Accuracy and Runtime Improvement techniques for power characterization of compiled memory arrays for SoC platforms

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Tanvi Upadhyaya
(14MECV29)

Electronics & Communication Engineering
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2016

Accuracy and Runtime Improvement techniques for power characterization of compiled memory arrays for SoC platforms

Major Project Report

Phase - II

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Tanvi Upadhyaya
(14MECV29)

Under the guidance of

External Project Guide:

Mr. Murali Sundaram
Engineering Manager,
Intel India Technology Pvt. Ltd.,
Bangalore.

Internal Project Guide:

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

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

Dec 2015

Certificate

This is to certify that the Major Project entitled **"Accuracy and Runtime Improvement techniques for power characterization of compiled memory arrays for SoC platforms"** submitted by **Tanvi Upadhyaya (14MECV29)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, 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 or diploma.

Date: Place: Ahmedabad

Internal Guide

Dr. N. M. Devashrayee
(PG Coordinator,EC)

Program Co-ordinator

Dr. N. M. Devashrayee
(Professor,EC)

Director

Dr. P. N. Tekwani
(Head of EE Dept.)
(Director, IT-NU)

Project Completion

This is to certify that the Major Project entitled **"Accuracy and Runtime Improvement techniques for power characterization of compiled memory arrays for SoC platforms"** submitted by **Tanvi Upadhyaya (14MECV29)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, Nirma University, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my 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 my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

for Intel Technology India Pvt. Ltd.,

Mr. Murali Sundaram
Engineering Manager
CTG-CMO India
Intel Technology India Pvt. Ltd.,
Bangalore.

Date: Place: Bangalore

Declaration

This is to certify that

1. The thesis comprises my original work towards the degree of Master of Technology in VLSI Design at Nirma University and has not been submitted elsewhere for a degree.

2. Due acknowledgment has been made in the text to all other material used.

- Tanvi Upadhyaya
14MECV29

Acknowledgements

I take immense pleasure in thanking my managers Mr. Murali Sundaram and Mr. Vikas Gadi. Also I want to thank Intel Technology India Private Limited, Bangalore for assigning me such project and guide me through.

I would like to express my gratitude and sincere thanks to my mentor Ms. P. Mridula at Intel Technology India Private Limited, Bangalore for their valuable guidance throughout this period. They have given me valuable advices and support for my project work which I am very lucky to benefit from.

I would like to express my gratitude and sincere thanks to our Director Dr. P. N. Tekwani, Head of Electrical Engineering Department for allowing me to undertake this thesis work and for his guidelines during the review process.

I would like to thank my Program Coordinator, Dr. N. M. Devashrayee, Professor, EC (VLSI Design), Institute of Technology, Nirma University, Ahmedabad for being a source of inspiration,giving valuable support and timely advice throughout the academic period.

I take immense pleasure to thank my team members Mr. Ramakrishna C Nallapeddi,Mr. Himanshu Patel,Mr. Raghavendra Chilukuri,Mrs. Anshuli Pandey and Mr. Gaurav S. Kabra and other colleagues in the Intel, special thanks for helping me on this path and for making project at Intel more enjoyable.

I wish to thank my classmates for their delightful company which kept me in good humor throughout the journey.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

- Tanvi Upadhyaya
14MECV29

Abstract

With the advent of increasingly complex systems, the use of traditional power estimation approaches is rendered infeasible due to extensive simulation times. Hardware accelerated power emulation techniques, performing power estimation as a by-product of functional emulation, are a promising solution to this problem. However, only little attention has been awarded so far to the problem of devising a generic methodology capable of automatically enabling the power emulation of a given system-under-test. In this paper, we propose an automated power characterization and modelling methodology for high level power emulation. Our methodology automatically extracts relevant model parameters from training set data and generates an according power model.

The number of memory instances per chip is increasing rapidly. To support the full range of process, voltage and temperature corners (PVTs) and the sensitivity to process variation, the number of data points per characterization run is growing exponentially. For dynamic power and leakage characterization, the advantages of proposed technique over the existing flow are capability to simulate bigger memory blocks,faster post-layout simulation and 30x runtime improvement when compared to existing solution with 0.8% accuracy trade off.

Contents

Certificate (Nirma) ii

Certificate (Intel) iii

Declaration iv

Acknowledgements v

Abstract vi

Abbreviations xii

1 Introduction 1

1.1 Motivation 1

1.2 Need and benefits of Memory Characterization 2

2 Literature Review 4

2.1 Basic Memory Structure 4

2.2 Memory development stages 5

2.2.1 Design Implementation 6

2.2.2 Design Verification 6

2.2.3 Design Characterization 7

3 Methods of characterization for compiled memories 9

3.1 Introduction 9

3.2 Power characterization flow 10

3.3 Overview of the energy/power estimation algorithm followed by the power characterization tool 12

3.3.1 Dynamic Energy Simulation 12

3.3.2 Leakage Power Simulation 14

3.4 Challenges faced in Memory Power Characterization 15

4 Runtime enhancements techniques in power characterization flow 17
4.1 Optimization in RC extraction methodology 17
4.1.1 By using Lumped Capacitance Extraction or Cap-only extraction:- . 19
4.1.2 StarRC Features to Accelerate Simulation:- 20
4.2 Optimization in power simulation methodology 28
4.2.1 Average case simulation with scaling fraction=1 28
4.2.2 Simulation with source on power gate 29
4.2.3 Simulation with multi core option 30
4.2.4 Configuring DC initialization algorithm 30
4.3 Optimizing simulator benchmarking through a regression suite 31

5 Experimental Results and Conclusion 34
5.1 Runtime Enhancement By using Lumped Capacitance Parasitic Extraction . 34
5.2 Runtime Enhancements By using Cap Only Extraction 34
5.3 Runtime Enhancement By Computing Simulation using ScalingFraction=1 . 35
5.4 Runtime Enhancement with source on power gate 35
5.5 By using Back Annotation (BA) Flow 35
5.6 Active net based flow 36
5.7 Multi-core options built in circuit simulator 36
5.8 DC initialization 37

Conclusion 38

References 39

List of Figures

1.1 Typical SoC model 1
1.2 Memory share in SoC 2
1.3 Total chip dynamic and static power dissipation 3

2.1 Memory Block Diagram 4
2.2 Memory Development Stages 5
2.3 Memory compiler development flow 7

3.1 Curve fitting with automatic characterization tool 10
3.2 Characterization tool methodology 11
3.3 Power characterization for input pins 12
3.4 Power characterization for input bus 13
3.5 Power characterization for output pins 14
3.6 Power Gated Designs 14

4.1 Increasing impact of parasitics on delay at advanced process nodes 18
4.2 RC line models 20
4.3 The active node flow between StarRC and CustomSim delivers push-button productivity 21
4.4 Setup Phase 22
4.5 Simulator selecting Active nets based on threshold voltage on nets 23
4.6 Simulation Phase 23
4.7 Default flow(Flat netlist based) 24
4.8 Back Annotation Flow 24
4.9 StarRC and CustomSim hierarchical back-annotation flow enables schematic netlist efficiency in post-layout simulation 25
4.10 StarRC hierarchical parasitic extraction supports top-down or bottom-up methodology 26
4.11 Context-specific device parasitics have pronounced impact on circuit performance at advanced nodes and therefore must be accurately extracted 27
4.12 Power Gated Design 30

4.13 Old methodology to benchmark two different simulator versions involving human intervention 32
4.14 Methodology followed in the regression suite to benchmark two different simulator versions 33

List of Tables

5.1 Comparative results for Distributed Vs Lumped Parasitic Extraction 34
5.2 Comparative results for Cap only Vs Distributed RC parasitic extraction . . 34
5.3 Comparative results for Scaling fraction=1 and 0.5 experiments 35
5.4 Comparative results for runs with and without source on power gated node . 35
5.5 Comparative results for flat netlist and back annotated flow 36
5.6 Comparative results for flat netlist and active net based flow 36
5.7 Comparative results for flat netlist and multi core flow 36
5.8 DC initialization 37

Abbreviations

IP Intellectual Property

SRAM Static Random Access Memory

ROM Read Only Memory

PVT Process Voltage Temperature

TAT Turnaround time

NW Number of Words

NB Number of Bits

HDL Hardware Description Language

RTL Register Transistor Logic

DRC Design Rule Check

LVS Layout Versus Schematic

LEF Library Exchange Format

ECSM Effective Current Source Model

CCS Composite Current Source

TTM Time To Market

TCAM Ternary Content Addressable Memory

AMS Analog/Mixed signal

SPF Standard Power Format

BA Back Annotation

PLX Post Layout Acceleration

Chapter 1

Introduction

This chapter describes the scope of the project and the need of optimizing runtime and accuracy in power characterization of memory compiled IPs.

1.1 Motivation

Systems-on-a-chip (SoC) designers must integrate a wide variety of intellectual property (IP). All kinds of memory, logic and control functions, with many memory sizes and types now being placed on chip.The upshot trends show that IC design, once logic dominant, has become memory dominant. In fact, it will be very common to see more than 100 different instances of SRAM, ROM and other specialty memories on a single SoC.
Figure 1.1 shows the upshot of single SoC in which memory is dominant on logic design

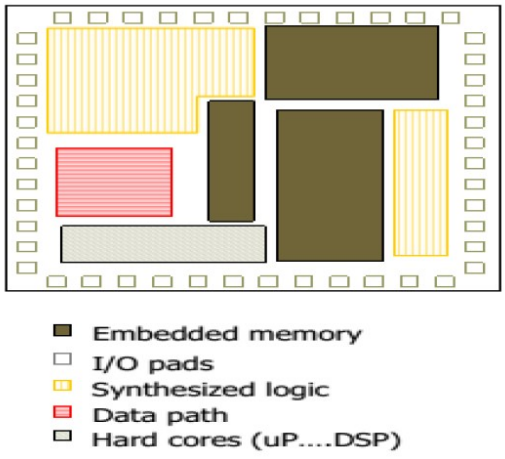

Figure 1.1: Typical SoC model

But to use these embedded memories in IC designs, accurate timing and power models are essential. And to generate such models, characterization is required. This often implies many simulations with updated Spice models.

At the same time, technology is moving rapidly from 0.25 micron to 0.18, 0.15 and 0.13 micron. Along the way, generic CMOS technology has diversified into variants such as high speed, low power and high density. Now, each process variant must have its own characterization. With high-volume products, it is increasingly common to manufacture a part at several foundries to meet customer needs. Since each process variant at each foundry is slightly different, it is necessary to recharacterize memory for the specific process and foundry to be used. That is why a tool that automatically characterizes embedded memory has become indispensable.

1.2 Need and benefits of Memory Characterization

It is known that from past two years, embedded memory requirement has grown to occupy 94% of the total area on the chip,allocating a mere 6% for reused logic or new logic designs. In System-On-a-Chip (SOC) designs, a variety of memories, logic and control functions are being integrated onto a single chip. Designers have access to a large variety of memory types and sizes. It is now common to see over 200 different configuration blocks of SRAM, Register Files, ROM and other speciality memories placed on a single SOC design.
Figure 1.2 shows an exponential growth in the embedded memory share in the overall SoC[1]

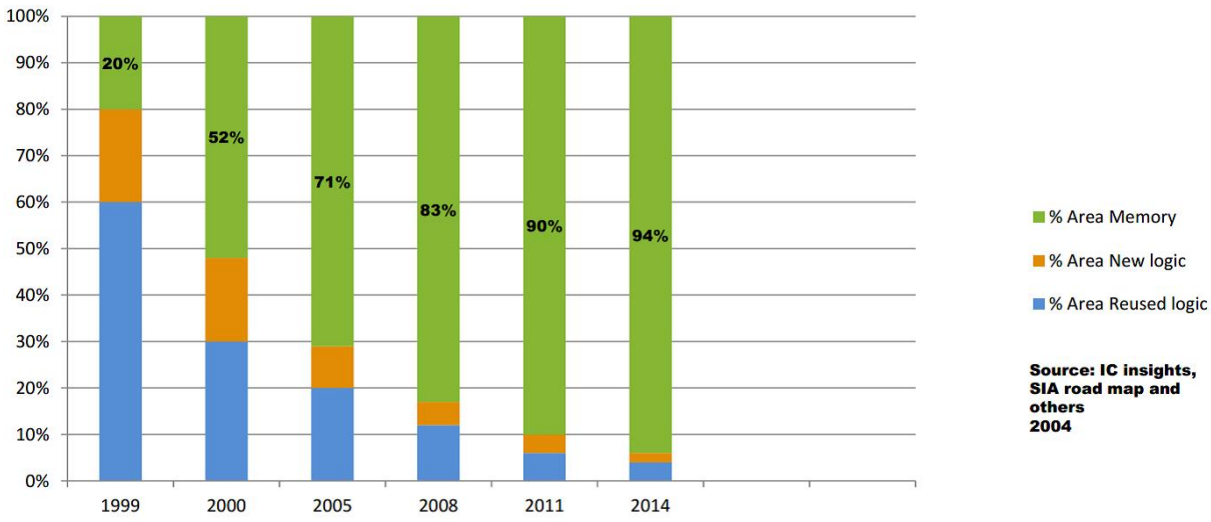

Figure 1.2: Memory share in SoC

So system designers are paying more and more attention to memory subsystem powers. In order to create power efficient designs, accurate power budgets for the memory subsystem are essential, whether it is for calculating battery life, planning cooling system, or determining the power supply.

Moreover, as a result of CMOS technology scaling, Leakage power dissipation has become a significant portion of total power consumption in deep submicron VLSI designs.
The International Technology Roadmap for semiconductors (ITRS) predicts that in a few years, the total leakage power of a chip may exceed the total dynamic power, and the projected increase in sub threshold leakage shows that it will exceed total dynamic power dissipation as technology drops below 65-nm feature size as shown in Figure 1.3.

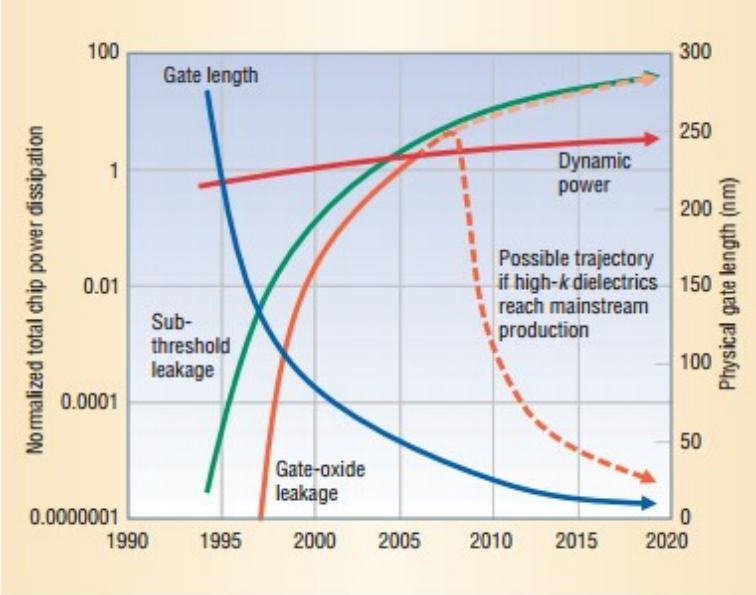

Figure 1.3: Total chip dynamic and static power dissipation

As leakage continues to increase, accurate leakage power estimation is needed to allow designers to make good design trade-offs. So there is crucial need for characterizing power for memories and the characterization should be done as fast as possible. The advantage of the implemented project are mostly as per the runtime of characterization by maintaining the accuracy as close as possible.

Memory characterization often requires hundreds of SPICE simulations. The number of memory instances per chip and the need to support a wide range of process, voltage and temperature corners (PVTs) make these simulations a daunting task. Also, the growing size of memory instances and sensitivity to process variation add more dimensions to an already challenging undertaking. Further, the need to create library variants for high-speed, low-power and high-density processes makes it imperative to automate the memory characterization flow.

In this project, we quantitatively show that it is possible to increase power characterization turnaround time (TAT) without affecting the accuracy when compared with traditional spice methods. This method is extremely important when it comes to bigger sizes of the memory array.

Chapter 2

Literature Review

2.1 Basic Memory Structure

Figure 2.1 shows the memory block structure with all inputs and outputs.

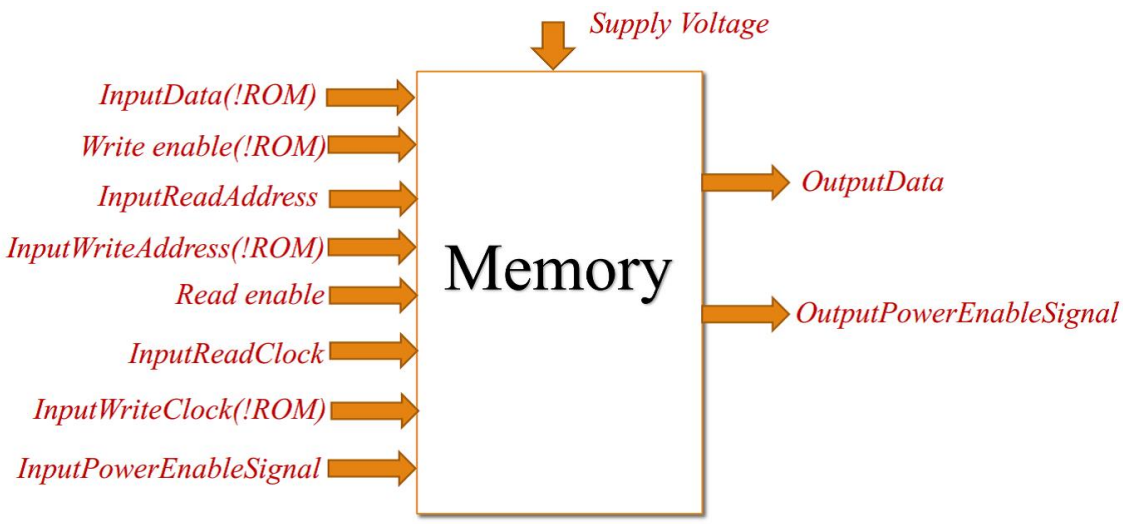

Figure 2.1: Memory Block Diagram

- **InputData:** The size of input data bus depends on the number of bits of the memory. E.g. For 256x4 memory, the size of input data bus is 4 bits i.e. InputData [0:3]
- **OutputData:** The size of output data bus depends on the number of bits of the memory.
  E.g. For 256x4 memory, the size of output data bus is 4 bits i.e. OutputData [0:3]
- **InputReadAddress / InputWriteAddress(!ROM):** The size of input address bus depends on the number of words of the memory.

E.g. For 256x4 memory, the size of input address bus is 8bits log2 (NW) i.e.InputRead/ WriteAddress [0:7]

- **ReadEnable / WriteEnable (!ROM):** These signals are used to enable read or write operation of the memory.
- **InputReadClock / InputWriteClock (!ROM):** These signals are the clock signals for read or write operations of memory.
- **InputPowerEnableSignal:** This signal is used for the power gated memory designs.
- **OutputPowerEnableSignal:** This signal is used as the acknowledgement for the SoC Team telling that all power gated transistors are enabled

2.2 Memory development stages

There are three phases in a memory development flow:

1. Design Implementation
2. Design Verification
3. Design Characterization

Figure 2.2 determined the broad view of these development phases.

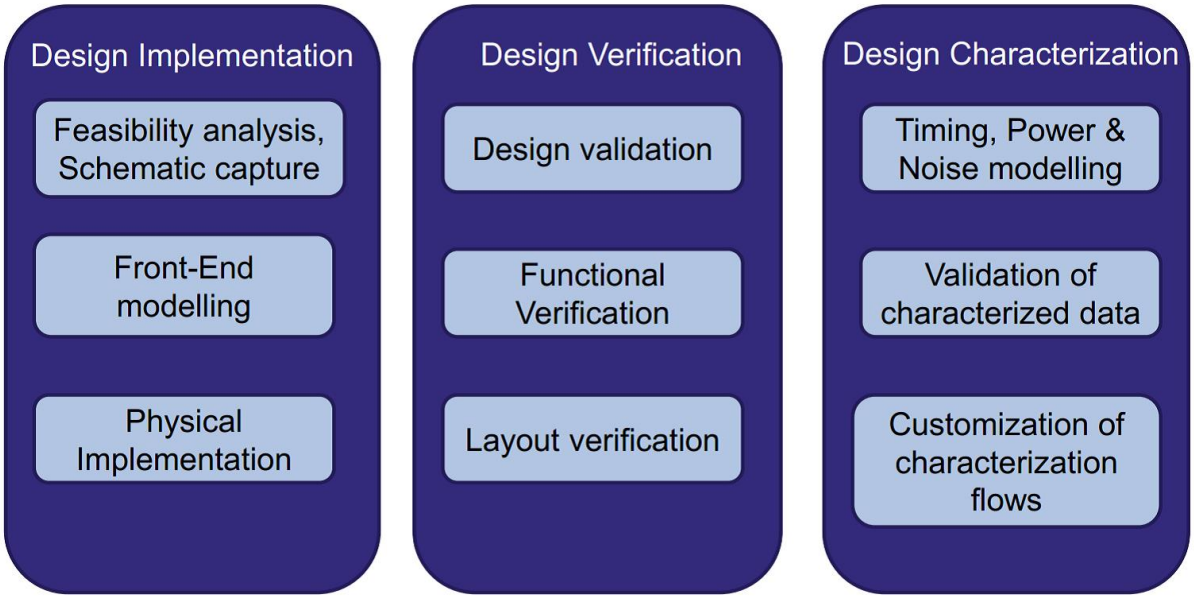

Figure 2.2: Memory Development Stages

2.2.1 Design Implementation

1. **Schematic Capture:** Schematic capture or schematic entry is a step in the design cycle of memories at which the schematic of the designed circuit is created by a designer. This is done with the help of a schematic capture tool also known as schematic editors. Normally the design constraints or specifications are known to the designers in this step. i.e. All the design constraints are freezed in this step.
2. **Front End Modelling:** Front end design includes digital design using HDL's, the design from gates and design for testability.
3. **Physical Implementation:** The conversion of the netlist into its geometrical representation is done in this step and the result is called a layout. This step follows some predefined fixed rules like the lambda rules which provide the exact details of the size, ratio and spacing between components. This step is further divided into sub-steps which are circuit partitioning,floor planning,routing,layout compaction and extraction and verification.

2.2.2 Design Verification

1. **Design Validation:** Design validation is the process of ensuring that the final product conforms to user specifications and requirements. It represents the final stage in the process of ensuring that new designs are fit for the intended purpose before release of the item or system for use. Validation of design outputs may be accomplished by a range of different methods dependant on the nature of the system or item covered by the design, and a combination of methods will generally be applicable to any given item.
2. **Functional Verification:** Formal verification is a technique used in different stages in design life cycle like front end verification, Logic Synthesis, Post Routing Checks. Formal verification step verifies the functional equivalence of two designs that are at the same or different abstraction levels (e.g., RTL-to-RTL, RTL-to-Gate, or Gate-to-Gate).
3. **Layout Verification:** After routing, your physical verification tool should give you zero DRC/LVS violations. However, the physical verification tool deals with abstracts like FRAM or LEF views. We use dedicated physical verification tools for signoff LVS and DRC checks.

2.2.3 Design Characterization

Memory characterization is a very time-consuming and error-prone process. Due to the complexities and varieties of the timing parameters involved, it has become a key part in the memory compiler development flow as shown in Figure 2.3.

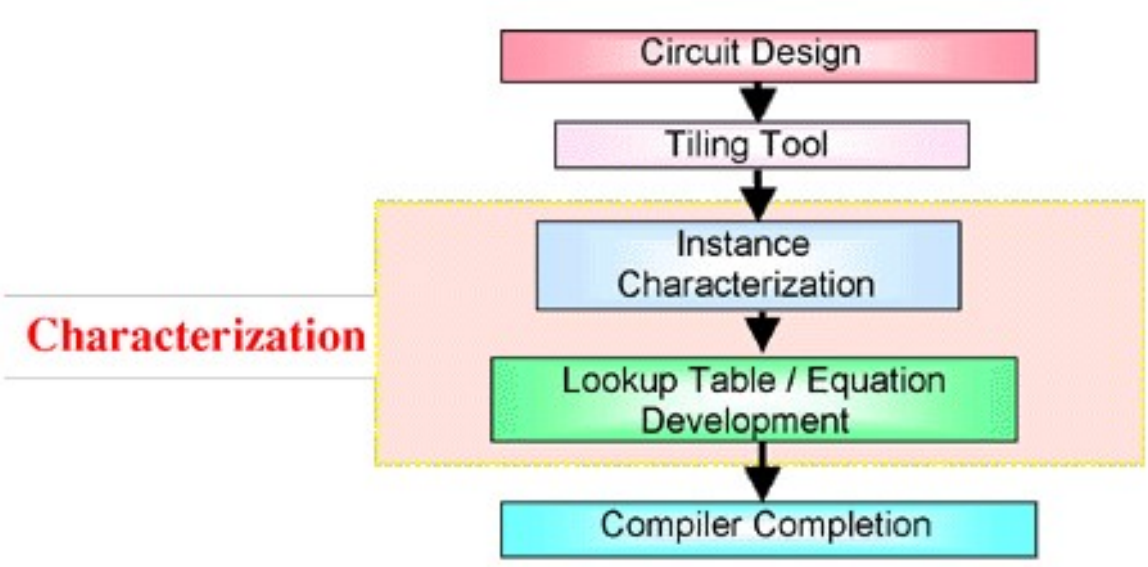

Figure 2.3: Memory compiler development flow

Typically, the characterization process builds timing models, which can be applied to all memory instances in the compiler. Variations on memory instances could be, among others, word count, bit count or aspect ratio. Memory is usually characterized in the following steps:

1. Select corner instances. Typical selections are based upon criteria such as small, large, narrow and wide.
2. Characterize the selected instances.
3. Build the lookup table or equations by fitting curves.
4. Generate timing model.

The conventional way of characterizing memory is a two-step method. First, manually build a parameterized composite Spice netlist of the tilable blocks. The major memory blocks, such as column, row and memory array are parameterized as functions of number of words, bits per word and multiplexing. Second, iteratively run simulations over the netlist and obtain timing information over the range of parameters.

The conventional method separately builds the netlists of major blocks for simulation instead of treating the memory instance as an entity from layout extraction. To

extend the model to different instance variations, compiler designers need to develop a parameterized method for timing estimation, which is sometimes called a black box model. Moreover, the conventional method is always based on the following simplified assumptions:

1. A memory can be segmented as the timing of a memory instance is just the summation of the timing of its major blocks. This assumption can lead to ignoring coupling and distributed effects, which become more pronounced for more advanced technology.
2. A memory can be parameterized–the Spice netlist of major blocks can be made as a function of words, bits per word and multiplexing to accurately reflect the actual layout over a broad range of multidimensional variations. The actual memory circuit is often nonlinear, and such things as special power and spacer cells add to this nonlinearity. If a mistake has been made in parameterization it will likely not be detected.

To produce high-quality models, the conventional method needs numerous correlations between hand-built critical-path netlists and the real ones extracted from the layout. For the present technology, the task of detailed correlation is getting more difficult because the function is multidimensional and multi-ordered.

Chapter 3

Methods of characterization for compiled memories

3.1 Introduction

Broadly speaking, there are two main methodologies for memory characterization. The first one is to characterize memory compiler-generated models and the second is to characterize individual memory instances. Further, there is an assortment of approaches for instance-based characterization, including dynamic simulation, transistor-level static timing analysis and ad-hoc divide and conquer.

Memory compilers construct memory instances by abutted placement of pre-designed leaf cells (e.g., bit-columns, word and bit line drivers, column decoders, multiplexers and sense amplifiers, etc.) and routing cells where direct connection is not feasible. The compiler also generates a power ring, defines power pin locations and creates various electrical views, netlists and any additional files required for downstream verification and integration.

Memory compilers do not explicitly characterize the generated cells but instead create models by fitting timing data to polynomial equations whose coefficients are derived from characterizing a small sample of memory instances. This approach enables memory compilers to generate hundreds or thousands of unique memory instances, differing in address size, data width, column/row density and performance. However, the model accuracy of this approach is poor.

To safeguard against chip failure due to inaccurate models, the memory compiler adds margins. These can lead to more timing closure iterations, increased power and larger chip area, however. In addition, the fitting approach doesn't work well for the advanced current-based models: effective current source model (ECSM) and composite current source (CCS), which are commonly used for timing, power and noise at 40 nm and below.

To overcome the inaccuracies of compiler-generated models, design teams resort to instance-specific characterization over a range of PVTs. This is a much more time-consuming

process that yields more accurate results. However, often due to limitations in the characterization approach and available resources, the accuracy improvement is not as much as it could be, while the cost is high.

3.2 Power characterization flow

The power characterization flow is a back and forth process in which the the power numbers are generated for 9 ebb points within the compiler size and curve fitting tools generates the equations for determining power numbers at all points within the compiler size for any range/size of memory.Figure 3.1 below depicts the methodology of automatic power characterization tool.

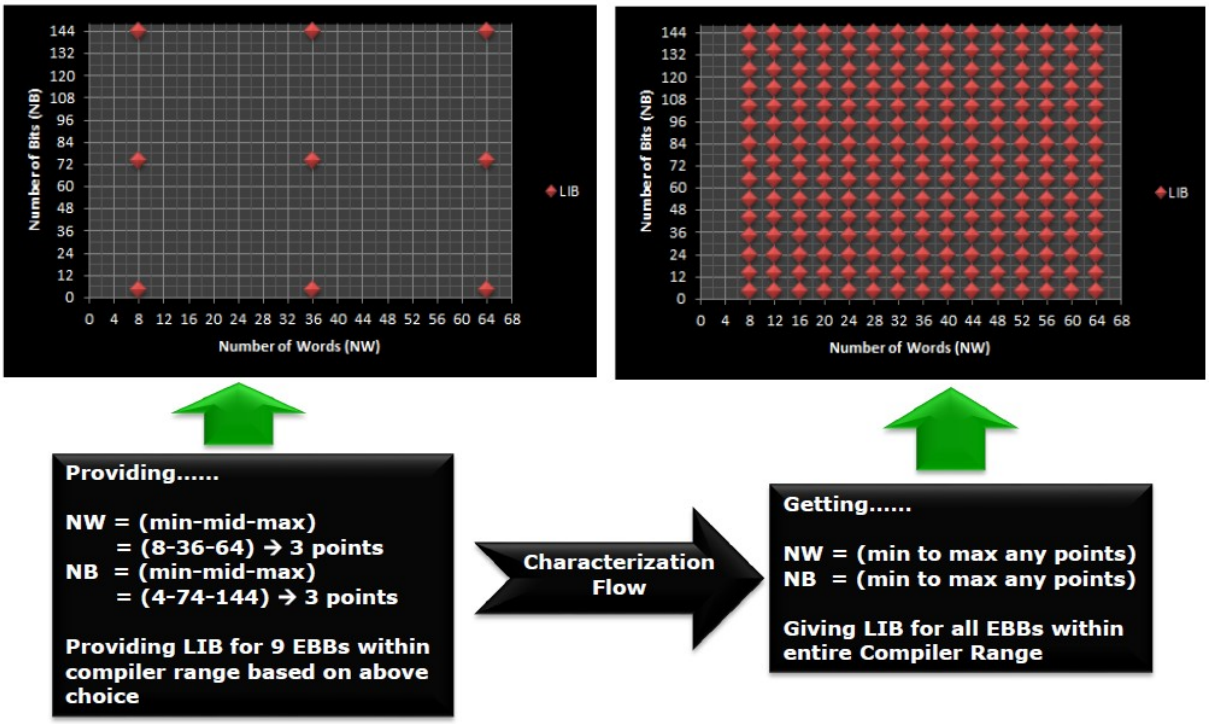

Figure 3.1: Curve fitting with automatic characterization tool

The characterization flow followed can be described by the underlying flow chart in Figure 3.2:-

Power characterization tool undergoes the following steps For each memory instance that is to be characterized at each PVT:-

1. The inputs required for power characterization are:-
    - Schematic Netlist (Spice netlist)

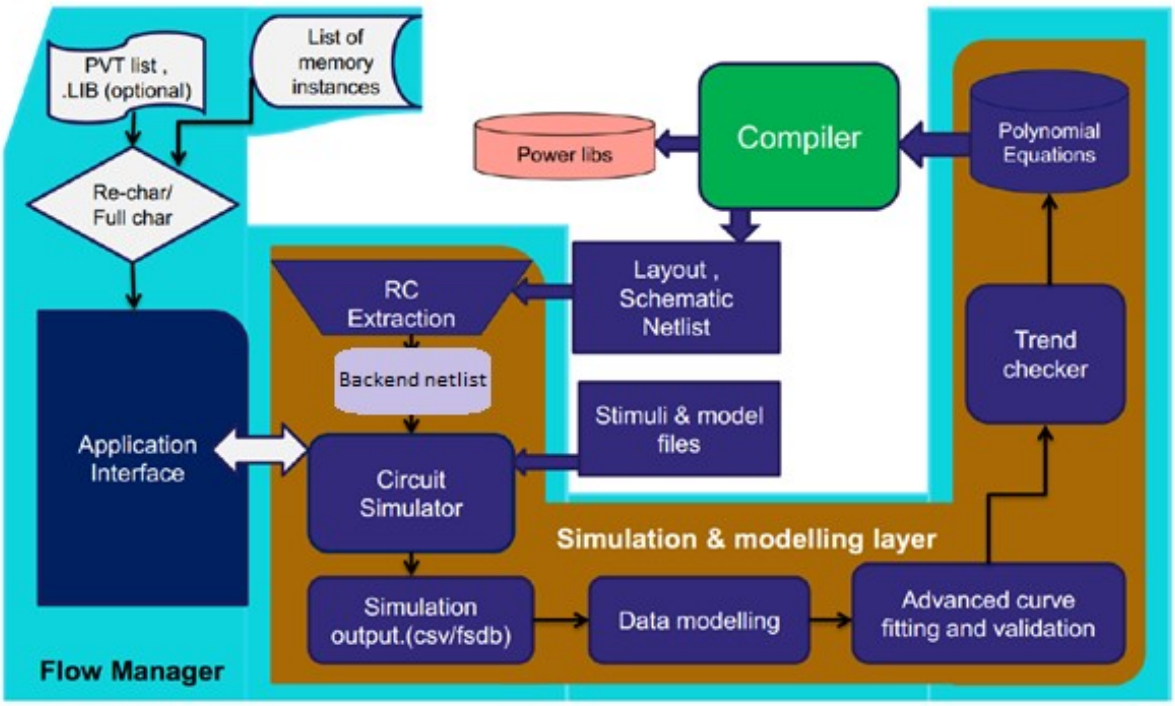

Figure 3.2: Characterization tool methodology

- Layout Netlist (gdsII format)
- List of PVT to be characterized
- List of memory instances to be characterized
- Stimuli and model filles

2. The inputs are given to the flow and schematic netlist is converted into internal tool format through a netlister tool

3. The netlist and layout inputs are given to the extraction tool which is used to extract all parasitic information of the memories. By doing RC extraction, all the parasitic information is present in two files alongwith some other files to determine the quality of the extraction data(correlation between spice netlist and extracted data).

4. The extracted data alongwith other inputs i.e. Stimuli and Model Files, List of PVT and List of memory instances to be characterized to generate all the leakage power and energy values in the memory circuit. The circuit simulator produces simulation results in the form of power calculation and energy values matrices and in the form of waveforms.

5. The files containing energy and power calculation are used to generate tables which can be given as input to the data modelling and curve fitting tool

6. The curve fitting tool generates polynomial equations through the tables for each power calculation based on 9 points data and these equations can be used to generate power numbers for each point within the smallest and the biggest point used.

3.3 Overview of the energy/power estimation algorithm followed by the power characterization tool

3.3.1 Dynamic Energy Simulation

Figure 3.3 illustrates the generic input transition energy estimation methodology used by the circuit simulator tool for both pin and buses.Relevant input and internal states are initialized as specified in the configuration file and a rise or fall stimulus with the specified input transition time is applied to the relevant input pin or a subset of pins of the bus beinf characterized.Power supply current probes are inserted and used to measure the relevant average supply current for the rise or fall event over a time interval that covers the entire rise or fall transition duration.The leakage current is then subtracted from the average supply current and used to estimate the energy for the input transition.The final energy is reported as an average of energies over different subsets of bus input pins.

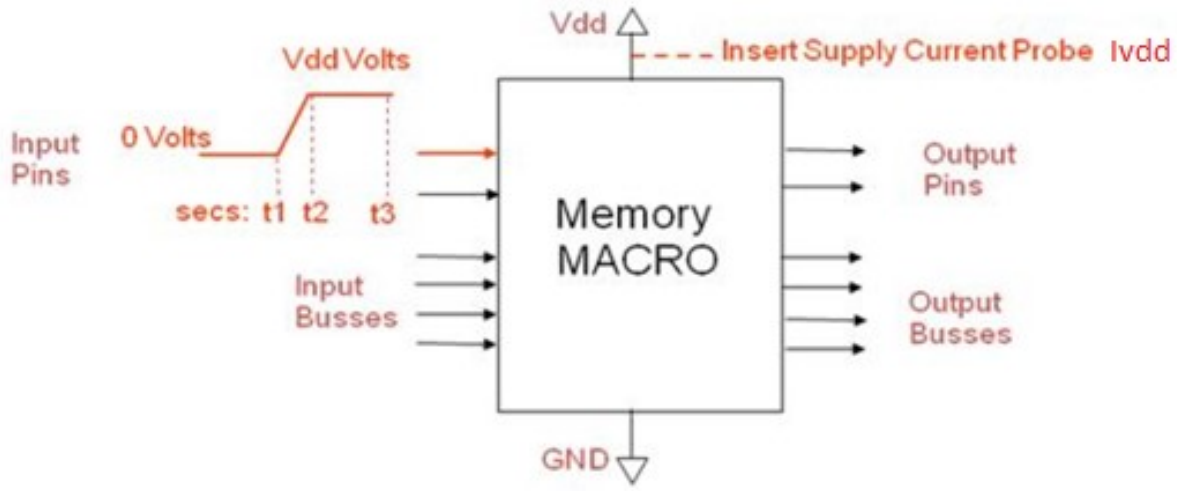

Figure 3.3: Power characterization for input pins

Input pin energy is estimated as:
Relevant Rise or Fall stimulus applied to single input pin
Energy = T * Vdd * {average Ivdd over interval T − average I_leakage over interval T}
T = t2 – t1
I_leakage = leakage current estimated as average supply current in interval (t2, t3)

For input pin simulation the assumption has to be made that there are no output pin rise/fall events existing within the time interval t2-t1

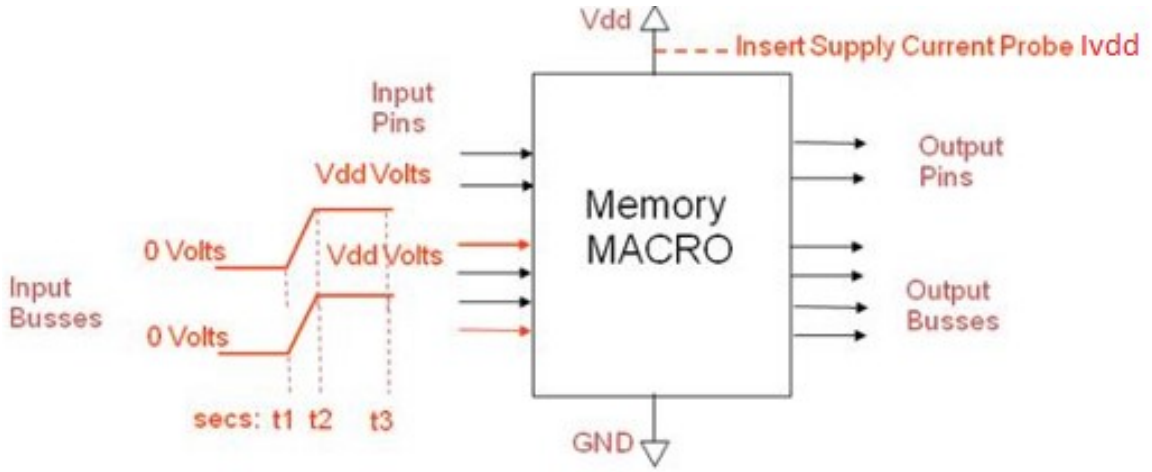

Figure 3.4: Power characterization for input bus

Figure 3.5illustrates input bus energy estimation methodology:-
Input bus energy is estimated as:
Relevant Rise or Fall stimulus applied to subset of multiple bus input pins
Energy = (T * Vdd * {average Ivdd over interval T − average I_leakage over interval T}) / N
T = t2 – t1
I_leakage = leakage current estimated as average supply current in interval (t2, t3)
N = Number of input bus pins with rise or fall events
Final Energy reported is average of Energies over different subsets of bus input pins

Figure 3.5 illustrates the output pin energy estimation methodology:-
Output pin energy (ie. related Input pin to Output pin power-ARC energy) is estimated as:
Relevant Rise or Fall stimulus applied to single related input pin causing the output pin event
Energy = T * Vdd * {average Ivdd over interval T − average I_leakage over interval T} − Factor
Factor = BASELINE_energy + $\frac{1}{2}$ C*Vdd*Vdd
BASELINE_energy = Energy for the related input pin events (with outputs stable)
PTPX adds back $\frac{1}{2}$ C*Vdd*Vdd as net switching energy
T = t2 – t1

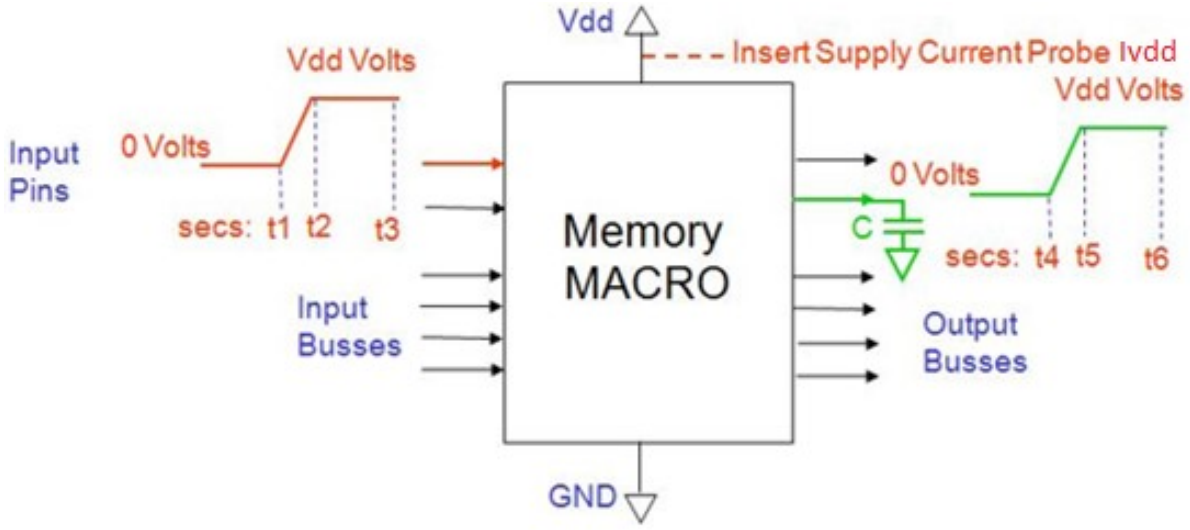

Figure 3.5: Power characterization for output pins

The tool also provides leakage current calculation over the time interval (t2,t3) which is calculated as
Leakage Power=Vdd * average I(leakage) over interval (t2,t3)
I(leakage)=leakage current estimated as average supply current in interval (t2,t3)

3.3.2 Leakage Power Simulation

The leakage power calculation is also having two different types

- LeakageActive Power Calculation
- LeakagePowerGated Power Calculation

The power gated designs have the structures as shown in Figure 3.6

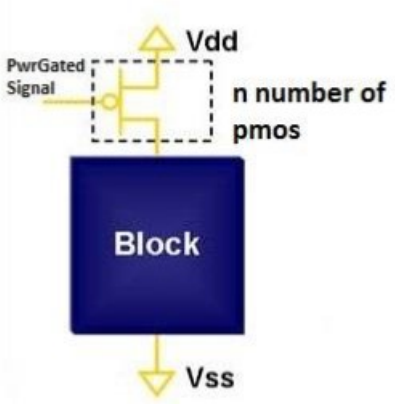

Figure 3.6: Power Gated Designs

1. LeakageActive Power Calculation
   In the power gated designs, when the "PwrGatedSignal=0", the pmos stack is on So there is leakage current so in this case both pmos stack and logic block leaks hence the leakage power is addition of the leakage powers of pmos stack and the leakage power of logic block.

2. LeakagePowerGated Power Calculation In the power gated designs, when the "PwrGatedSignal=1", the pmos stack is off So there is no(very small-can be neglected) leakage current so in this case the leakage current of logic block is considerable hence the leakage power is due to the leakage current through logic block. When "PwrGatedSignal" is '1, there is a complete shutdown of memory and the memory outputs are "unknown" in this mode.

3.4 Challenges faced in Memory Power Characterization

Memory Compiler plays a dominant role in determining power, performance and area of modern SoC design. Generating accurate and efficient power models at every stage of design has become major concern to design engineers. Along with accuracy, runtime is another critical parameter. The power characterization flow should be able to support 15 to 19 different process, voltage and temperature corners (PVTs) with reasonable accuracy and efficient runtime, say 48hours for an additional PVT request from the customer. This is the primary motivation behind exploring various flows within circuit simulator engine and coming up with optimized settings to support power characterization flow for memory compilers. Although the characterization flow manager algorithm is parallelized at multiple levels, the simulation runtime is what counts at the end. To brief on, here are the challenges that we face on memory power characterization front.

- Runtime vs. accuracy trade off
- Faster DC convergence time for post layout simulation
- Capacity issues with spice engine

The two major factors that affect simulation run time are:

1. **Design build time**
   Design build time implies the time taken to parse the schematic or parasitic netlist. Memory designs become more complex with the increasing demand for feature sets. This in turn results in an exponential rise in the number of MOSFETs in memory schematic. With TCAM blocks which contain more than 2.5 million transistor count,

the circuit build time along with the parasitic data becomes significant. Hence flattened netlist based simulation no longer holds to be a feasible solution. Here comes the need for pre-characterization technique such as dynamic pruning, back annotation flow or active net based flow which is explained in Chapter 4.

2. **DC initialization time**
   DC initialization time has always been a bottleneck for spice based simulations. Increase in number of latches and power gated nodes in the circuit aggravate this challenge. DC initialization time accounts for 80% of the total runtime. Therefore it is clear that we need to cut down the time taken for DC operating point calculation in order to achieve faster convergence. The default convergence method in circuit simulator is much faster compared to spice method. Another approach is to do DC analysis at pre-layout stage and reuse the operating points during post-layout simulation. This helps in saving the DC initialization time as well as the time taken to load the flat netlist.

Chapter 4

Runtime enhancements techniques in power characterization flow

4.1 Optimization in RC extraction methodology

Need for Parasitic Extraction

Semiconductor process technology has been continually scaling down for the past four decades and the trend continues. Shrinking process geometries, combined with the use of new device structures and an increasing number of metal layers at each new process node, are introducing millions of new parasitic effects in designs.Reduction in design sizes and complexities are increasing the sensitivity of circuits to parasitics due to the increasing impact on signal timing, noise and power. To ensure a successful silicon design and meet tape-out schedules, IC designers need an advanced parasitic extraction that delivers signoff accuracy and increased designer productivity.

Post-layout simulation runtimes are increasing 2x to 4x with every new process generation as chip transistor counts double and new parasitic effects come into play.Designers of custom digital, analog/mixed-signal (AMS) and memory integrated circuits (ICs) must now manage an ever-increasing volume of post-layout data while meeting the razor-thin design margins and tight project cycle times. These conflicting challenges are driving the need for more accurate and efficient parasitic extraction and simulation solutions to accelerate verification and achieve first time right silicon.

In general, advancing process technology is magnifying several device and interconnect parasitics considered secondary in previous technologies to primary factors affecting circuit behavior.For instance, interconnect and device parasitic effects are estimated to account for over 60 percent of the delay at 28-nm as shown in Figure 4.1.

Also,the project cycle times are shrinking or remaining constant at 12 to 18 months.The

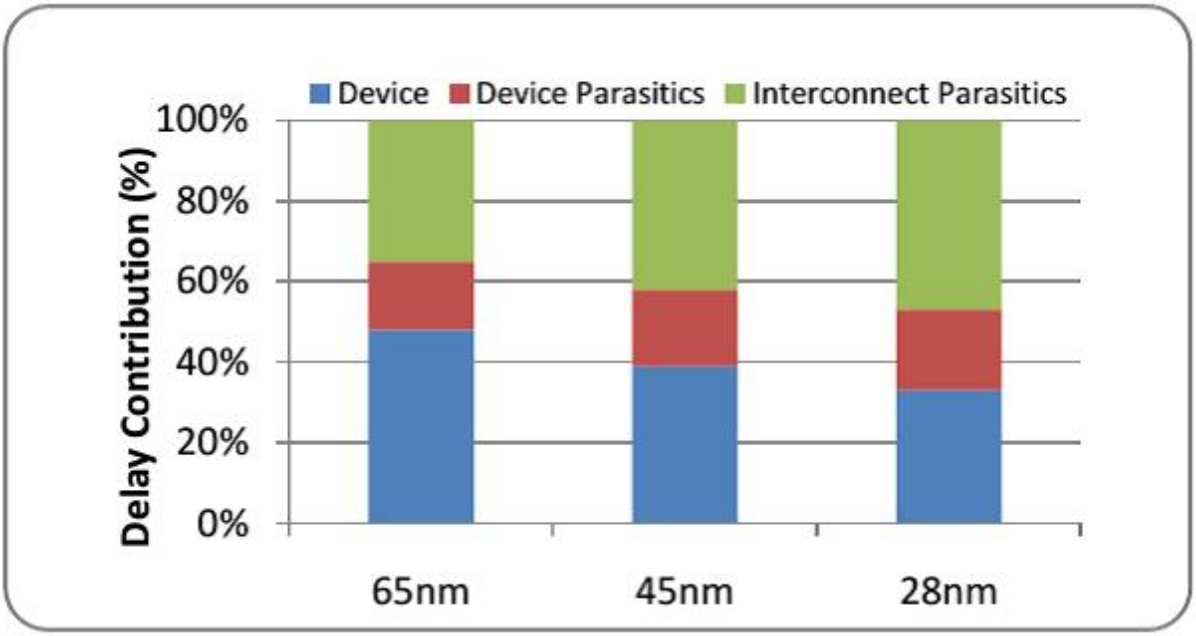

Figure 4.1: Increasing impact of parasitics on delay at advanced process nodes

number of post-layout simulation runs is increasing with the increase in the number of process corners needed for timing, signal integrity and power signoff.As a result,with ever-increasing accuracy requirements, potentially unmanageable volumes of data (billions of transistors and billions of parasitics) and a severe loss of productivity,we need a high-yielding successful silicon design with high delivering signoff accuracy indeed calling for a need of advanced parasitic extraction solution to boost simulation performance and designer productivity.

Post-Layout Simulation Challenges:

IC designers face three inter-related challenges in post-layout simulation, as listed below:

- Increasing extraction runtime and storage memory
- Increasing parasitic netlist size
- Increasing simulation runtime and storage memory

As the number of transistors increases, the number of nets in the design also increases proportionally. This means that the extraction tools need to extract, manage, and pass much more data to downstream simulation tools, resulting in an increase in parasitic extraction runtime. However, simulation runtime and capacity create larger bottlenecks in post-layout verification compared to parasitic extraction. Simulation runtime and capacity are directly related to the parasitic netlist size as well, that is, to the number of parasitic elements or nodes in the generated netlist. A full-chip parasitic extraction of all nets in a design can lead to unnecessary simulation inefficiency without improving accuracy. In order to meet

this challenge, the extraction tools must have the ability to selectively extract only the nets that are important for circuit behavior and netlist only the parasitics that are needed for accuracy.

Another challenge is associated with the varying simulation methodologies and verification goals, due to the designer's specific application needs. For instance, designers may want to take advantage of the pre-layout schematic netlist hierarchy and use hierarchical back-annotation to boost simulation performance. They may also want to trade off accuracy for higher performance and capacity. Consequently, the extraction and simulation tools must have the ability to support various verification flows and provide adequate flexibility to make necessary accuracy vs. performance and capacity trade-offs, if needed.

Overall, post-layout simulation flows have multiple productivity challenges. The parasitic extraction tools must offer advanced techniques to optimize and address all key areas of bottlenecksextraction runtime and capacity, parasitic netlist size and simulation runtime and capacity.

4.1.1 By using Lumped Capacitance Extraction or Cap-only extraction:-

Distributed RC Model-The distributed RC model is combination of n number of lumped RC model. The distributed RC model is more appropriate in terms of accuracy. It is known however that the distributed RC line can be approximated by a lumped RC ladder network. The distributed rc-model is complex and no closed form solutions exist.

Lumped RC Model-This model lumps the total wire resistance of each wire segment into one single R and similarly combines the global capacitance into a single capacitor C. This simple model, called the lumped RC model is pessimistic and less accurate. But for reducing the run time of extraction, this model is quite useful.

The behaviour of the distributed rc-line can be adequately modelled by a simple RC network.Figure 4.2 shows both models.

As the lumped RC modelling is having less number of R and C as it is approximation to the original RC circuit so the extraction will take comparatively less time than distributed RC modelling.

The runtime improvement results for this methodology are shown in chapter5.

Cap-only extraction-Sine the current during the transition depends on the charging and discharging of parasitic and interconnect capacitance,we parse only parasitic capacitance value for the post layout simulation.By doing this,the parasitic netlist size reduces to a great extent and the simulation runtime also decreases as expected.

The runtime improvement results for this methodology are shown in chapter5.

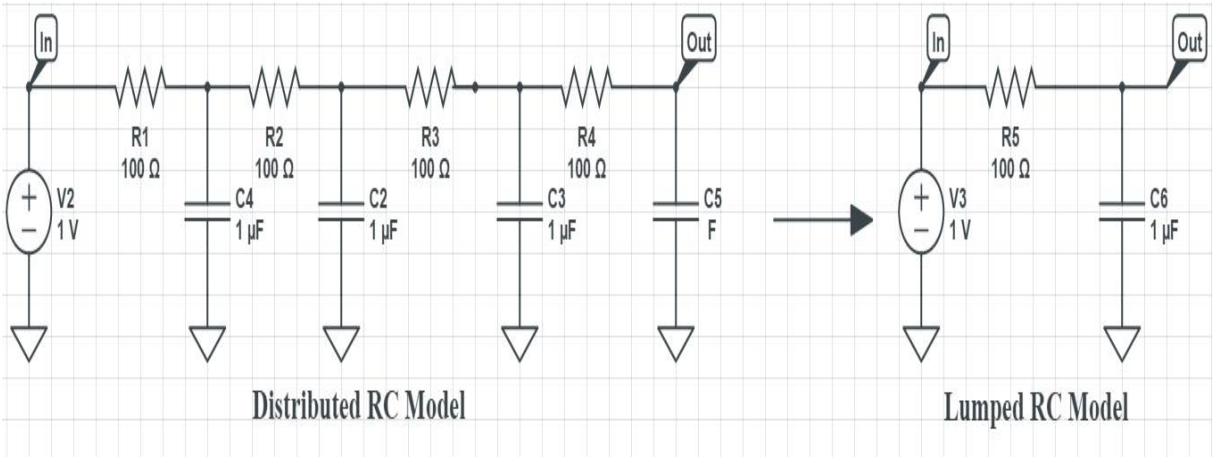

Figure 4.2: RC line models

4.1.2 StarRC Features to Accelerate Simulation:-

Synopsys'StarRC parasitic extraction toolhave demonstrated up to 10x acceleration in simulation performance for most applications.In particular, StarRC's exclusive interface with Synopsys' CustomSim™ simulation solution provides unique productivity advantages to the users of the two tools. Some of the proven StarRC extraction techniques to increase post-layout verification productivity are presented in this paper, as follow:-

- Active node extraction and simulation with CustomSim
- Post-layout acceleration with CustomSim hierarchical back-annotation
- Hierarchical parasitic extraction
- Selective parasitic extraction
- Selective parasitic netlisting

The following sections describe the techniques in detail and highlight the benefits of each for specific design applications

1. **Active Node Extraction and Simulation**

   StarRC's active node back-annotation flow with CustomSim provides an order of magnitude performance and capacity benefit for large transistor-level designs. Designers performing post-layout simulation may be sacrificing performance by extracting all nets in the design and passing full parasitics information to downstream simulators, even though only a fraction of the nets may actually impact timing or accuracy. This is especially true for signals that are not switching or might be inactive during the course of the simulation period. A comprehensive "all net" extraction may not be necessary in such situations, since the parasitics have no effect on circuit performance or on the functionality. Increased data volume only results in an increased extraction

runtime, netlist size, and decreased simulation performance with no meaningful gain in accuracy or silicon predictability. This is particularly applicable to memory designs; a memory (such as an SRAM) generally has a small portion that is accessed at a given time, thus making memory designs very attractive for an active node extraction and simulation flow.

The active node flow between StarRC and CustomSim is shown in Figure 4.3. The flow consists of a push-button command in CustomSim that generates a list of nodes with voltages exceeding a user-specified value in pre-layout simulation. StarRC then directly reads the list and extracts the active nets and the nets coupled to the active nets. The extraction of the active nets only and its dominant coupling neighbors significantly reduces the runtime, as well as the netlist size while guaranteeing accuracy on each extracted net. As the simulation runtime is proportional to the number of nodes and parasitic elements in the netlist, the reduced netlist size improves simulation performance as well. In some design cases, this can reduce the extraction task from hundreds of thousands of nets to only a few thousand nets, thus significantly accelerating simulation. The StarRC and CustomSim active node flow is proven on large transistor-level designs, yielding greater than 10x productivity benefit.The results will be presented in chapter 5.

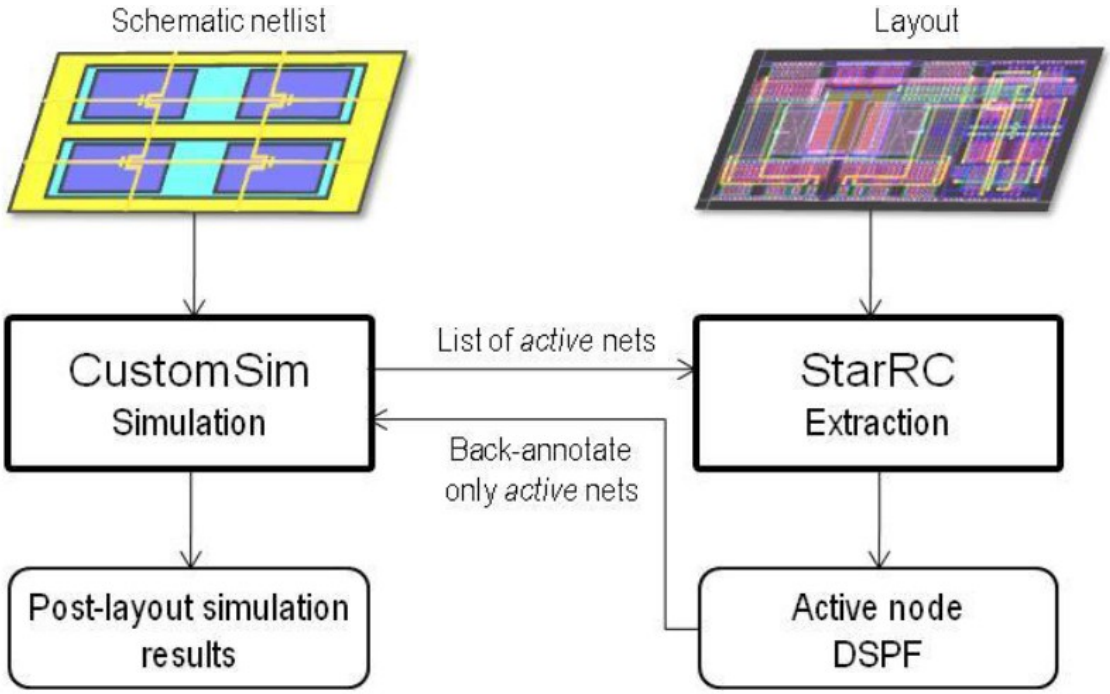

Figure 4.3: The active node flow between StarRC and CustomSim delivers push-button productivity

The Active Net Based flow is a utility provided with Back Annotation flow.There are

two phases to complete this flow: **setup phase and simulation phase**.

In the setup phase, the circuit simulator tool generates an active net file containing the active net names. The tool determines if a net is active by monitoring the change in voltage value.

In the simulation phase, the active net file generated from the setup phase is parsed to Star-RC to only extract the parasitics on the active nets into a parasitic netlist file in SPEF or SPF format; therefore, the parasitic netlist file is much smaller when you extract all the nets. The parasitics are back-annotated to the ideal schematic netlist for final simulation.

The selective-net back-annotation extraction-simulation post-layout flow is very efficient for high latency medium-to-large designs, such as SRAM designs. Even though the setup phase is required to generate an active net file, the total run time is still faster than extracting and simulating the whole design. In addition, the overall memory usage is reduced.

Detailed setup and simulation phase during the selective-net back-annotation extraction-simulation post-layout flow is given as below:-

Setup Phase :-
Run simulator to generate an active net file with the command *set_ba_active* file to generate the active net file in simulator.By default, if a net varies more than 100mV, it is considered active.

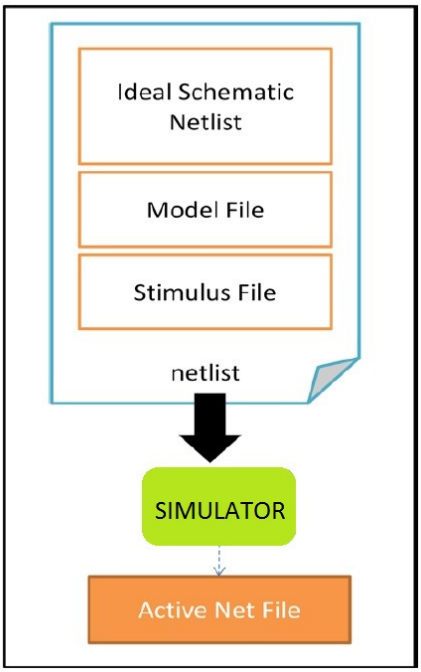

Figure 4.4: Setup Phase

Once the active net file is generated the extracted RC details is much less as compared to the RC parasitic details for the whole network.Figure 4.7 below depicts the advantage of active net approach over traditional approach

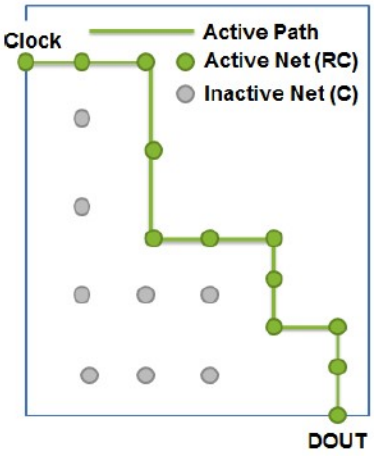

Figure 4.5: Simulator selecting Active nets based on threshold voltage on nets

Simulation Phase :-

Run simulator to read-in the active net file to only back-annotate the parasitic on those nets.

The *set_ba_option* can be used to load in the active net file.

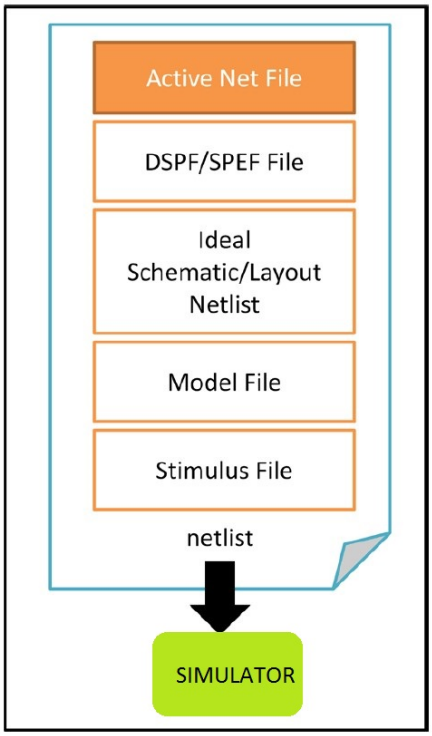

Figure 4.6: Simulation Phase

2. **Post-layout Acceleration with CustomSim Hierarchical Back-annotation**

   Flat netlist based flow is used by default.The stimuli options and post layout data is stitched into the flattened netlist which is input to the simulator. The disadvantage here is the huge netlist parsing time. The DC initialization time for post-layout netlist is much higher than that for pre-layout netlist.

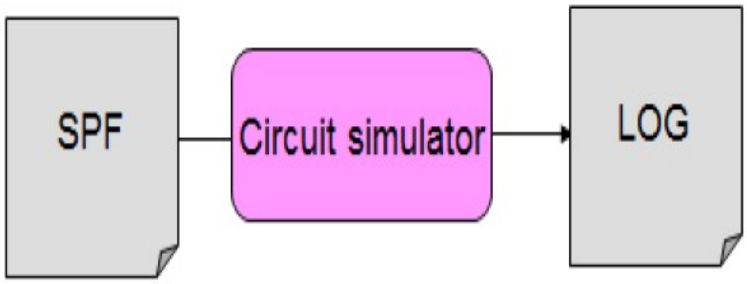

Figure 4.7: Default flow(Flat netlist based)

StarRC offers industry-leading performance and capacity combined with silicon-accurate flat parasitic extraction. StarRC's flat full-chip extraction in conjunction with CustomSim's hierarchical back-annotation enables designers of large memory and custom ICs to achieve the best combination of fastest simulation turn-around time and golden signoff accuracy.

In general,Back annotation flow enables back-annotation of the specified parasitic netlist file. More than one parasitic netlist file can be back-annotated.

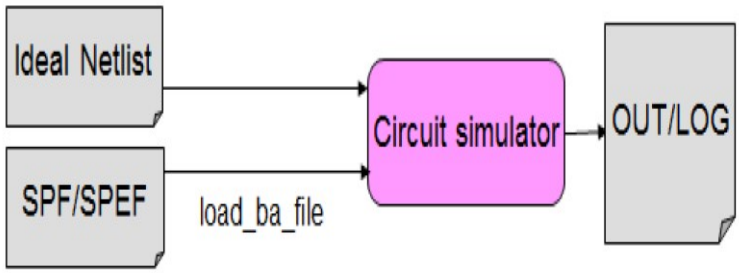

Figure 4.8: Back Annotation Flow

Designers typically reuse the pre-layout simulation test-benches based on the schematic netlist for their post-layout simulation with the extracted parasitics.Hierarchical simulation in CustomSim is driven by the schematic netlist hierarchy, as well (see Figure 4.9). CustomSim takes advantage of the natural design hierarchy to provide the highest-efficiency simulation. CustomSim's advanced Post-layout Acceleration (PLX)

option includes isomorphic hierarchical back-annotation technology that allows designers to easily annotate the post-layout parasitics onto the pre-layout schematic netlist, while preserving the original hierarchy. Combined with the flat extraction parasitics from StarRC, this offers the most effective flow to achieve the twin goals of higher simulation performance and signoff accuracy. StarRC's flat extraction technology accounts for all neighboring net coupling capacitances as well as thickness and width variation effects that are essential for signoff accuracy. On the other hand, the PLX technology provides the maximum annotation of the detailed parasitics using design matching of the post-layout netlist and schematic netlist.

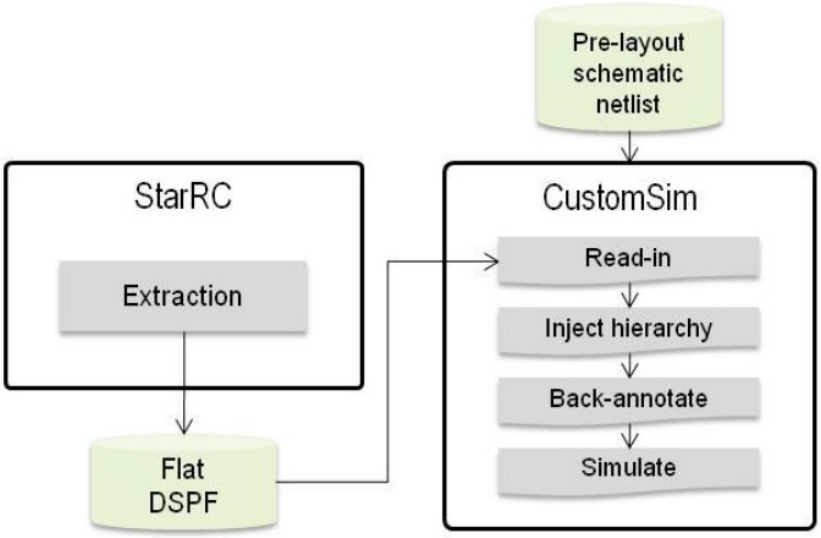

Figure 4.9: StarRC and CustomSim hierarchical back-annotation flow enables schematic netlist efficiency in post-layout simulation

3. **Hierarchical Parasitic Extraction**

StarRC's hierarchical extraction and netlisting offers circuit designers another option to improve their post-layout verification productivity. For most analysis, a flat parasitic extraction with hierarchical back-annotation provides the best combination of accuracy and performance. However, in cases such as the reliability analysis of large memory or custom system-on-chip (SoC) designs, designers may need to extract billions of signal and power net parasitics. Flat parasitic extraction, though most accurate, could be time-consuming in such cases; therefore, designers may prefer hierarchical extraction to speed the parasitic extraction process. StarRC's hierarchical extraction complements its flat extraction technology by providing optimized hierarchical parasitic data for high-capacity signal and power net analysis.

The hierarchical extraction technology allows designers to perform bottom-up or top-down simulations, depending on the demands of their methodology. As shown in Figure 4.10, hierarchical extraction improves the capacity and runtime by extracting only one instance of a block per hierarchy level while accounting for as much context specific capacitance at the cell boundary as possible. Generally, hierarchical extraction is best suited for simulation tools that can take advantage of the hierarchical data to improve runtime, or in situations where more flexibility in simulation netlist handling may be desired.

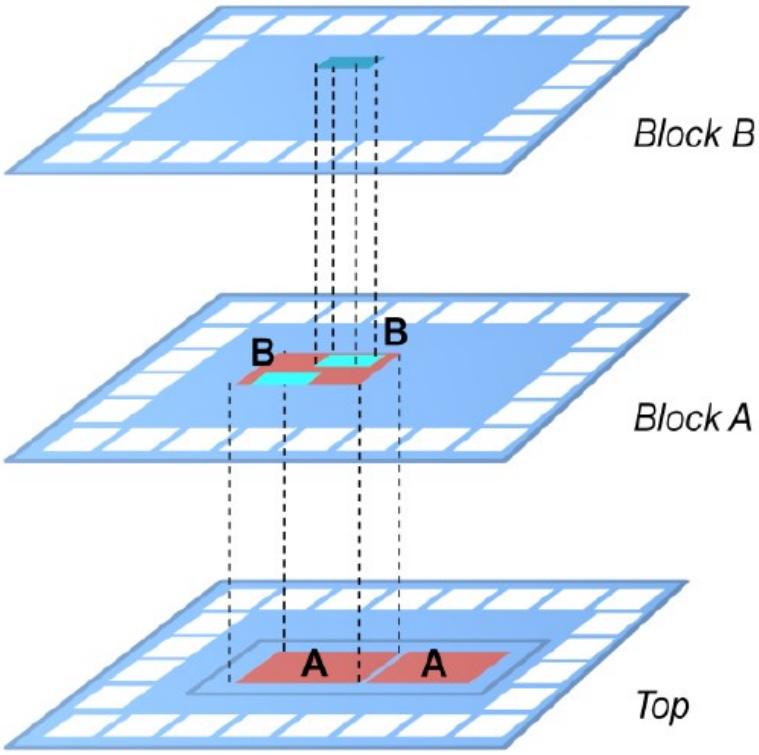

Figure 4.10: StarRC hierarchical parasitic extraction supports top-down or bottom-up methodology

4. **Selective Parasitic Extraction**

The selective device parasitic extraction feature of StarRC allows custom IC designers to choose and retain the critical device parasitics in their designs while ignoring less important coupling capacitances and power net parasitics. This increases simulation efficiency while meeting accuracy requirements.

At the 40-nm process technology, device parasitics such as gate-to-contact capacitance (Cco) and gate-to-diffusion capacitance (Cf) become context-specific (see Fig-

ure 4.11)a, for example, they become more sensitive to the layout environment and have a pronounced impact on circuit performance. The magnitude of these capacitances is seen to increase by at least 2x compared to previous technology; therefore an accurate in-context device parasitic extraction becomes more important. In larger process nodes, these capacitances were generally included in device SPICE models, assuming pessimistic values with contact spacing at minimum and not differentiating shared source/drain diffusion.However, at advanced nodes, this pessimism must be removed and the device parasitics must be accurately extracted to achieve better silicon predictability.

Power net extraction has generally been assumed to be ideal in post-layout simulation flows, but this assumption may be inaccurate when predicting circuit behavior at smaller process nodes. To achieve signoff accuracy at smaller process nodes, designers may need to extract resistance and coupled capacitance for both power and signal nets, but this could prohibitively increase the simulation runtime. An effective alternative solution may be to simulate the design with grounded capacitance. However, due to the growing impact of context-specific device capacitances and due to the associated Miller effect, designers may prefer to selectively retain the gate-to-contact and gate-to-diffusion capacitances to ensure accuracy, but ground rest of the coupling capacitances between the interconnecting layers to improve efficiency.

Also, metal-to-diffusion contact resistance plays a major role in power network simulation at 40-nm (see Figure 4.11b). For example, diffusion contact resistance for a 40-nm process is of the order of tens of ohms, whereas metal sheet resistance is of the order of a few milli-ohms. Hence, power network resistance is dominated by contact resistance and the extraction tools must have the capability to accurately extract this resistance and also provide a selective extraction based on layers for productivity, if needed.

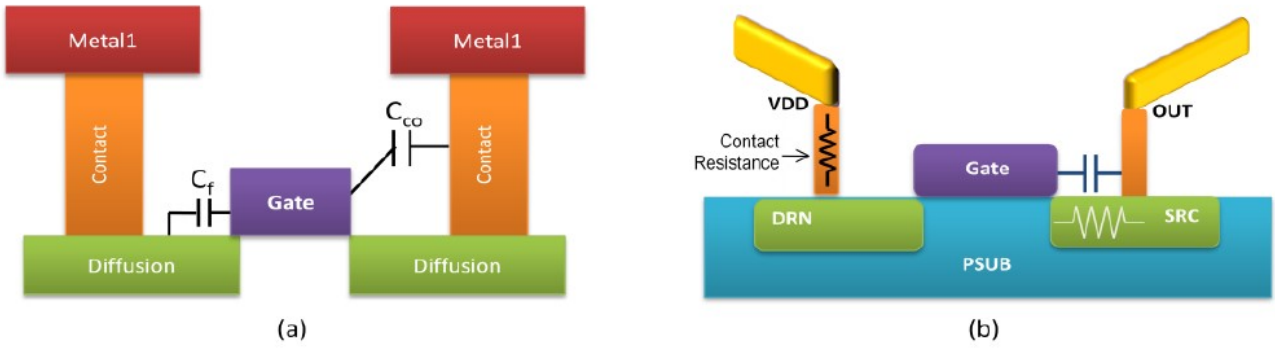

Figure 4.11: Context-specific device parasitics have pronounced impact on circuit performance at advanced nodes and therefore must be accurately extracted

Overall, parasitic extraction tools must have the ability to selectively extract device parasitics that may dominate the circuit behavior to enable the most efficient simulation while preserving accuracy. StarRC provides an advanced feature to selectively choose device parasitic extraction based on database layers. The innovative reduction algorithms in StarRC allow designers to selectively retain device parasitics, such as diffusion resistance and gate-to-contact coupling capacitances, while treating interconnect on power nets as super conductive (ideal) and grounding other coupling capacitancesthis produces the most efficient netlist for simulation. The selective extraction capability enables designers to achieve significant runtime savings as demonstrated by over 5x simulation speed-up on production designs.

5. **Selective Parasitic Netlisting**

   StarRC provides flexible netlisting options to reduce the parasitic netlist size and proportionally increase simulation performance. As mentioned earlier, the simulation runtime is proportional to the number of nodes or parasitic elements in the netlist. The circuit behavior and functional requirements of the designs determine which nodes or parasitic information need to be passed to post-layout simulation tools.

   StarRC's proven netlist reduction techniques allow designers to selectively filter the parasitics that may have negligible or no impact or circuit performance based on their design knowledge. It offers easy-to-use resistance and capacitance filtering as well as net filtering based on user-defined values. For instance, the user can easily reduce the number of nets in the netlist by filtering the nets based on total capacitance. The netlist reduction in StarRC preserves the total grounded and coupling capacitance, point-to-point resistance, and user-defined circuit delay to achieve the desired accuracy.

   The experiments with selective parasitic netlisting shows that there is a reduction in the number of elements in the parasitic netlist when the selective netlisting is switched on. This reduction enables significant simulation speed-up with no accuracy loss.

4.2 Optimization in power simulation methodology

4.2.1 Average case simulation with scaling fraction=1

As discussed in previous chapter,the methodology we follow for dynamic energy calculation and leakage power(active and gated) calculation initially required two types of simulation:-

- **Worst Case Power Calculation**

  The Worst case power calculation is done by initializing the state (internal) nodes in such a way that all nodes toggle in the circuit so that there will be maximum power dissipation in the circuit.

- **Best Case Power Calculation**

  The best case power calculation is done by initializing the state (internal) nodes in such a way that not a single node toggle in the circuit so that there will be minimum power dissipation in the circuit.

The final power calculation was done as

Average Case Power Calculation

The average case power calculation is done by using weighted scale factor, provided by the circuit designer in the configuration file, for calculation of average case power values using following formula

Average case power = [Worst case power x ScalingFraction] + [Best case power x (1- ScalingFraction)]

But the methodology has changed over a period of time and now only the worst case power calculation is considered as the primary power calculation by putting ScalingFraction=1
The formula for computing average case energy becomes
Average case power = [Worst case power x ScalingFraction] + [Best case power x (1- ScalingFraction)]
= [Worst case power x 1] + [Best case power x (1-1)]
Average case power = Worst case power

Hence in the new methodology the number of simulations becomes 50% compare to previous methodology hence the runtime for computing the average case energy values becomes 50% at the cost of very small accuracy. As the power or energy numbers need not to be accurate as timing values so small accuracy loss is acceptable. Similarly for the leakage simulation also the number of simulations reduced to 50% so the runtime also reduced to 50%.

The runtime enhancement results are shown in chapter 5.

4.2.2 Simulation with source on power gate

The hspice simulation tool is best suited for an accurate dynamic/leakage power simulation but it trades off with a high simulation time to provide sign-off accuracy and meet designer productivity.

The memory compiler IPs consist of the gated designs as shown in the figure 4.12 below.The design consists of a large number of such pmos transistors to provide the gated supply to the design.Sometimes the pmos transistors are provided as a gated slice explicitly between the bit slices in the design.Hence the partitioning done before hspice simulation becomes difficult.

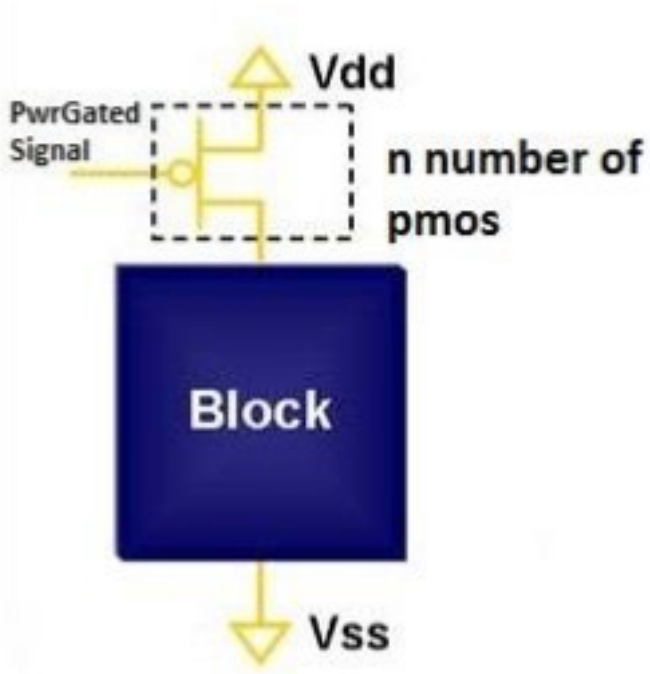

Figure 4.12: Power Gated Design

If we short the pmos transistor during dynamic energy calculation and active leakage power simulation,the partitioning into smaller blocks is done efficiently and hspice simulation becomes parallelized on the multiple threads.Hence we achieve great runtime improvement for the simulation with a trade off of ¡1% in accuracy which is feasible.

The runtime enhancement results are shown in chapter 5

4.2.3 Simulation with multi core option

Circuit simulator has built in feature to support mutli-core simulations at the cost of more number of licenses. This feature enables the user to make efferent utilization of parallel computing resources to speed up power simulations.

4.2.4 Configuring DC initialization algorithm

Circuit simulator supports various configurations of DC initialization algorithm.

1. Static - Partitions the circuit before solving the operating point.
2. Adaptive - Automatically adjusts the partition during the solving stage for better DC Convergence.
3. Spice - is the most accurate DC method. It does not partition and uses the Newton-Raphson iteration and pseudo-tran approaches for solving one single matrix. Spice DC method takes weeks' time to converge on bigger blocks like TCAM and possess more challenges for post layout simulation.

Default DC algorithm in circuit simulator is automatic, where it starts from an aggressive method, and if DC convergence is not achieved, it goes to more conservative method. Back Annotation flow reuses initial conditions form pre-layout simulations to post-layout.

The runtime enhancement results are shown in chapter 5

4.3 Optimizing simulator benchmarking through a regression suite

For compiled memory IPs the post layout simulation is usually carried out by hspice simulation tool or fast-spice simulation tool.Both shows trade off in accuracy and simulation runtime.In order to give accurate sign-off and designer productivity,we explore different simulator options and environment settings.

To test the new version of simulator with the newly added options,we benchmark the same set of pvt corners and instances using the simulator version with added options with a reliable trend giving simulator version.We usually compare the accuracy difference and runtime improvement results in a user readable format i.e excel sheet.

Flowchart 4.13 below specifies the steps we followed for a traditional benchmarking task.

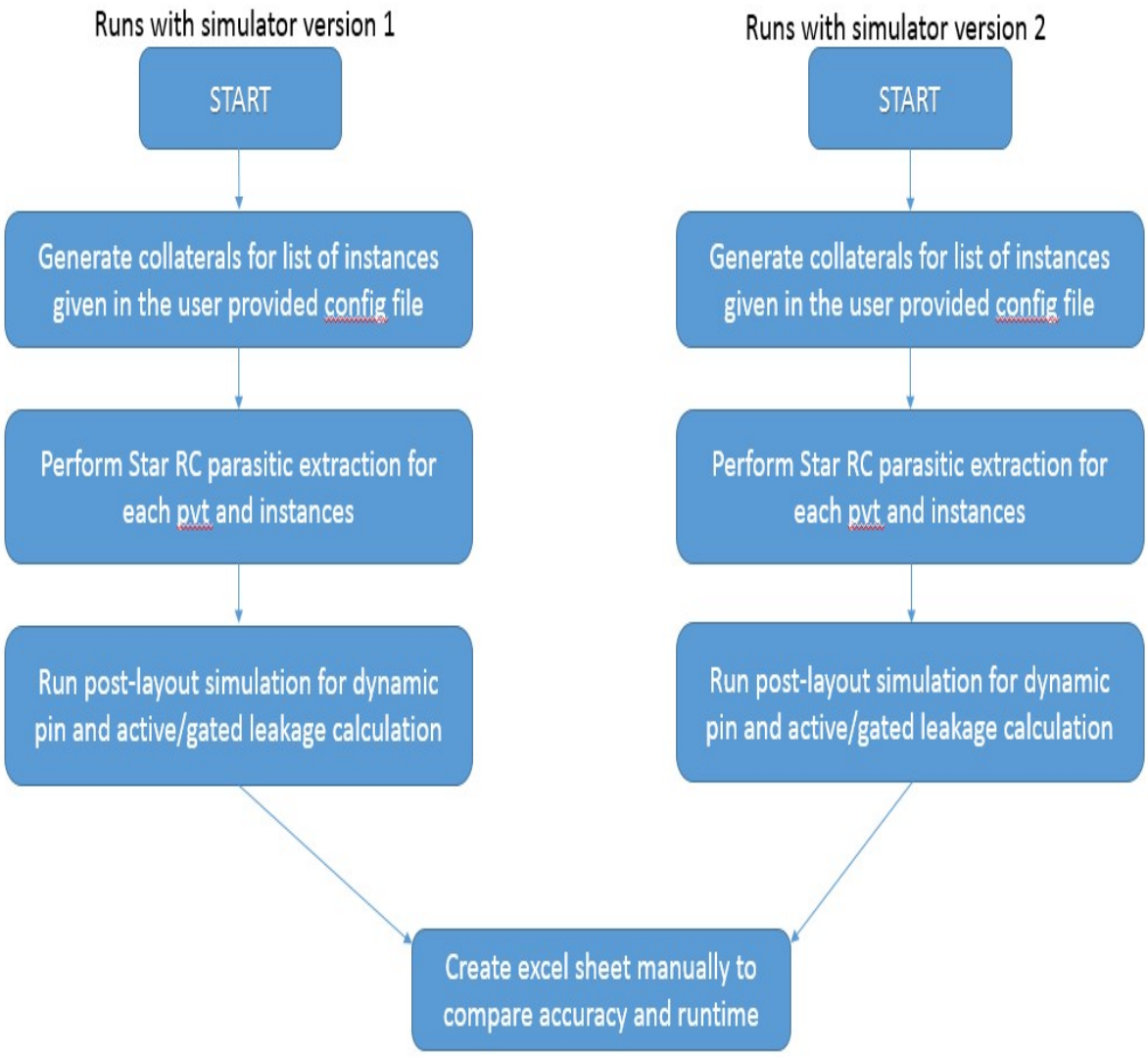

Figure 4.13: Old methodology to benchmark two different simulator versions involving human intervention

Hence,the existing method was a time consuming and tedious job as it required a lot of human intervention.

To ease the benchmarking task in order to provide accurate sign-off and designer productivity,a regression suite is developed.

1. The regression suite developed can benchmark the results to compare hspice vs fast-spice simulator of fast-spice vs fast-spice simulator with some newly added switches in order to tweak the methodology.

2. The regression suite generates the collaterals and perform RC parasitic extraction only

one time in order to maintain the corresponding variables as constant between the two regressions.

3. It creates an excel sheet automatically once both the regressions completes in order to compare the accuracy loss between the two simulator versions and showcase the runtime improvement we achieve by exploring new simulator options.

A self-explanatory flowchart below 4.14 describes various steps that are being carried out while benchmarking new version of simulator with the developed regression suite.

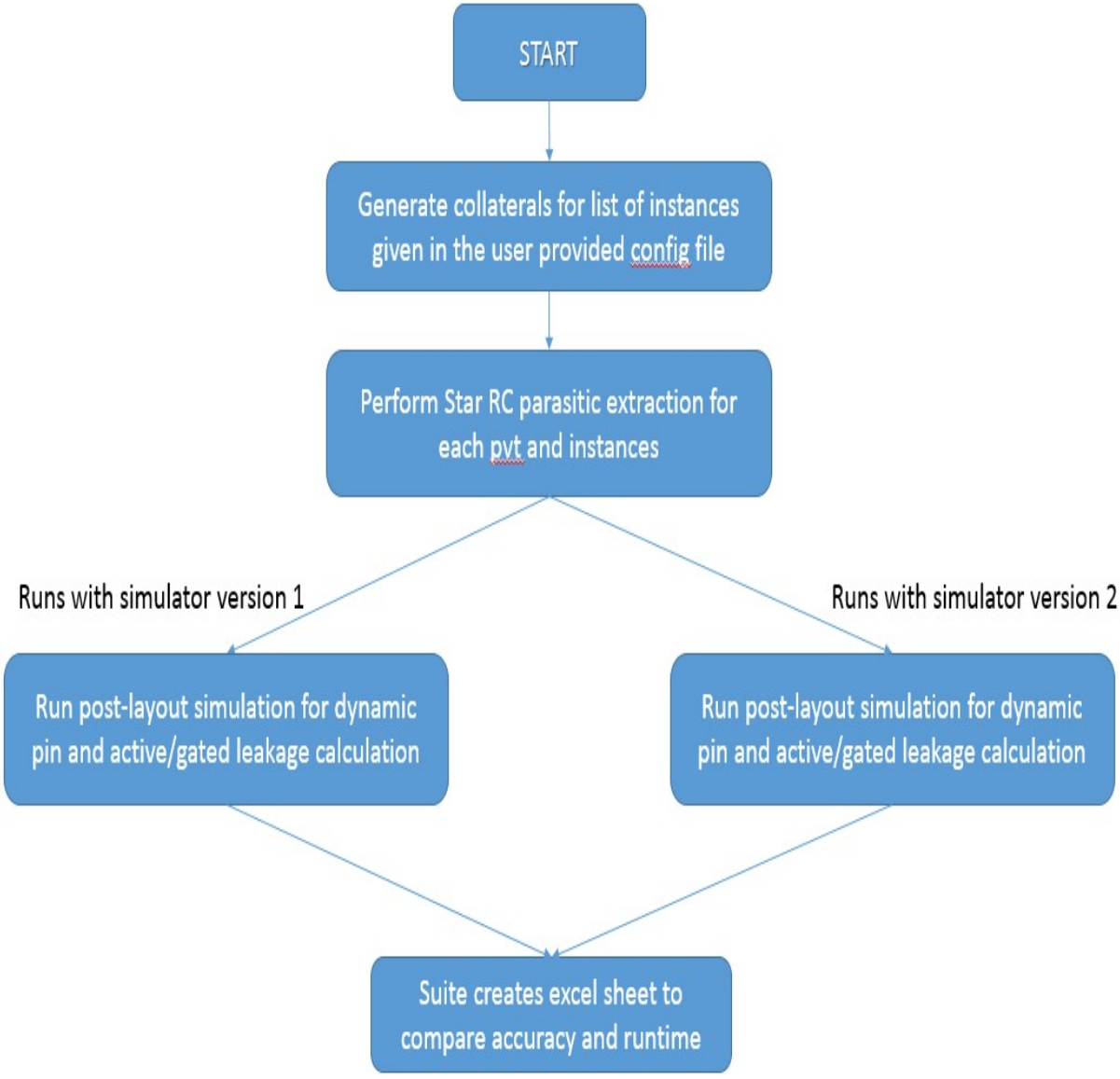

Figure 4.14: Methodology followed in the regression suite to benchmark two different simulator versions

The runtime improvement results are shown in chapter 5.

Chapter 5

Experimental Results and Conclusion

5.1 Runtime Enhancement By using Lumped Capacitance Parasitic Extraction

As the lumped RC modelling is having less number of R and C as it is approximation to the original RC circuit so the extraction will take comparatively less time than distributed RC modelling. The runtime improvement results for this methodology are shown in table 5.1

Table 5.1: Comparative results for Distributed Vs Lumped Parasitic Extraction

Parameters	**Distributed parasitic Extraction**	**Lumped Parasitic Extraction**
Runtime	X Hours	0.75X Hours
Accuracy	X Joules	0.96X Joules

The result shows that there is 25% of runtime improvement but at the small cost of accuracy in the energy values of pin based characterization.

5.2 Runtime Enhancements By using Cap Only Extraction

The cap only extraction methodology ignores the resistance(R) information of the RC model of the line and only C value is included in the extraction data.Since the extraction data is reduced in length,the simulation undergoes at a much faster rate as depicted in the table 5.2 below:

Table 5.2: Comparative results for Cap only Vs Distributed RC parasitic extraction

Parameters	**Distributed parasitic Extraction**	**Cap only Extraction**
Runtime	X Hours	0.11X Hours
Accuracy	X Joules	0.99X Joules

The result shows that there is 89% of runtime improvement but at the small loss of accuracy of 0.01% in the energy values of pin based characterization.

5.3 Runtime Enhancement By Computing Simulation using ScalingFraction=1

As in the new methodology the number of simulations becomes 50% compare to previous methodology hence the runtime for computing the average case energy values becomes 50% at the cost of very small accuracy. As the power or energy numbers need not to be accurate as timing values so small accuracy loss is acceptable. Similarly for the leakage simulation also the number of simulations reduced to 50% so the runtime also reduced to 50%. The runtime enhancement results are depicted in table 5.3.

Table 5.3: Comparative results for Scaling fraction=1 and 0.5 experiments

Parameters	**Dynamic**		**Leakage Active**		**Leakage Power Gated**	
	Old	**New**	**Old**	**New**	**Old**	**New**
Runtime	X Hours	X/5.1 Hours	X Hours	0.6X Hours	X Hours	0.41X Days
Accuracy	4-7% accuracy loss		2-4% accuracy loss		1-4% accuracy loss	

5.4 Runtime Enhancement with source on power gate

Providing source on the power gated node enables us to simulate for active leakage numbers and dynamic pin numbers at a much faster rate. The experiment is done on typical corner with hspice simulator as the simulation tool.Table 5.4 shows the runtime enhancement for medium size instances.

Table 5.4: Comparative results for runs with and without source on power gated node

Parameters	**Dynamic**		**Leakage Active**	
	Default	**Source on PG**	**Default**	**Source on PG**
Runtime	X Hours	0.1X Hours	X Hours	0.1X Hours
Accuracy	0.8% accuracy loss		0.6% accuracy loss	

5.5 By using Back Annotation (BA) Flow

“As shown in the table 5.5, the flow enables significant improvement in simulation performance with no change in accuracy. It is important to note that the majority of the improvement in the simulation runtime when using hierarchical back-annotation is during the transient analysis stage. This is due to the fact that CustomSim back-annotates the

parasitics onto the original schematic netlist as explained above, thus allowing the simulation to take full advantage of the design hierarchy."[11]

Table 5.5: Comparative results for flat netlist and back annotated flow

Test case ROM(device count= 0.5million	**Post layout simulation on entire design**	**Back annotation flow**
Accuracy(cuurent during transition[mA])	X (mA)	0.987X(1.3% accuracy loss)
Runtime	R(seconds)	0.56 R

The major advantages of using BA flow over flattened netlist areReuse of initial conditions from pre-layout simulations to post-layout Netlist parsing time is reduced Added flexibility for the designer to disable a number of unused nets.

5.6 Active net based flow

Table 5.6 shows that beck annotating the selected nets in this flow which is a runtime-saver.

Table 5.6: Comparative results for flat netlist and active net based flow

Test case ROM(device count= 0.5million	**Post layout simulation on entire design**	**Active net based flow**
Accuracy(cuurent during transition[mA])	X (mA)	0.983X(2.1% accuracy loss)
Runtime	R(seconds)	0.469 R

5.7 Multi-core options built in circuit simulator

Table 5.7: Comparative results for flat netlist and multi core flow

Test case ROM(device count= 0.5million	**Post layout simulation on entire design**	**Multi core options**
Accuracy(cuurent during transition[mA])	X (mA)	0.983X(2.1% accuracy loss)
Runtime	R(seconds)	0.249 R

The overall runtime for power characterization for memory compiler at medium ranges proves to be 2.5 times faster than default method.

5.8 DC initialization

Table 5.8: DC initialization

IP Type	**Biggest block(Transistor Count)**	**Spice method**	**Initialization time with previous version**	**Initialization time with latest version with hot fix**
Single port Register File	Device count = 0.6 million	Stuck at DC initialization stage over a week	X Hours	0.67X Hours

Conclusion

Memory power characterization plays a major role in total design turnaround time. Our experiments were focussed on improving the runtime from simulator perspective thereby reducing circuit build time and DC initialization time. As a result, the overall runtime for power simulations were reduced to 0.1X with 0.8% accuracy loss by sourcing the power gated node the the rail voltage Vcc when compared to default spice method. The Runtime has been reduced by maintaining small accuracy loss.

The hspice simulator shows a great runtime improvement for the acitve leakage power number and dynamic pin energy calculation,but could not apply the same methodology for power gated leakage number as no voltage should be applied to the memory circuit while calculating power gated leakage.So we follow the same default methodology for these runs. Thus the methodology helping in partitioning the circuit efficiently while simulating for leakage calculation proves to be beneficial to meet the sign off accuracy and less simulation runtime for the designers.

References

[1] International Technology Roadmap for Semiconductors, 2003. `http://www.publicitrs.net`

[2] SIA Roadmap `http://www.semiconductors.org`

[3] P Mridula,Dayanand N Naik, *"A Novel Approach towards Power Characterization of Compiled Memory IP"*,SNUG,2014

[4] Synopsys Inc *CustomSim User-Guide Version I-2013.12,2013*

[5] Synopsys Timing Constraints and Optimization *User Guide Version J-2014.09-SP2,December 2014*

[6] Brachmann,C;Genser,A;Steger,C;Weiss,R;Haid,Josef , *"Automated Power Characterization for Run-Time Power Emulation of SoC Designs"*,Digital System Design,2010 13th Euromicro Conference on Sept,2010

[7] Charles Longway,Conexant Systems Inc.,You-Pang Wei , *"Automatic Memory IP characterization"* article in Legend Design Technology Inc

[8] Federico Politi,Ahmed Elzeftawi, *"Dynamic partitioning speeds Memory Characterization"*,Cadence Design systems

[9] You-Pang Wei and Bill Wasserman, *"Reliability based characterization of memory IP in SoC Designs"*,Legend Design Technology, Inc. Santa Clara, California, U.S.A.

[10] StarRC, Parasitic Extraction, Datasheet, Synopsys `http://www.synopsys.com`

[11] Synopsys Inc *"Parasitic Back Annotation for Post Layout Simulation"*, Application Note ,SIMUCAD