charging and discharging of capacitors in logic and interconnects as circuits switch. For ASIC designs, the dynamic power consumed by the charging or discharging of the output load external to the cell is classified as switching power, while the dynamic power dissipated within the cell is generally classified as the internal power.

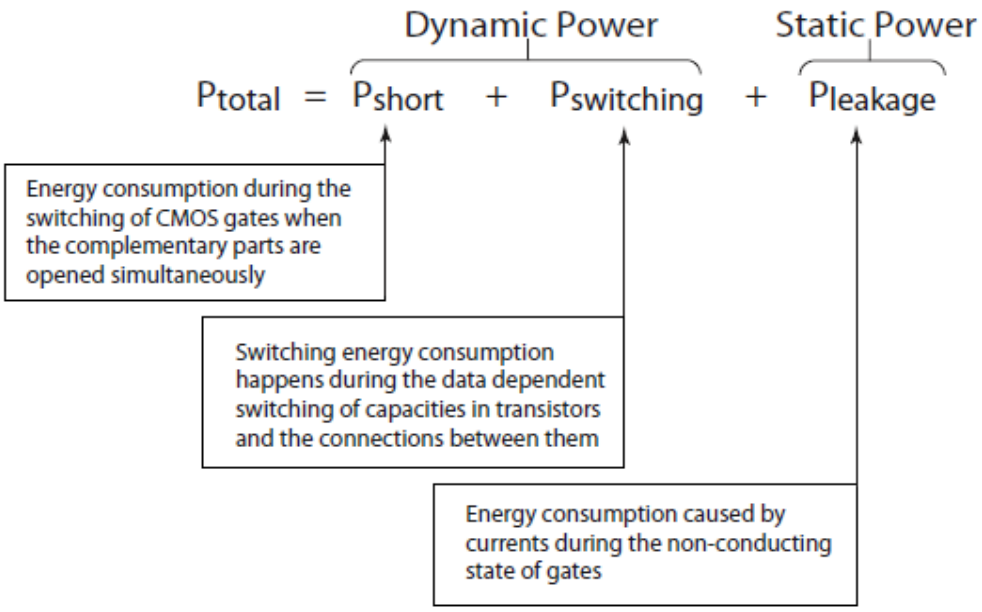


Figure 2.10: Sources of power consumption

Switching Power

The switching power is determined by the capacitive load and the frequency of the logic transitions on a cell output.

$$\textbf{Switching Power} = ½ * Cload * V^2 * f$$

where the total load capacitance (*Cload*) is the sum of the net and gate capacitances on the driving output, and the frequency (*f*) is the rate of state transitions.

Cell Internal Power

The internal power is caused by the charging of internal loads as well as by the short-circuit current between the N and P transistors of a gate when both are on.

$$\textbf{Internal Power} = (½ * Cint * V2 * f) + (V * Isc)$$

Short Circuit Power

As the input signals transition, both N and P type transistors can be on simultaneously. During this time, current *Isc* flows from Vdd to Gnd causing the dissipation of short-circuit power. The short-circuit power is affected by the dimensions of the transistors, the load capacitance on the output, and the transition time of the input signals. Circuits with slow transition times can dissipate excessive short circuit power as both N and P transistors are on for an extended period. Short-circuit current increases with slower signal transition times.

The diagram below describes the power components for a simple buffer cell

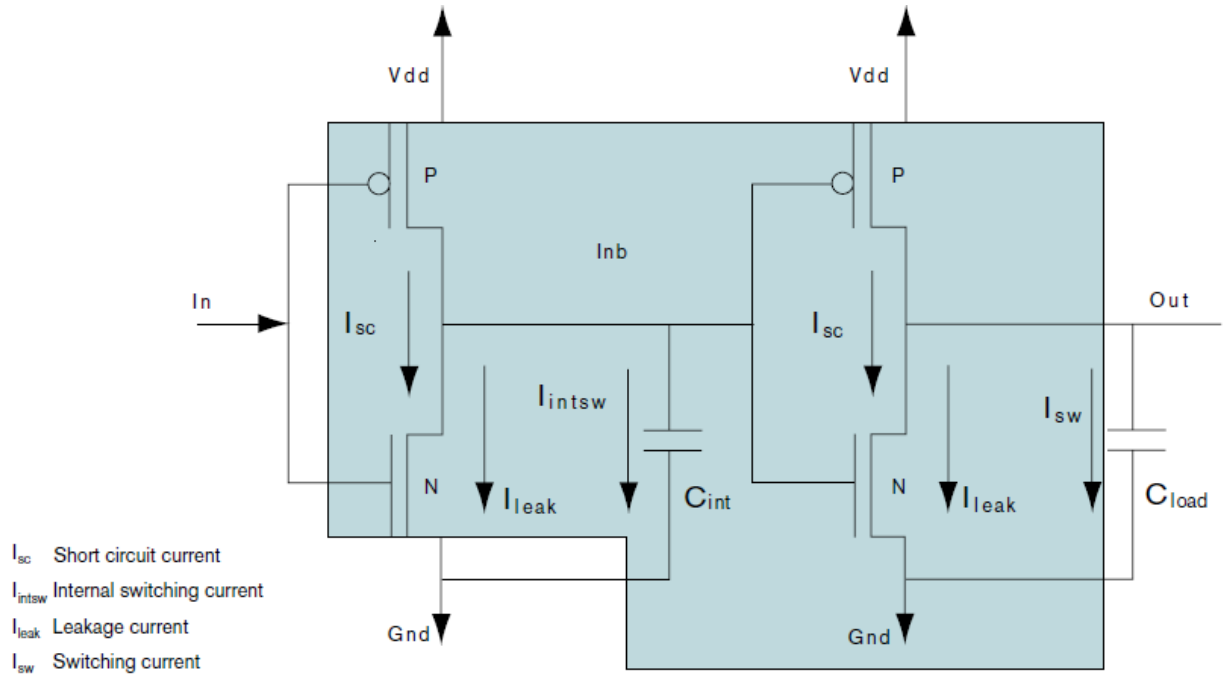


Figure 2.11: Components of Power Dissipation

The leakage current *Ileak* can vary based on the transistor states. For example, when the input signal *In* is high, and the N transistor is on, the leakage will differ compared to when the N transistor is off. When a rising signal is applied at the input, internal power is dissipated due to *Isc* and *Iintsw*. During the transition from low to high, the N transistor turns on and the P type transistor turns off resulting in *Isc* from Vdd to Gnd. Additionally, internal switching power is incurred in charging and discharging of *Cint*. The switching power on the Out net is due to *Isw* charging and discharging *Cload*. The greater this dynamic switching current is, the faster you can charge and discharge capacitive loads, and your circuit will perform better.

Static Power

Static power is dissipated in several ways. Some are due to the reverse-biased diode leakage from the diffusion layers and the substrate, but the largest percentage of static power results from source-to-drain sub-threshold leakage current. This is caused by reduced threshold voltages which prevent the gate from completely turning off and hence allow this leakage current (*Ileak)*.

The leakage power is dependent on the voltage, temperature and state of the transistors.

Leakage Power = $V * I_{leak}$.

Leakage current is usually small, but in large ultra low power designs (such as memories), it can represent the majority of the standby power consumed by the device

The adoption of smaller process geometries and lower supply voltages has resulted in an exponential growth in static power. Once ignored as a negligible source of power consumption, static power is now the dominant source of power consumption

Many analog circuits have static power, since they are continuously "on" and drain power at a relatively stable rate.

Switching power is the dominant source of power drain on a CMOS chip. Each time a transistor switches state, it either pumps charge into a capacitor (the load), or drains the capacitor to ground. In a full cycle, charge is taken from the rail and pumped to ground. This produces a current, which, multiplied by the rail voltage, equals power. If the transistor does not switch as often, less charge moves and the power is lower. This is why limiting the activity of transistors is an excellent way to limit power.

The power problem for the CMOS devices did not surprise IC foundries, and they have been aggressively lowering their voltages. But for anyone who has watched power consumption grow in microprocessors even as voltage is lowered, it is clear that simply lowering voltages is not enough.

From the above discussion, it is clear that significant work lies ahead for reducing and managing power dissipation in CMOS devices. With the market desire to shrink size and put

systems on chips, advances in circuit design and materials are clearly required for managing this situation, since neither speed nor size is helping power reduction. Otherwise, Moore's law continues to be applicable and the power trends observed here will continue to apply.

Power Optimization

The increasing speed and complexity of today's designs implies a significant increase in the power consumption of very-large-scale integration (VLSI) chips. To meet this challenge, researchers have developed many different design techniques to reduce power. The complexity of today's ICs, with over 100 million transistors, clocked at over 1 GHz, means manual power optimization would be hopelessly slow and all too likely to contain errors. Computer-aided design (CAD) tools and methodologies are mandatory.

With the sources of power dissipation in modern integrated circuits well understood, we can start to explore the various sorts of power reduction techniques. Power or energy minimization can be performed at many stages in the design process and may address different targets such as dynamic or static power.

2.9 POWER OPTIMIZATION

The increasing speed and complexity of today's designs implies a significant increase in the power consumption of very-large-scale integration (VLSI) chips. To meet this challenge, researchers have developed many different design techniques to reduce power. The complexity of today's ICs, with over 100 million transistors, clocked at over 1 GHz, means manual power optimization would be hopelessly slow and all too likely to contain errors. Computer-aided design (CAD) tools and methodologies are mandatory.

With the sources of power dissipation in modern integrated circuits well understood, we can start to explore the various sorts of power reduction techniques. Power or energy minimization can be performed at many stages in the design process and may address different targets such as dynamic or static power.

Power optimization at different levels of abstraction

1. Architecture level or System level.
2. RTL level.
3. Gate level.
4. Layout level.

Architecture Level or System Level

A significant amount of recent research has sought to reduce the power consumption of processors, including high-performance general purpose microprocessors. Power consumption can be reduced at many levels, from the circuit level to the application level. Power reduction available due to optimization at the architecture level identifies three broad categories of wasted energy that either has been, or could potentially be, targeted for reduction. Those categories are program waste, speculation waste, and architectural waste. The focus of the research is the reduction of power and energy dissipation in high-performance, high instruction level parallelism processors. These categories of program waste encompass much of the ongoing research in architectural power reduction - assuming solutions that do not fundamentally change the processor or its perfomance.

Speculation waste results from speculative fetching and execution of instructions beyond unresolved branches, inevitably leading to wasted energy when those branches are mispredicted. These instructions are fetched into the processor and consume pipeline resources and energy, but are never committed and do not change permanent processor state. Examples of energy and performance optimizations that target this type of waste include pipeline gating and aggressive branch predictors.

Architectural waste results from the static sizing of architectural structures, such as caches, instruction queues, branch predictors, etc. Memory array structures are often designed to hold a large amount of data in order to obtain good overall performance, but a given program usually cannot exploit this at all times. Instruction queues are designed with a large number of entries when often there is little parallelism in the running code. For performance reasons, these structures are made as large as possible. Making them too small can also waste power (causing more miss traffic, increasing misspeculation, etc.). Examples of power and performance optimizations that target this type of waste include selective cache ways and dynamically resized issue queues.

RTL (Register Transfer Language) Level

The most common design methodology for reducing ASIC power consumption at RTL level is through the use of RTL clock gating. This feature causes inactive clocked elements to have clock gating logic automatically inserted which reduces power consumption on those

elements to zero when the values stored by those elements are not changing. The RTL clock gating feature allows easily configurable, automatically implemented clock gating which allows maximal reduction in power requirements with minimal designer involvement and no software involvement. The methodology was proven in a 200K-gate ASIC, which implemented full scan testing and used RTL clock gating to reduce its power consumption by two-thirds.

In the traditional synchronous design style used for most HDL and synthesis-based designs, the system clock is connected to the clock pin on every flip-flop in the design. This results in three major components of power consumption: 1) power consumed by combinatorial logic whose values are changing on each clock edge; 2) power consumed by flip-flops (this has a non-zero value even if the inputs to the flip-flops, and therefore, the internal state of the flip-flops, is not changing); and 3) the power consumed by the clock buffer tree in the design. RTL clock gating had the potential of reducing both the power consumed by flip-flops and the power consumed by the clock distribution network

RTL clock gating works by identifying groups of flip-flops which share a common enable term (a term which determines that a new value will be clocked into the flip-flops). Traditional methodologies use this enable term to control the select on a multiplexer connected to the D port of the flip-flop or to control the clock enable pin on a flip-flop with clock enable capabilities. RTL clock gating uses this enable term to control a clock gating circuit which is connected to the clock ports of all of the flip-flops with the common enable term. Therefore, if a bank of flip-flops which share a common enable term have RTL clock gating implemented, the flip-flops will consume zero dynamic power as long as this enable term is false.

Gate Level

At gate level the most effective method for power optimization is through gate sizing. Gate sizing technique involves downsizing certain gates which meet a specific set of constraints. These constraints involve the timing constraints, library constraints and the design constraints. A timing surplus in a path is utilized to replace a gate with a smaller one in a standard cell library by applying a theory of optimal gate sizing. Layout is then modified by engineering change order (ECO), and timing is verified by a static timing analyzer (STA). Since timing is verified by STA, an industry standard scheme, no new timing issue rises. It is

good enough if delay model is accurate only for good guidance of gate sizing. This is the method that has been implemented in this project and the results are also shown.

Layout level

Physical design fits between the gate-level specification and the geometric (mask) representation known as the layout. It provides the automatic layout of circuits minimizing some objective function subject to given constraints. Depending on the target design style (General Cells, Standard Cells, Gate Arrays, FPGAs), the packaging technology (printed circuit boards, multichip modules, wafer-scale integration) and the objective function (area, delay, power, reliability), various optimization techniques are used to partition, place, resize and route gates.

Under a zero-delay (glitch-less) model, the switching activity of gates remains unchanged during layout optimization, and hence, the only way to reduce power dissipation is to decrease the load on high switching activity gates by proper netlist partitioning and gate placement, gate and wire sizing, transistor reordering, and routing. Layout problems become more complicated under a real-delay model, which accounts for glitches in the circuit, because layout optimization operations influence the glitch activity in ways that cannot be accurately and reliably predicted.

In the recent past, post-layout optimization techniques (such as buffer and wire sizing, local restructuring and re-mapping) for power reduction (or area and delay recovery given a fixed power budget) have become commonplace. The advantage of these techniques is that re-synthesis tools allow more global changes to the circuit structure compared to layout tools. At the same time, the re-synthesis tools have access to detailed post-layout information that allows accurate estimation of circuit area, delay and power dissipation.

I

CHAPTER 3

THE PROBLEM STATEMENT AND PROPOSED SOLUTION

3.1 THE PROBLEM STATEMENT

ALGORITHM-

To develop an algorithm to come up with an Intelligent SLACK MANIPULATION STRATEGY which redistributes slack in a manner which is most optimal in terms Power Optimization

IMPLEMENTATION- GATE SIZING and CELL SWAPPING

The Algorithm mainly focuses on:

Recovering leakage power from timing path which has positive margin, where the core algorithm is zero slack algorithm. It is done by two methods:

→Method 1: Use sizing algorithm to safely (DRC check) size down the cells on timing path (improves area, reduces leakage)

→ Method 2: Use swapping algorithm to safely (DRC check) swap to a low Vt equivalent cell on the timing path (reduces leakage & dynamic power)

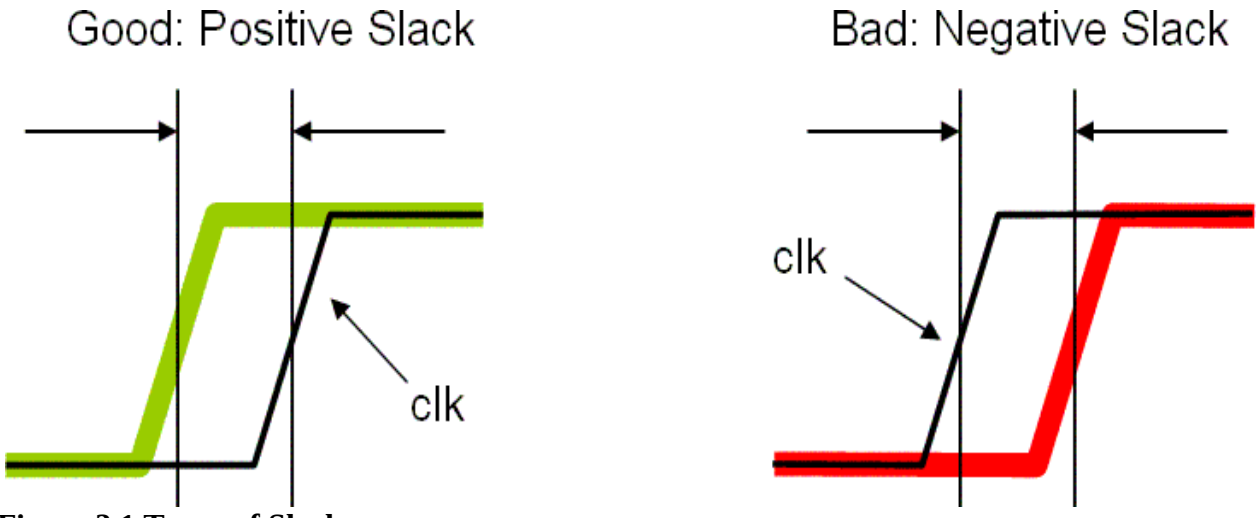


Figure 3.1 Types of Slack

3.2 THE PROPOSED SOLUTION- AN INTELLIGENT SLACK MANIPULATION ALGORITHM

PREREQUISITES

Potential Slack: An Effective Metric of Combinational Circuit Performance

From a timing standpoint, non-critical paths can be characterized by timing ***slack*** of their constituent logic modules.

The slack for a module provides an upper bound of its delay increase without violating timing constraints and, hence, represents a ***"potential"*** capability of obtaining are power reduction.

On the other hand, at physical level, slack can be interpreted as a measure of wiring "freedom" (opposed to delay "constraints") in timing-driven placement and routing, and hence as a predictor of final circuit timing.

A well-known technique for slack management is the ***Zero-Slack Algorithm*** **(ZSA)** . Its improved versions of for applications in placement have also been proposed. **As** its name implies, ***potential slack*** is the slack that can potentially be used for optimizing a circuit (the exact definition will be given in the next section). In this paper, we claim that potential slack is a very effective metric of combinational circuit performance.

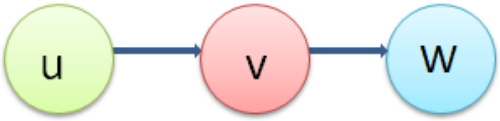
Consider a combinational circuit which consists of a set of ***modules M = (ml , m2 , ..., m,},*** and a set of ***nets N = {nl ,n2 , ..., nl}.*** Naturally, we associate the circuit with a ***directed-acyclic graph G = (V , E),*** where a node ***v* E *V*** denotes a module, and there is an edge **e E *E*** from node ***vi*** to ***vi*** if ***vj*** is an immediatefanout of ***vi*** . *Also,* each node ***v*** is associated with a ***delay d(v),*** which stands for the delay of node ***v*** itself plus the interconnection delay

A ***vector*** of delays ***D(V) = [d(v,) d(v2) ... d(v,)]*** is called ***delay distribution*** of the circuit. For convenience, suppose that the ***arrival times*** for all primary inputs are zero and that the
required times for all primary outputs are the given timing constraints.

A well-known procedure to compute arrival time, ***a(v),*** and required time, ***r(v),*** for each node ***v*** is given recursively by

$$\text{arrival time } a(v) = \max_{u \in FI(v)} (a(u) + d(v))$$

$$\text{required time } r(v) = \min_{w \in FO(v)} (r(w) - d(w))$$



where ***FO(v)*** is a set of fanins of node ***v,*** and ***FO(v)*** a set of fanouts.

Slack of node ***v*** is defined as

$$s(v) = r(v) - a(v)$$

A vector of slacks **S(V)** = ***[s(v1) s(v2) ... s(v,)]*** is called ***slack distribution*** of the circuit.

Total slack of circuit is

$$|S(V)| = s(v_i)$$

The circuit is said to be ***safe*** if and only if

$$S(V) \geq 0.$$

A *slack assignment* is a vector of ***incremental*** delays

$$\Delta D(V) = [\Delta d(v1), \Delta d(v2), \ldots\ldots, \Delta d(vn)] \geq 0$$

which updates the delay distribution from ***D(V)*** to

$$D_{\Delta}(V) = D(V) + \Delta D(V)$$

And hence updates the slack distribution from

$$S(V) \text{ to } S_{\Delta}(V)$$

If

$$S\Delta(V) \geq 0$$

then the slack assignment *D(V)* is said to be ***Effective*** The effective slack of the assignment is

$$|\Delta D(V)| = \sum \Delta d(vi)$$

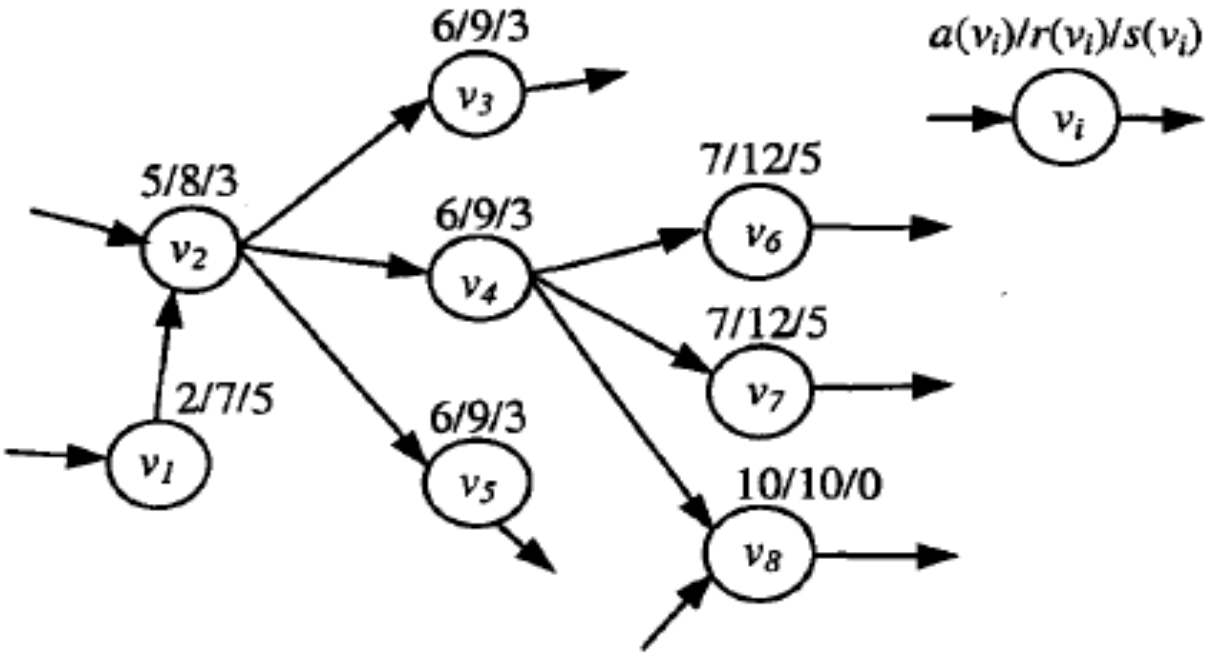


Figure 3.2 Example circuit for slack calculation

Potential Slack for the circuit is The maximum effective slack for all possible slack assignment. The Zero Slack Algorithm is the core algorithm and the slack calculation described is the basic starting point for the algorithm.

CHAPTER 4

TOOLS USED AND METHODOLOGY ADOPTED

The tools used to implement any idea are like the weapons without which you cannot cannot even start the journey. Intricate knowledge and understanding about the tool can go long way to reduce the extra burden and focus on more meaningful aspects of any work.

Tools used for all purposes of development and testing are **Synopsys™ IC COMPILER** and to talk to the tool, the language used for scripting was **TCL(Tool Command Language)**. The platform used was **Unix Operating System**.

Other scripting language used for automization at various steps was **PERL (Practical Report Extraction Language)**

This Chapter is a brief peek about the tools used and the methodology adopted to use them.

4.1 TOOLS USED

Synopsys™ IC COMPILER

IC Compiler is an integral part of the Synopsys Galaxy™ Design Platform that delivers a complete design solution, including synthesis, physical implementation, low-power design, and design for manufacturability (DFM). IC Compiler is a single, convergent, chip-level physical implementation tool that includes flat and hierarchical design planning, placement and optimization, clock tree synthesis, routing, DFM, and low-power capabilities that enable designers to implement today's high-performance, complex designs. Widely adopted and recognized as the industry standard for physical implementation, IC Compiler provides best-in-class quality of results (QoR), strong signoff correlation, and powerful DFM capabilities.

Ease of Use

IC Compiler advances ease-of-use with core commands (place_opt, clock_opt, and route_opt) to deliver best out-of the- box results. The IC Compiler GUI provides intuitive and easy-to-use features that enable designers to resolve issues at all design stages. The GUI enables fast analysis, visualization, debugging and repair features.

All of these shared technologies and key advances in IC Compiler enable the Galaxy Design Platform to deliver the best QoR in terms of timing, area, power, rout ability, testability and yield as well as faster TAT and a predictable path to first-silicon success. Today designers are using IC Compiler successfully to tape out numerous complex, high-performance, and low-power designs at 130nm to 45nm and below geometries.

Features

- High throughput for designs in mainstream silicon technologies
- High performance for advanced silicon technologies
- Comprehensive optimization capabilities meet timing, area, power, signal integrity, routability and yield objectives
- Predictability during the implementation process
- Single timer
- Complete netlist-to-GDSII solution for best QoR and TTR
- Signoff
- Highly correlated with golden signoff solutions: PrimeTime SI and StarRC
- Shares common infrastructure and technologies with PrimeTime, common cell delay calculation and SDC constraints to ensure tight correlation
- Improves TTR by eliminating unnecessary margins
- Speeds design closure by using exact signoff timing and extraction information

TOOL COMMAND LANGUAGE

Tcl (Tool Command Language) is a dynamic programming/scripting language based on concepts of Lisp, C, and Unix shells. It can be used interactively, or by running scripts (programs) which can use a package system for structuring, hence allowing to do much with little code. Tcl is available for Linux, Windows, Mac OS X, as well as other platforms, as open-source software under BSD-like license, or as pre-built binaries.

Tcl interfaces natively with the C language. This is because it was originally written to be a framework for providing a syntactic front-end to commands written in C, and all commands in the language (including things that might otherwise be keywords, such as **if** or **while**) are implemented this way. Each command implementation function is passed an array of values that describe the (already substituted) arguments to the command, and is free to interpret those values as it sees fit.

Digital logic simulators often include a Tcl scripting interface for simulating Verilog, VHDL and SystemVerilog hardware languages.

3.2 METHODOLOGY ADOPTED

PART 1

1 Run synthesis and placement to get initial timing data
2 Interactively get logical modules and their connectivity using Synopsys™ IC COMPILER
3 Convert to a adjacency matrix for modeling purposes
4 Run proposed algorithms scripted in TCL and get the timing data
5 Get the Outputs: Outputs will be the cell instance name and the extra delays each Can tolerate without violating timing constraints.
6 Apply the knowledge gathered for Power Optimization

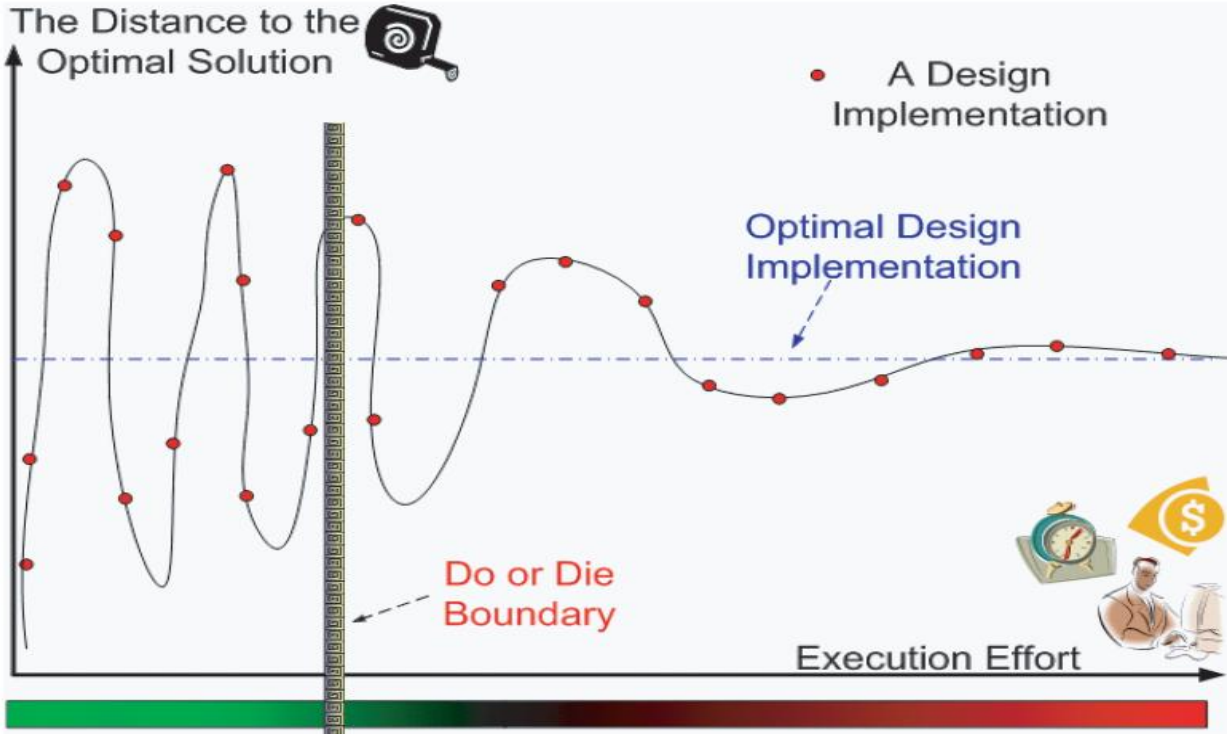


Figure 4.1 Optimum Design Implementation

PART 2

1 Implement downsizing of cells based on the buffer slack it can tolerate
2 Implement swapping of cells to more optimized cells in such a way that timing is not violated.

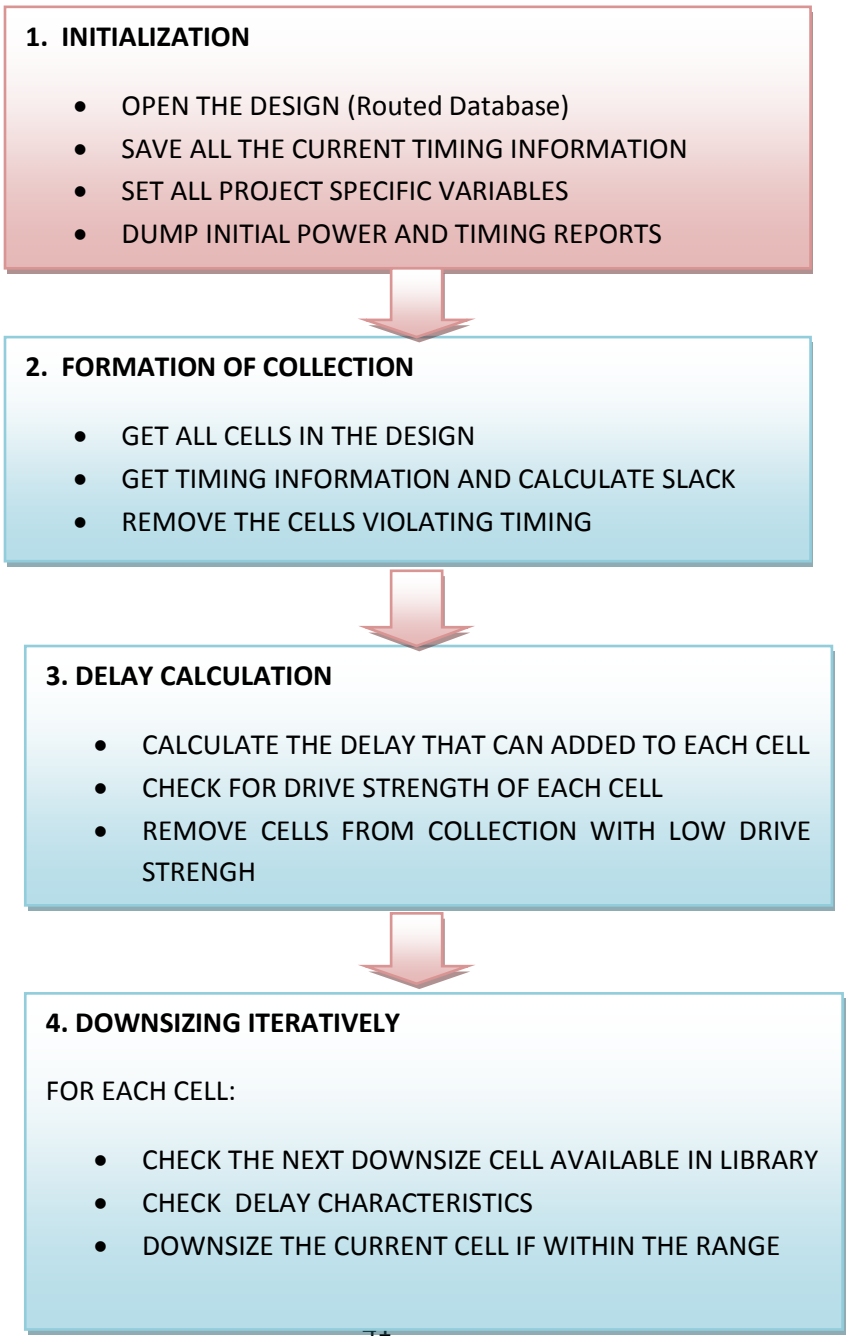
Throughout part 2 of the methodology DRC checks are carried out at vital points in the flow i.e. after downsizing and after swapping.

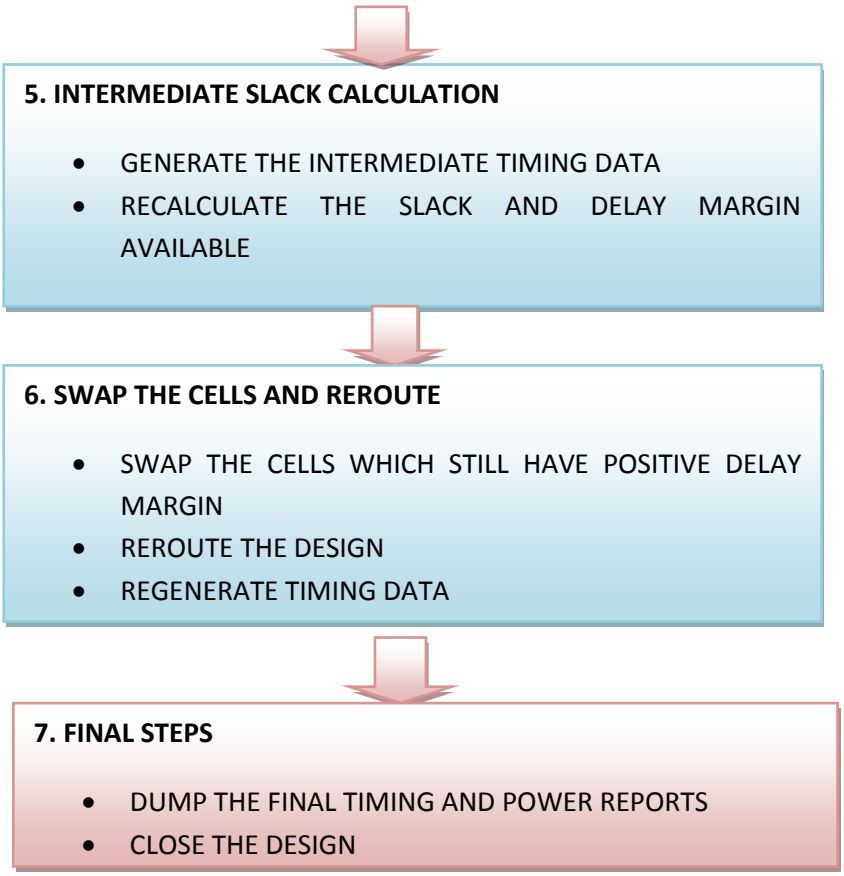
CHAPTER 5

THE ALGORITHM

This chapter shows the higher abstraction level flow along with explanation of the algorithm. The scripting language used for implementation was TCL which was used to command the Synopsys™ IC COMPILER.

5.1 THE FLOW OF THE ALGORITHM





---

1. INITIALIZATION:

After opening the design in the Synopsys™ IC Compiler shell, all the project specific variables are initialized. In this step all the timing information prior to running the core algorithm is saved for comparison. Timing reports and power reports are also generated interactively from the shell.

2. FORMATION OF COLLECTION:

In the second module of the algorithm, all the cells and their connectivity data is gathered from the design and for each cell, the slack is calculated after calculating arrival and required times.Thus we can discard cells which are violating the constraints i.e cells having negative slack or very less positive slack.

3. DELAY CALCULATION

In the third module, the buffer delay i.e the amount of delay that can be added to each cell is calculated.One another check for drive strength is made by calculating the capacitance ratio. It is the ratio of the actual capacitance the cell is driving to the maximum capacitance it can drive. Those cells having cap ratio less than 70 % are suitable candidates for downsizing as the cell is already driving less load than its capacity.

4. DOWNSIZING ITERATIVELY

In the next module , a separate procedure for cell sizing is called.This procedure takes input as the cell instance and returns the corresponding next downsize cell.

The delay characteristics are checked my the core algorithm and the current cell instance link is changed in the library if the delay is within available range.

5. INTERMEDIATE SLACK CALCULATION

At this stage the slack and delay margin are again recalculated. After this update there still might be cells where delay can be added.

6. SWAP THE CELLS AND REROUTE

The cells where still more delays can be added are swapped with corresponding lower leakage cell from the library in a similar manner the cells were downsized.
The design is rerouted as routing cleanup is required after downsizing the cells because the size of actual pin locations on cells will also change .
Timing data is generated again and checked for violations. Any critical path violations should not occur.

7. FINAL STEPS

After the core algorithm finishes , timing and power reports are dumped for comparison and the design is closed.

The sizing algorithm, the swapping algorithm and the slack calculation algorithm are written as separate procedures in tcl and called by the core algorithm as per the flow.

CHAPTER 6

RESULTS

The Results or the format of the outcome is the efficacy of the algorithm in achieving downsizing and swapping of cells in a careful manner so as not to violate the original timing

6.1 RESULTS

Thus efficacy of the algorithm is measured in two ways:

1. **The Power Savings (in %) :**

The ulterior motive of the algorithm is to have reduction in power which is maximum achievable at gate level. Leakage power and Dynamic Power have been captured for each block after running the algorithm.

2. **Bucketizing the Power Savings:**

Bucketizing is a good method to keep a check on the correct functionality of the algorithm as well to make the algorithm more flexible.

In this method the positive path constraints given as input to the algorithm are varied and Power savings are captured in specific buckets for ex. Bucket of 0 ps means that all the cells whose slack is greater than 0 ps are considered by the algorithm for downsizing and swapping.The power savings should me more in this bucket.The data for buckets of 0 , 10 , 20 , and 30 ps are captured.

The difference in the original timing after performing Static Timing Analysis and the timing at all major points and after completion of algorithm is compared to make sure the algorithm is utilizing only the positive slacks keeping the Critical Path Slack essentially same.

The power savings are plotted against different slack margin buckets to know the trend the algorithm follows in power savings .The graphs below show clearly that power savings are proportional to the number of cells available in design i.e. more the cells in a given bucket of positive slack margin, more opportunity the algorithm will get to downsize and swap and hence more power savings.

The table 6.1 below shows the power savings for 5 test cases taken for experimentation.The highest power savings achieved was approximately 6.12 % in leakage power and 1.16 % in dynamic power.

BLOCK	MARGIN	DOWNSIZING MODE		DOWNSIZING and SWAPPING MODE	
	(ps)	Power Savings (%)		Power Savings (%)	
		Leakage	Dynamic	Leakage	Dynamic
A					
	0	5.99	1.1	6.12	1.16
	10	2.8	0.83	3.79	0.91
	20	2.16	0.71	2.86	0.78
	30	1.77	0.6	2.37	0.66
B					
	0	3.67	0.9	3.86	0.99
	10	3.58	0.89	3.8	0.98
	20	3.57	0.87	3.69	0.97
	30	3.51	0.86	3.63	0.95
C					
	0	2.6	0.6	3.29	0.75
	10	2.33	0.5	2.82	0.65
	20	1.72	0.42	2.25	0.51
	30	1.28	0.32	1.72	0.5
D					
	0	2.41	0.71	3.21	0.92
	10	1.55	0.44	1.98	0.57
	20	1.12	0.33	1.45	0.78
	30	0.79	0.25	1.08	0.31
E					
	0	2.07	0.46	2.51	0.61
	10	1.52	0.37	1.89	0.47
	20	1.15	0.31	1.67	0.36
	30	0.9	0.3	1.05	0.32

Table 6.1 : Experimental data of Power Savings as a function of positive slack margin

Also the bucket of 0 ps gives the highest power savings in all test cases as the number of cells identified as candidates of swapping and downsizing will be maximum with this bucket and thus the functionality of the algorithm is also verified.

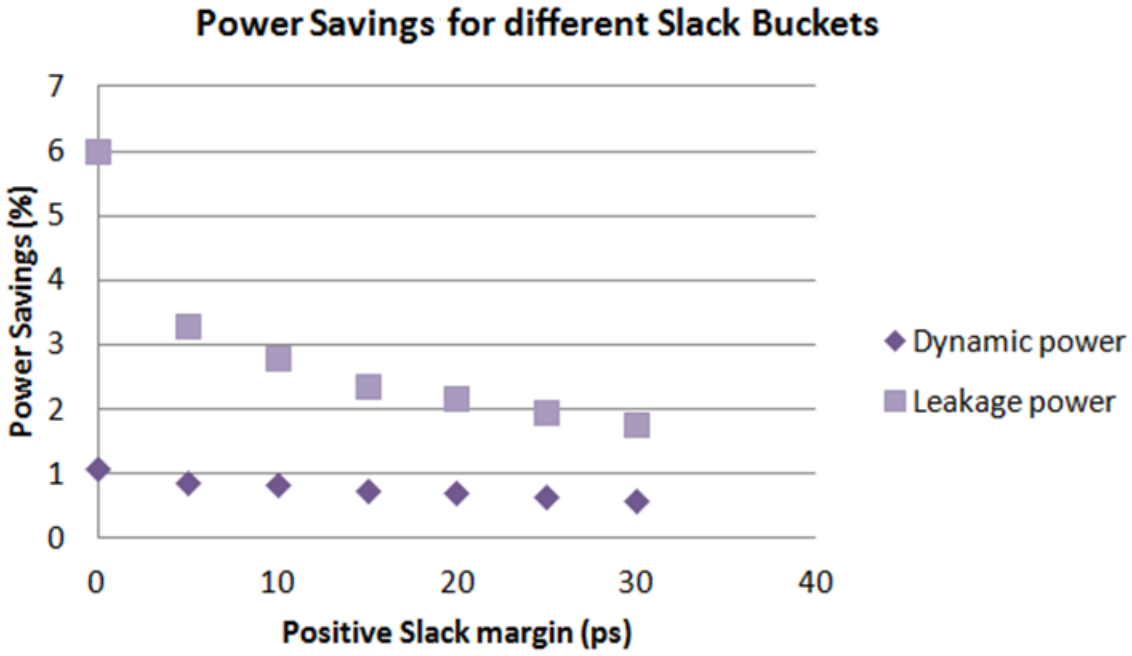
Graphical representation of behavior of all test cases is also captured in two modes

- ➔ Downsize Mode (D)- In this mode only the sizing down of gates take place.
- ➔ Downsize and Swapping Mode (DS)- In this mode swapping is an enhancement after downsizing so the results are improved.

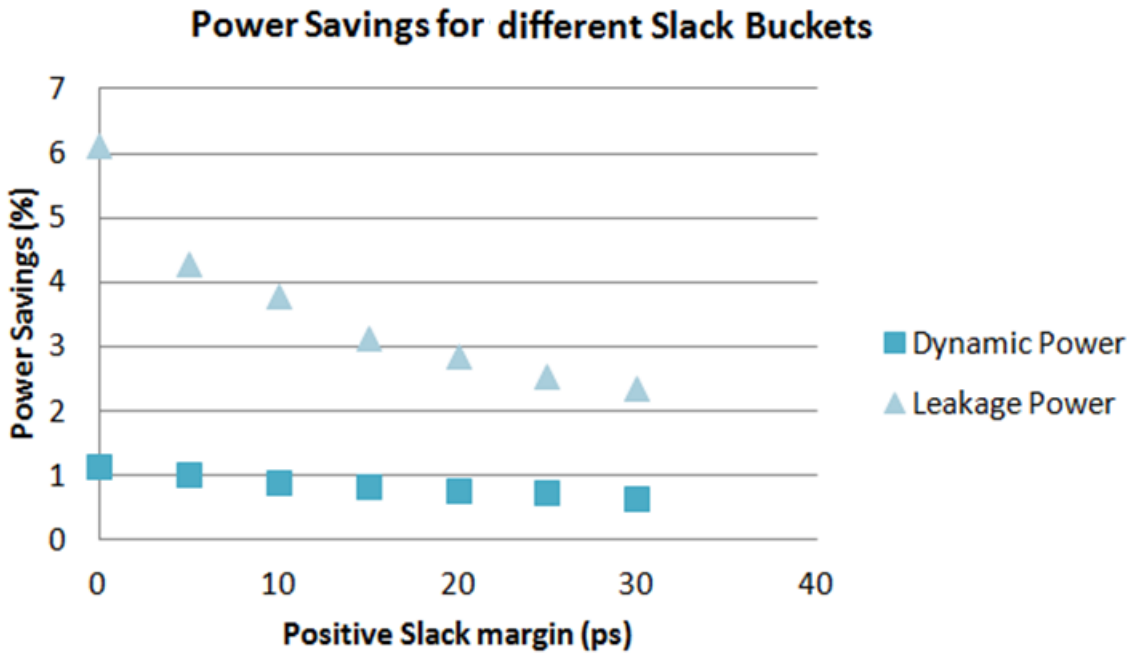
6.2 GRAPHICAL ANALYSIS OF RESULTS

BLOCK A-

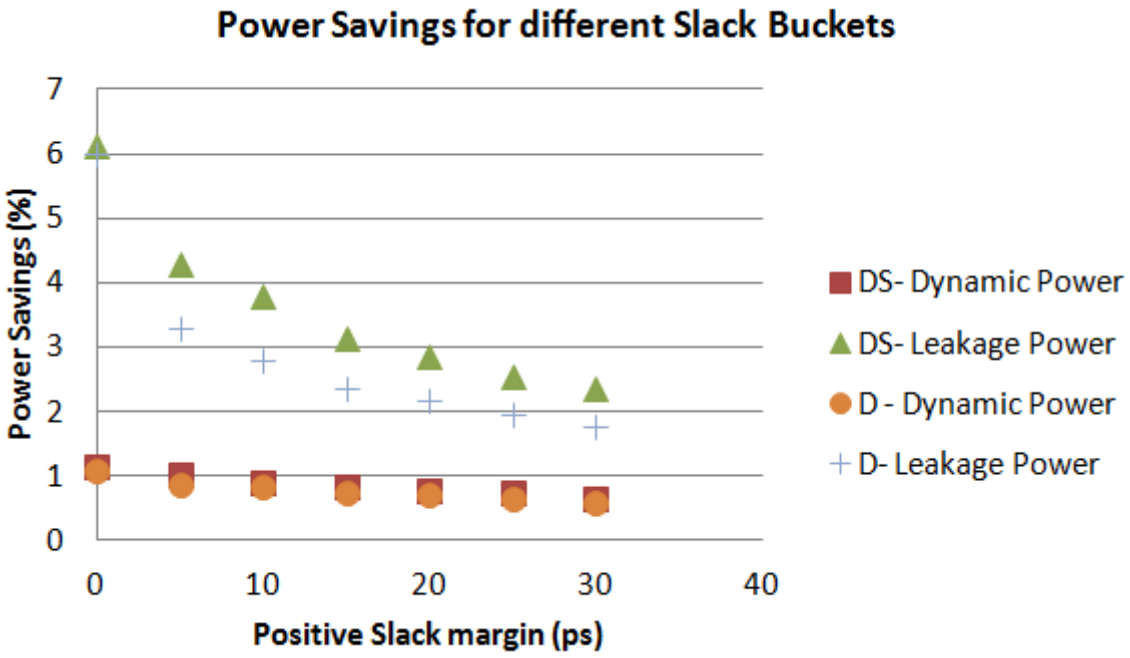
-GRAPH FOR DOWNSIZING



-GRAPH FOR DOWNSIZING AND SWAPPING

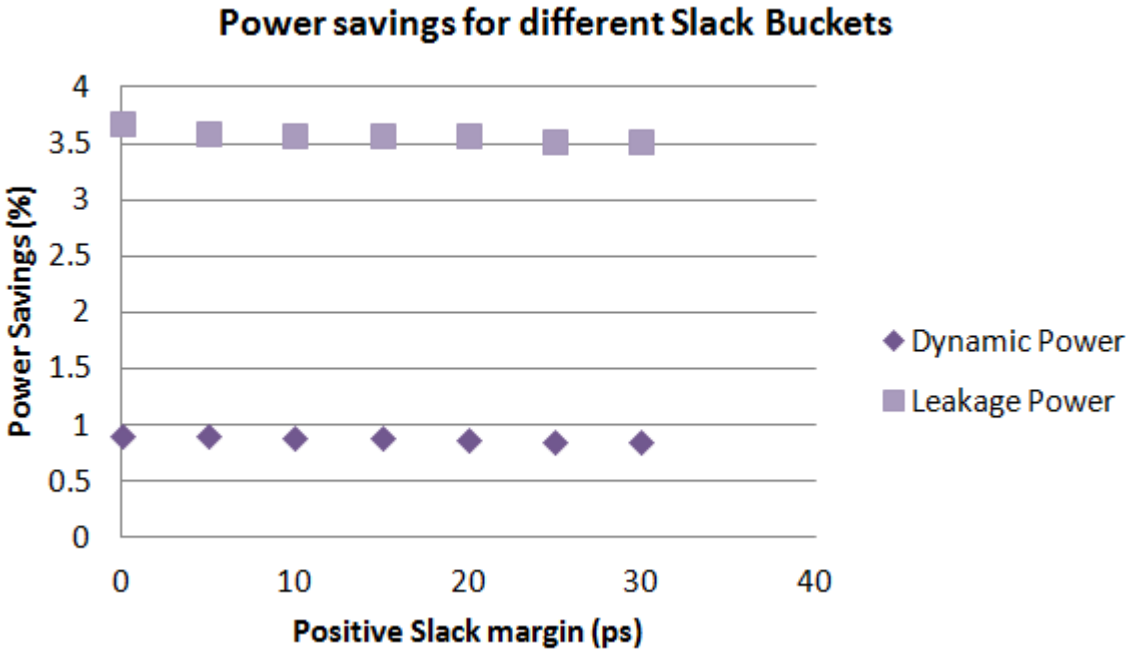


-GRAPHS WITH OVERLAPPING RESULTS OF DOWNSIZING AND DOWNSIZING AND SWAPPING

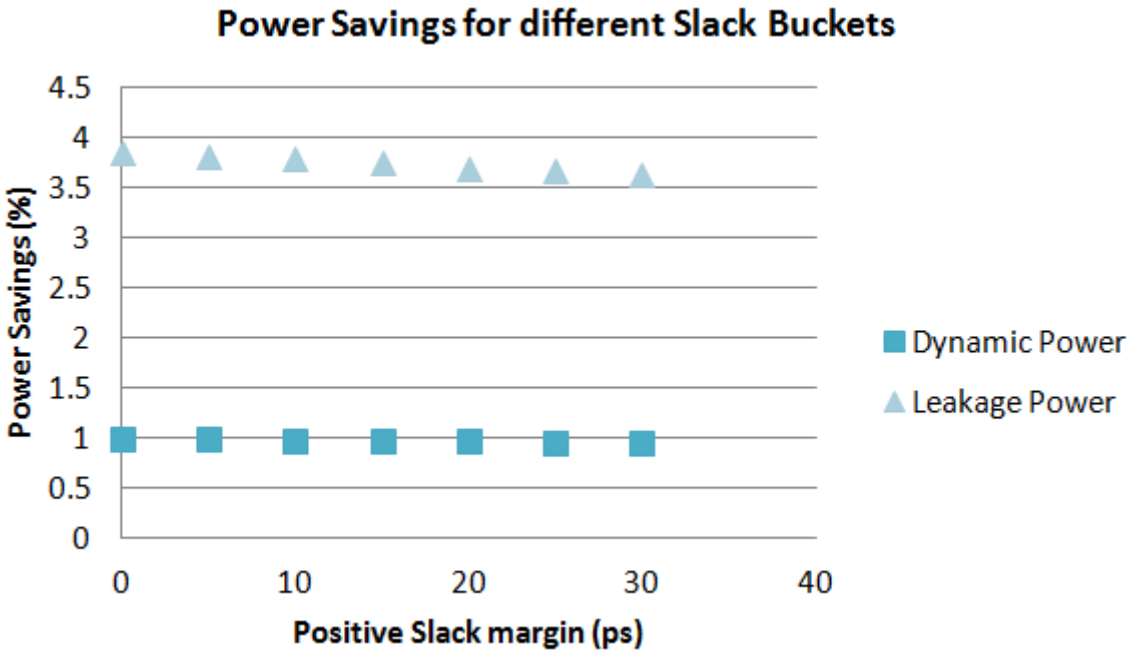


BLOCK B-

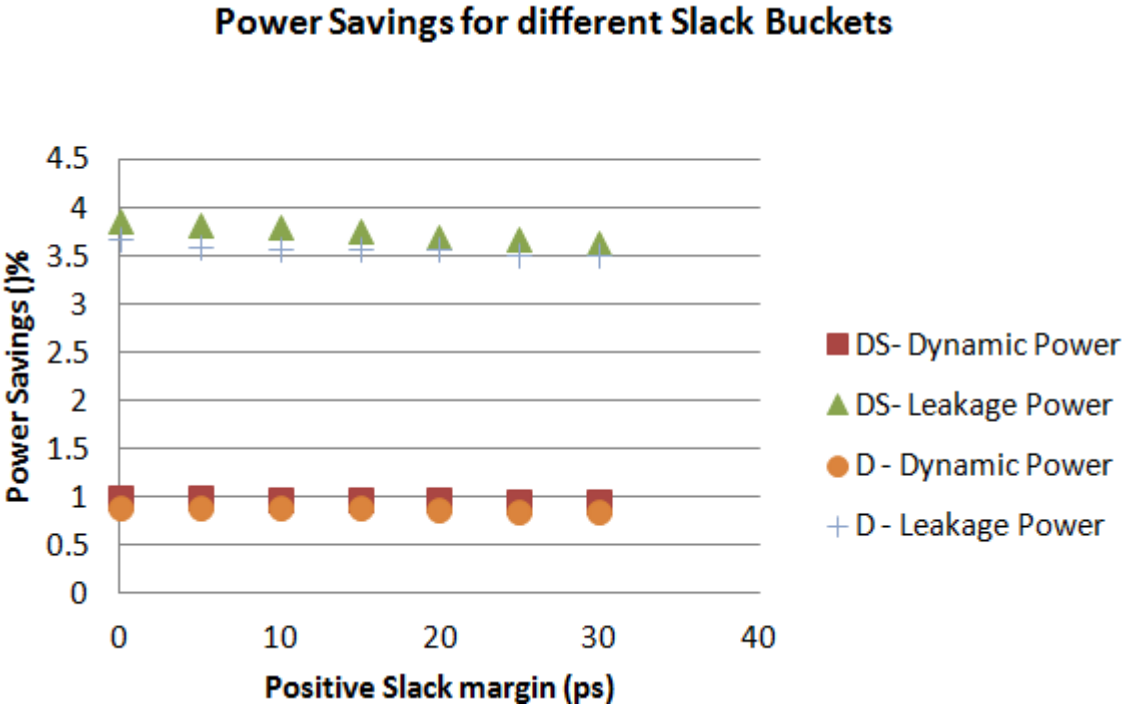
-GRAPH FOR DOWNSIZING



-GRAPH FOR DOWNSIZING AND SWAPPING

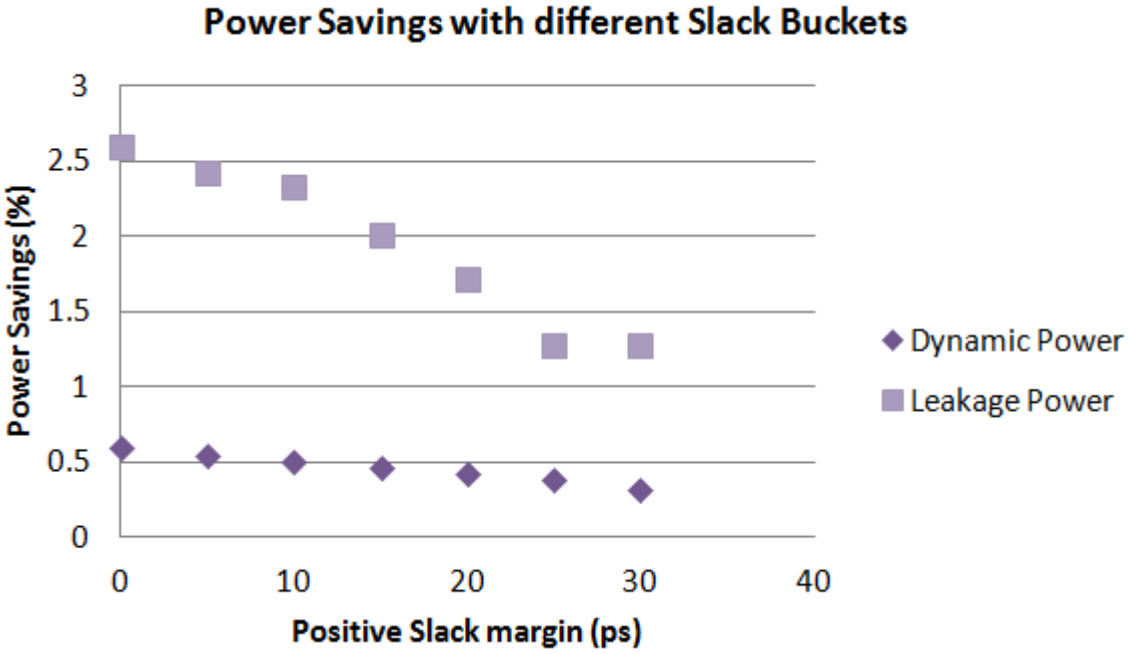


-GRAPHS WITH OVERLAPPING RESULTS OF DOWNSIZING AND DOWNSIZING AND SWAPPING

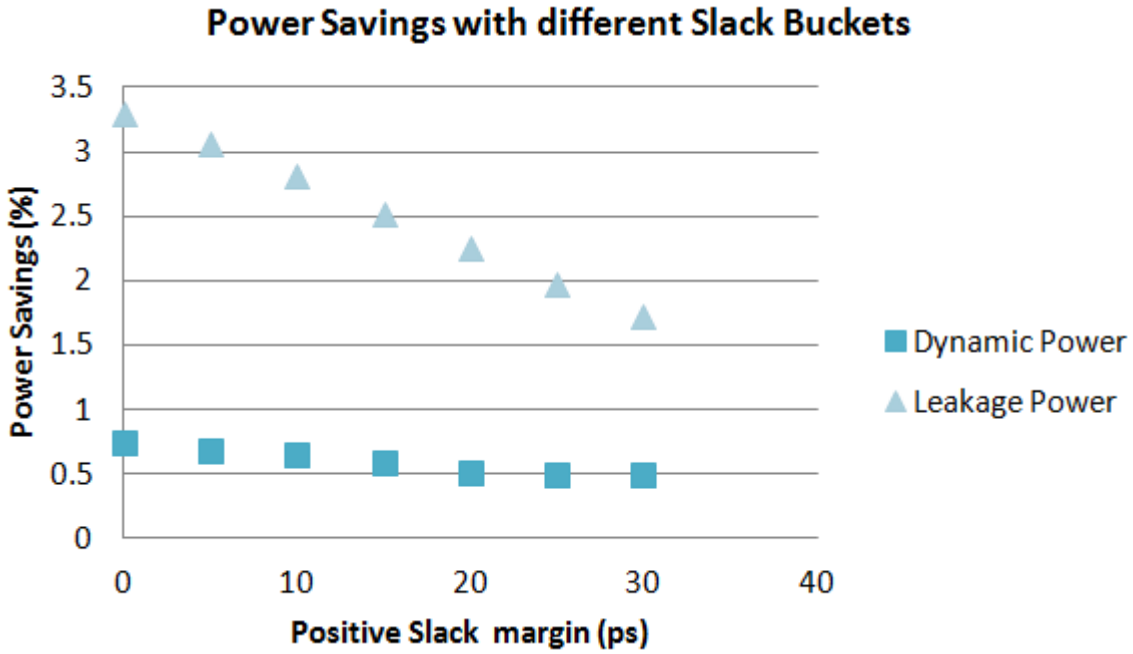


BLOCK C-

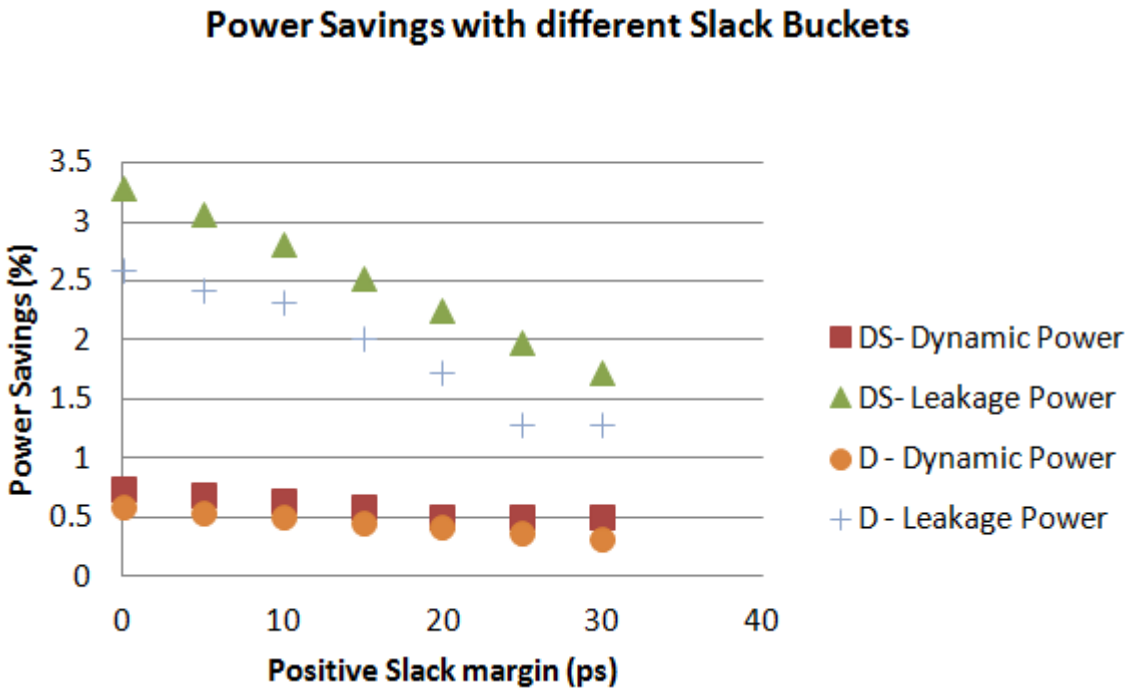
-GRAPH FOR DOWNSIZING



-GRAPH FOR DOWNSIZING AND SWAPPING

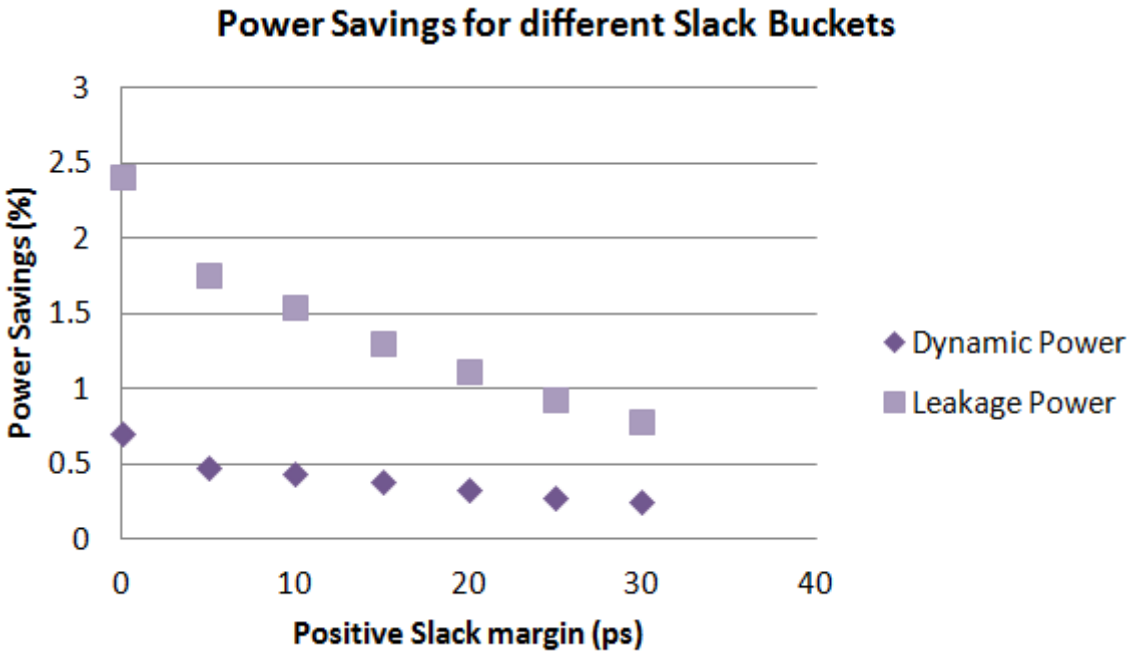


-GRAPHS WITH OVERLAPPING RESULTS OF DOWNSIZING AND DOWNSIZING AND SWAPPING

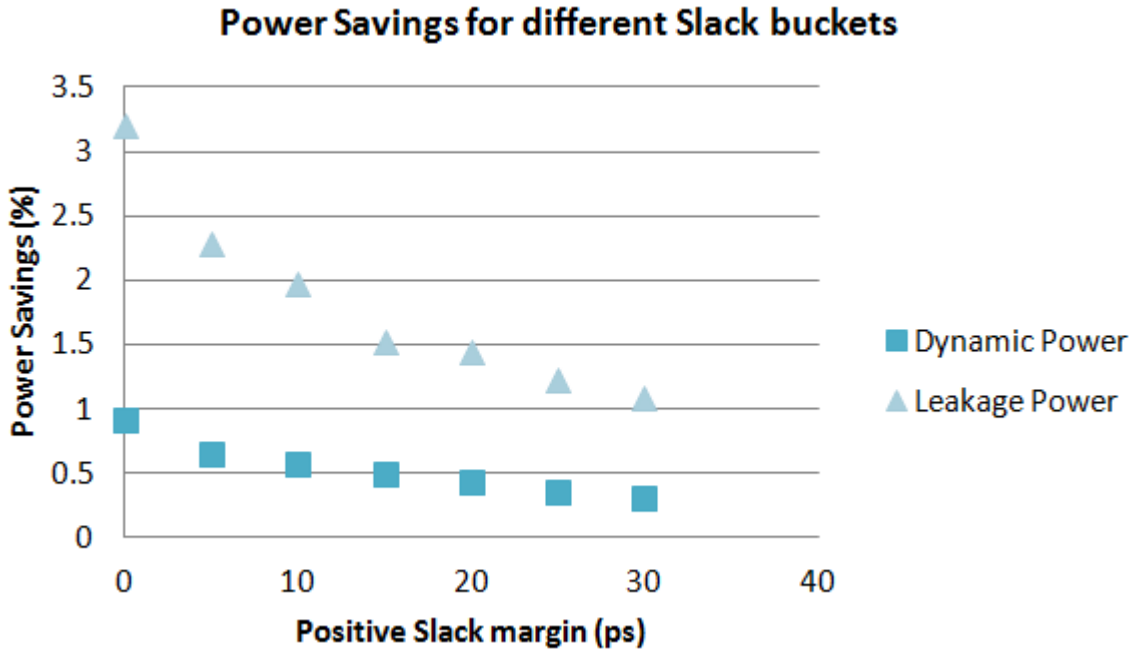


BLOCK D-

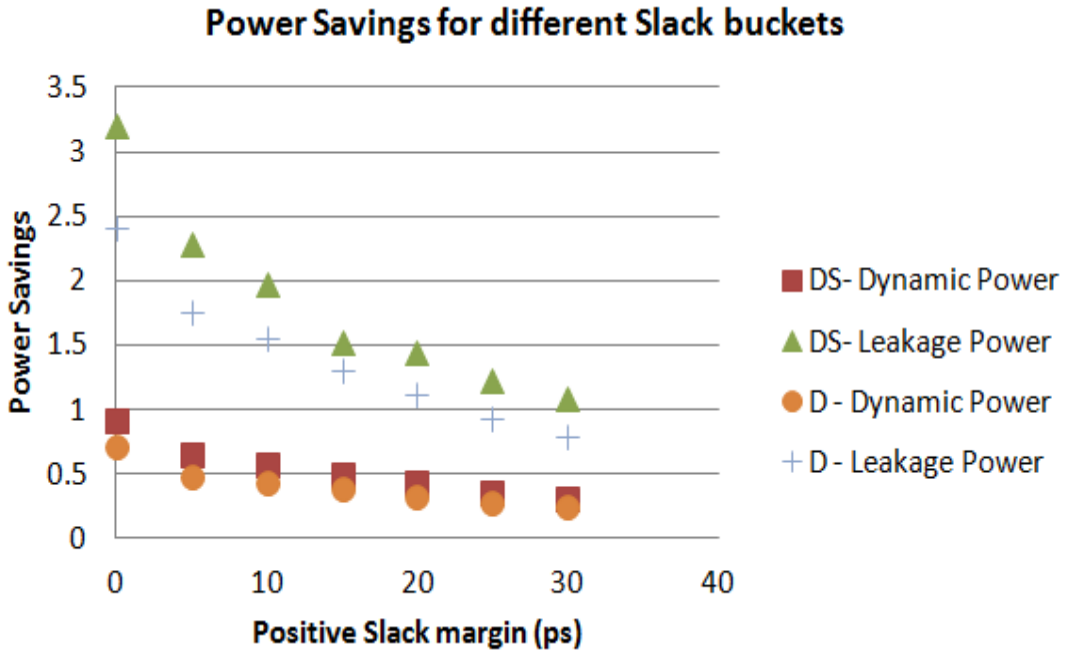
-GRAPH FOR DOWNSIZING



-GRAPH FOR DOWNSIZING AND SWAPPING

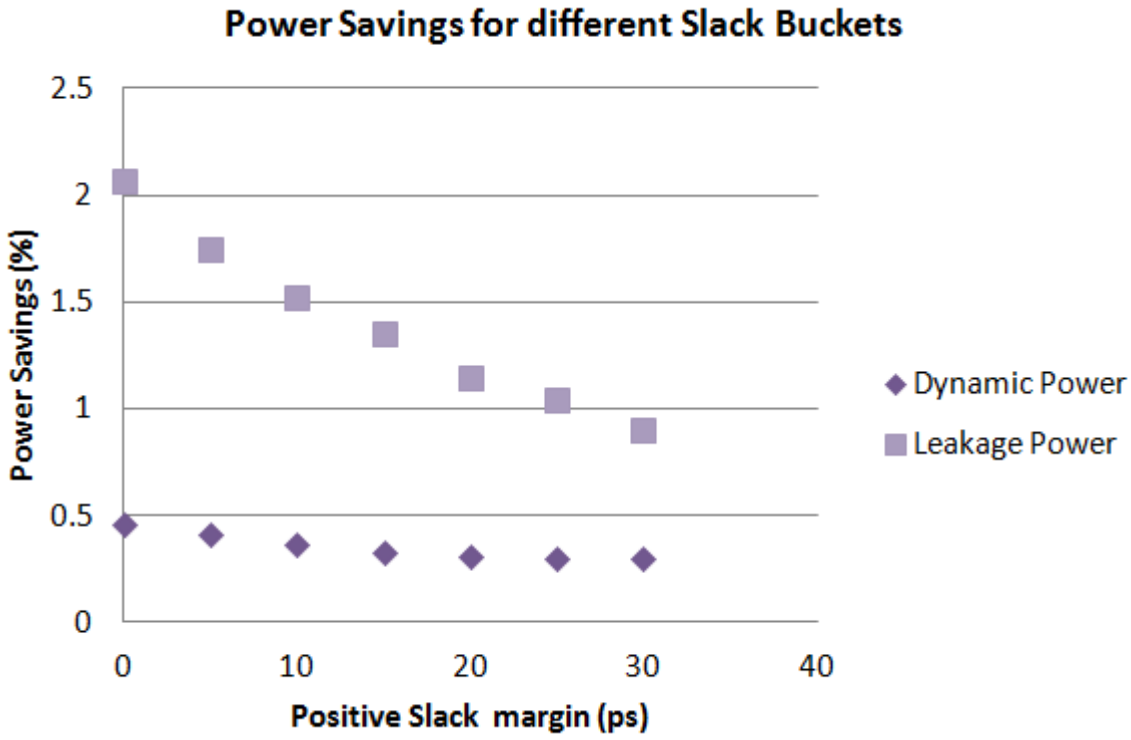


-GRAPHS WITH OVERLAPPING RESULTS OF DOWNSIZING AND DOWNSIZING AND SWAPPING

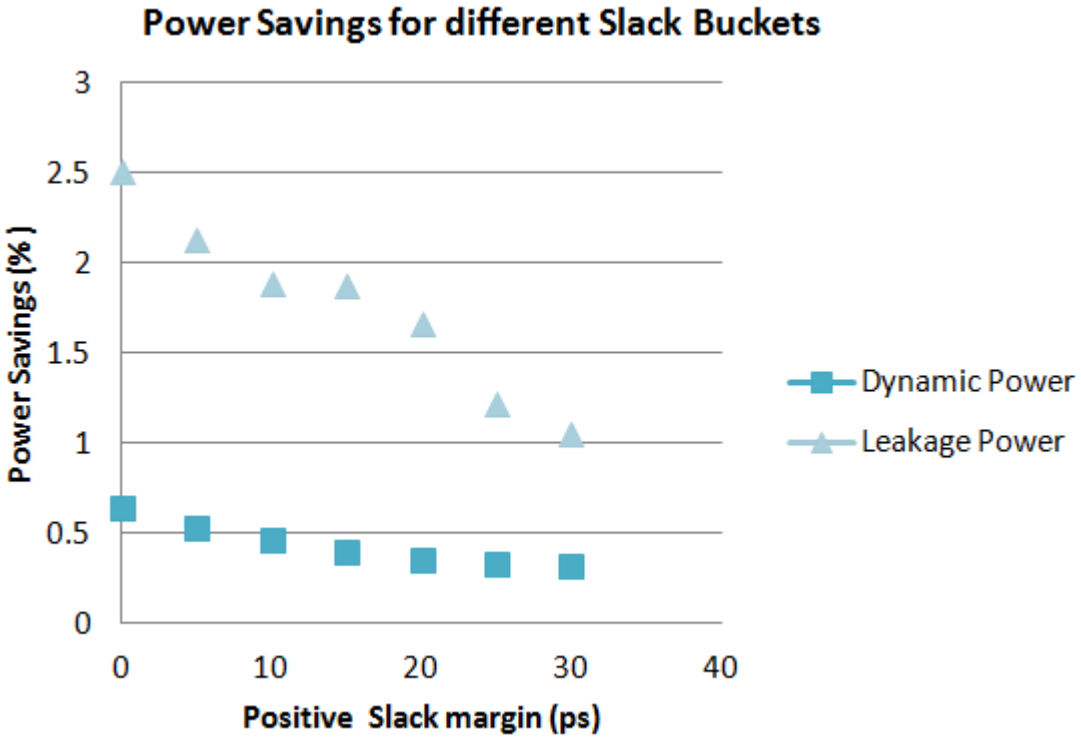


BLOCK E-

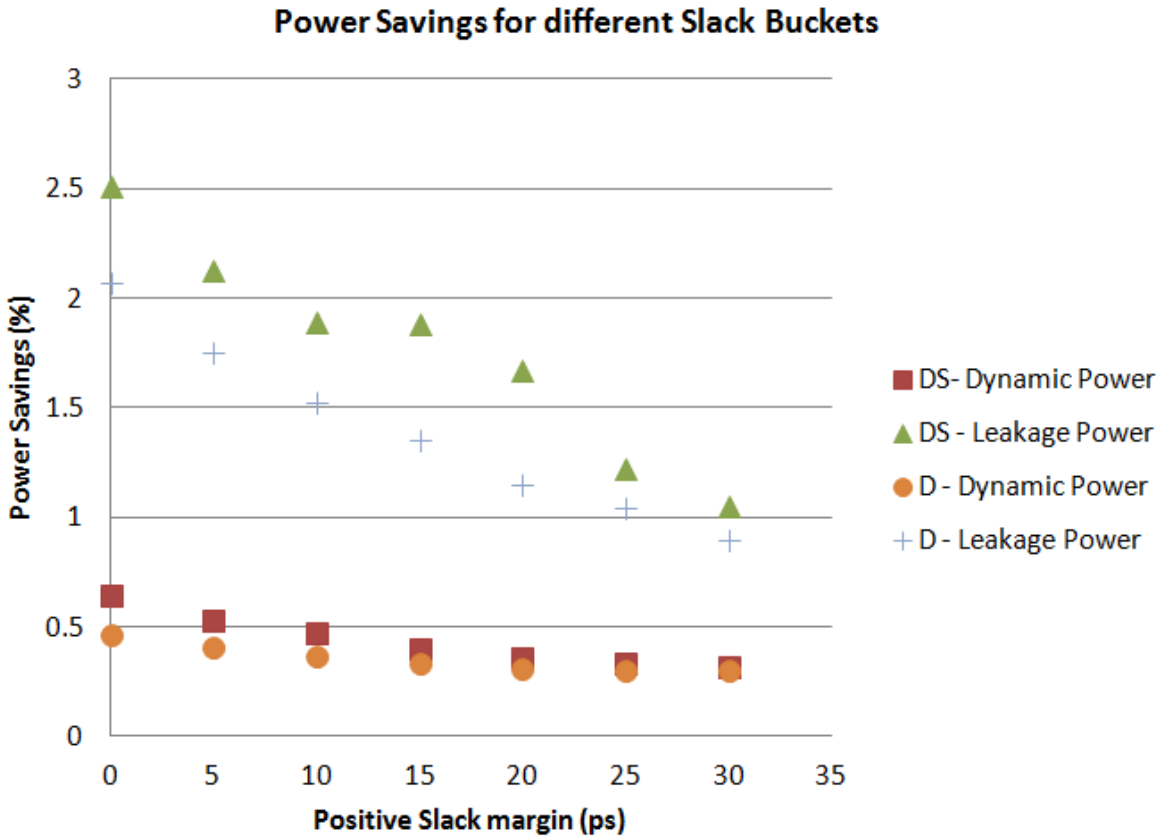
-GRAPH FOR DOWNSIZING



-GRAPH FOR DOWNSIZING AND SWAPPING



-GRAPHS WITH OVERLAPPING RESULTS OF DOWNSIZING AND DOWNSIZING AND SWAPPING



The graphical representation of the experimental results thus capture the functionality and behavior of the algorithm in entirety. The algorithm was run after routing and the design was than rerouted to get the timing information.The timing was not violated in any case.

CHAPTER 7

CONCLUSION AND FUTURE DIRECTION

7.1 CONCLUSION

The algorithm developed and tested has given considerable power savings of maximum 6.12 % leakage power and 1.16 % in dynamic power without affecting critical path timing. These savings are significant for a High Frequency Microprocessor as the one block for test case is representative of 140 similar blocks in the section.

Thus the functionality and the efficiency of the algorithm is verified through the experimental results and is fit for use in the flow development in the industry.

7.2 FUTURE DIRECTION

In future, the formulation can include more and more timing critical parameters like effect of noise and crosstalk on the signal interconnect line to obtain more accurate results. . Also, the sequential cells can be considered for downsizing by including its setup and hold time constraints in the linear programming formulation to obtain even more power reduction when compared to the case where only combinational cells are considered for downsizing, which is the case proposed in this project.

Also the use of Linear Optimization as a complex microprocessor design contain millions of cells , a Linear programming approach can be embedded within the algorithm as a solution to increased run time for large blocks.

REFERENCES

1 H. R. Lin and T. Hwang, "Power reduction by gate sizing with path oriented slack calculation", in Proc. ASP-DAC'95, pp.7-12, Aug. 1995.

2 C. Chen, X. Yang, and M. Sarrafzadeh, "Potential slack: an effective etric of combinational circuit performance", in Proc. ICCAD'OO, pp. 198-201, Nov. 2000.

3 M. Sarrafzadeh, D.A. Knol, G.E. Tellez, "A delay budgeting algorithm ensuring maximum flexibility in placement", IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 16, No. 11, pp. 1332-1341, Nov. 1997

4 H. Youssef and E. Shragowitz, "Timing constraints for correct performance," *Proc. of ICCAD, pp. 24-27, 1990.*

5 Swartz and C. Sechen, "Timing Driven Placement for Large Standard Cell Circuits," in *Proceedings ofDAC,* 1995

6 L. A. Wolsey. *Integer Programming.* New York, NY, Wiley-Interscience Publisher, John Willey & Sons Inc., pg. 37-52, 1998

7 D. A. Papa, T. Luo, M. D. Moffitt, C. N. Sze, Z. Li, G.-J. Nam, C. J. Alpert and I. L. Markov, "RUMBLE: An Incremental, Timing-Driven, Physical-Synthesis Optimization Algorithm", *IEEE Trans. on CAD*

8 Feng. Gao and J. P. Hayes, "Total power reduction in CMOS circuits via gate sizing and multiple threshold voltages", in Proc. DAC05, Anaheim, California, pp. 31-36, June 2005.27(12) (2008), pp. 2156-2168.