Methodology of RTL Power Analysis for modules on System-on-Chip (SoC)

A Major Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology

in

Electronics & Communication Engineering

(VLSI Design)

By

Patel Keyuri Mahendrabhai

(14MECV17)

Electronics & Communication Engineering Branch

Department of Electrical Engineering

Institute of Technology

NIRMA UNIVERSITY

Ahmedabad-382 481

May 2016

Methodology of RTL Power Analysis for modules on System-on-Chip (SoC)

A Major Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology

in

Electronics & Communication Engineering

(VLSI Design)

By

Patel Keyuri Mahendrabhai

(14MECV17)

Under the Guidance of

Dr. Niranjan M. Devashrayee

Electronics & Communication Engineering Branch

Department of Electrical Engineering

Institute of Technology

NIRMA UNIVERSITY

Ahmedabad-382 481

May 2016

Declaration

This is to certify that

(i) 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.

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

Patel Keyuri Mahendrabhai

Certificate

This is to certify that the Major Project entitled **"Methodology of RTL Power Analysis for modules on System-On-Chip(SoC)"** submitted by Patel Keyuri Mahendrabhai (14MECV17), towards the partial fulfillment of the requirements for the degree of Master of Technology in 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 or diploma.

Date: Place: Ahmedabad

Guide Program Co-ordinator

Dr. N. M Devashrayee
Professor, EC

Dr. N. M Devashrayee
Professor, EC

HOD Director

Dr. P. N Tekwani
Head of EE Dept

Dr. P. N Tekwani
Director, ITNU

Certificate

This is to certify that the Major Project entitled **"Methodology of Power Analysis for modules on System-On-Chip(SoC)"** submitted by Patel Keyuri Mahendrabhai (14MECV17), towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design of Nirma University, Ahmedabad is the record of work carried out by her at Broadcom Communications Technologies Pvt Ltd., under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.

Date: Place: Bangalore

Project Guide

Mr. Prasad Kotra
(Associate Technical Director, IC Design)
Broadcom Communications Technologies Pvt Ltd.

Project Manager

Mr. Rajesh Rajan
(Sr. Manager, IC Design)
Broadcom Communications Technologies Pvt Ltd.

Acknowledgements

It gives me immense pleasure to express my gratitude towards people who have been a part of my internship journey. I consider my internship opportunity at Broadcom as a great milestone in my career. I am highly thankful to Mr. Rajesh Rajan (Project Manager) for his constant support and encouragement. I would like to express deep sense of gratitude to my project guide Mr. Prasad Rao Kotra for his invaluable guidance and appreciation. I am thankful to all my team members for providing me their needed help. I would like to thank Broadcom Communications Technologies Pvt Ltd. for providing me this opportunity to work with highly skilled and enthusiast team members. I am highly indebted to Dr. Niranjan M. Devashrayee, Institute of Technology, Nirma University for his timely guidance and support. I am thankful to Dr. P. N Tekwani, Head of Electrical Engineering Department for allowing me to undertake this thesis work. Last, but not the least, I am thankful to my Parents for being a great source of motivation in my life.

- Keyuri Patel
(14MECV17)

Abstract

Low Power technology has become popular due to the increasing trend of portable applications available in the market. In the past few years design approach for complex SoC has changed to a great extent. Power, Performance and Area have always been the design constraints for IC designers with power gaining utmost importance in low power applications. Various methodologies like design reuse and design IP are being adopted for complex SoC applications. Early Power Analysis at RTL Design stage thus become important to ensure Power consumption does not cross the Power budget. My team in Broadcom is working on design and verification of Low Power LTE Modem based SoC application. Power Analysis of all blocks on Modem IP is being carried out using PowerArtist tool from Ansys. Power Analysis flow at RTL level for various blocks on SoC has been described in this thesis. Methodology adopted for Power Analysis at RTL development stage differs greatly from gate-level in terms of accuracy, power analysis iteration time and power reduction opportunities. For large design, time required for Power Analysis at gate-level would be very high with very high accuracy as design would be in final stage with timing analysis. But Power reduction opportunities would be very less post-synthesis and also time to market has to be taken care. Thus, Power Analysis approach at RTL level using PowerArtist (Design for Power) tool which incorporates PACE (PowerArtist Calibrator and Estimator) technology has been adopted. PACE technology, which creates a feedback loop between physical and RTL design processes to help ensure accuracy and consistency of RTL power estimates. PACE looks at key design elements, such as clock tree topology, wire capacitance and cell selection distributions, to create design and technology aware models. Average Power Analysis for complex SoC helps to debug power hotspots in design. Average Power Analysis with RTL simulation approach has been described in detail in this thesis.

Contents

List of Tables

List of Figures

List of Acronyms

SoC System-On-Chip

PACE PowerArtist Calibrator and Estimator

RTL Register Transfer Logic

CMOS Complementary MOSFET

BiCMOS Bipolar Complementary MOSFET

FSDB Fast Signal DataBase

VCD Value Change Dump

SPEF Standard Parasitic Extraction Format

IOT Internet of Things

LTE Long Term Evolution

IP Intellectual Property

CPU Central Processing Unit

GPU Graphics Processing Unit

DFT Design for Testability

STA Static Timing Analysis

PMU Power Management Unit

CPF Common Power Format

UPF Unified Power Format

PST Power State Table

DUT Design Under Test

CTS Clock Tree Synthesis

HVT High Voltage Threshold

SVT Standard Voltage Thresold

LVT Low Voltage Thresold

GAF Global Activity File

TDP Thermal Design Power

EDA Electronic Design Automation

Chapter 1

Introduction

1.1 Importance of Low Power applications

In the past, due to high degree of process complexity and the exorbitant costs involved, low-power circuit design and applications involving Complementary MOSFET (CMOS) and Bipolar Complementary MOSFET (BiCMOS) technologies were used only in applications where very low power dissipation was absolutely essential, such as wrist watches, pocket calculators, pacemakers, and some integrated sensors. However, low-power design is becoming the norm for all high-performance applications, as power is the most important single design constraint. Although designers have different reasons for lowering power consumption, depending on the target application, minimizing the overall power dissipation in a system has become a high priority.

One of the most important reasons for this trend is the advent of portable systems. As the "on the move with anyone, anytime, and anywhere" era becomes a reality, portability becomes an essential feature of the electronic systems interfacing with non-electronic systems, emphasizing efficient use of energy as a major design objective.

Portability has numerous factors to be considered. First, the size and weight of the battery pack is fundamental. It is not at all practical for a portable system to have an unreasonably heavy battery pack and itrestricts the amount of battery power that can be loaded at any one instance of

time. Second, the convenience of using a portable system relies heavily on its recharging interval. A system that requires frequent recharging is inconvenient and hence limits the user's overall satisfaction in using the product.

Although the battery technology has improved over the past few years, its capacity has only managed to increase by a factor of two to four in the last 30 years. To illustrate the importance of low-power design, consider a future portable multimedia terminal that supports high-bandwidth wireless communication; bi-directional motion video; high-quality audio, speech, and pen-based inputs and full texts/graphics. The power of such a terminal when implemented using off-the-shelf components not designed for low power is projected to reach approximately 40 W. Based on the current Nickel-Cadmium (NiCd) battery technology,which offers a capacity of 20 W-hour/pound, a 20-pound battery pack is required to stretch the recharge interval to 10 hours[1]. Even with new battery technologies, such as the rechargeable lithium or polymers,battery capacity is not expected to improve by more than 30 to 40 % over the next 5 years. Hence, in the absence of low-power design techniques, future portable products will have either unreasonably heavy battery packs or a very short battery life. The issue of power also embraces reliability and the cost of manufacturing non portable high-end applications. The rapidly increasing packing density, clock frequency and computational power of microprocessors have inevitably resulted in rising power dissipation. The trends relating to the power consumption of microprocessors indicate that power has increased almost linearly with area-frequency product over the years. For example, the DEC21164, which has a die area of 3 cm^2 and runs on a 300-MHz clock frequency, dissipates as much as 50 W of power [1]. Such high power consumption requires expensive packaging and cooling techniques given that insufficient cooling leads to high operating temperatures, which tend to exacerbate several silicon failure mechanisms. To maintain the reliability of their products, and avoid expensive packaging and cooling techniques, manufacturers are now under strong pressure to control, if not reduce, the power dissipation of their products [1].

Finally, due to the increasing percentage of electrical energy usage for computing and communication in the modern workplace, low-power design

is in line with the increasing global awareness of environmental concerns. As a result, power has emerged as one of the most important design and performance parameters for integrated circuits. Only a few years ago, the power dissipation of a circuit was of secondary importance to such design issues as performance and area. The performance of a digital system is usually measured only in terms of the number of instructions it can carry out in a given amount of time,that is, its throughput. The area required to implement a circuit is also important as it is directly related to the fabrication cost of the chip. Larger die areas lead to more expensive packaging and lower fabrication yield. Both effects translate to higher cost. Because the performance of a system is usually improved at the expense of silicon area, a major task for integrated chip (IC) designers in the past was to achieve an optimal balance between these two often-conflicting objectives[1]. Now, with the rising importance of power, this balance is no longer sufficient. Among the products requiring low-power management are the following :

- Consumer, wireless, and handheld devices: cell phones, personal digital assistants (PDAs), MP3 players, global positioning system (GPS) receivers, and digital cameras

- Home electronics: game consoles for DVD/VCR players, digital media recorders, cable and satellite television set-top boxes, and network and telecom devices

- Tethered electronics such as servers, routers, and other products bound by packaging costs, cooling costs, and Energy Star requirements supporting the Green movement to combat global warming

Today, IC designers must design circuits with low-power dissipation without severely compromising the circuits' performance.

1.2 Necessity of Early Power Analysis at RTL level

- The Register-transfer level (RTL) input description of the design allows the designer to make optimizations and trade-offs very early in the design flow.
- The presence of functional blocks in an RTL description makes the complexity of the architectural design much more manageable even for large chips due to its large enough granularity than the corresponding gate- or circuit-level descriptions.

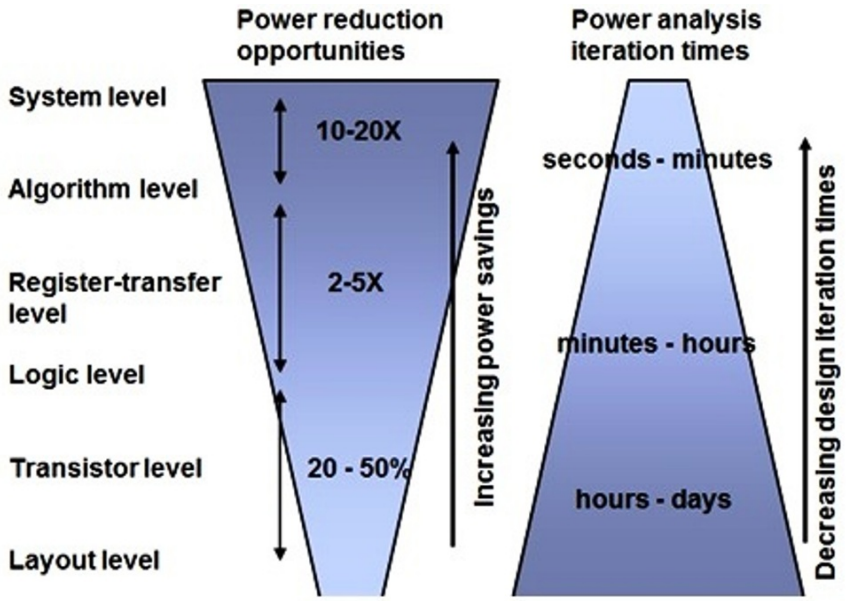

Figure 1.1: Power Analysis at different levels

It is advisable to measure power continuously during RTL development so as to ensure that any design changes do not violate the allocated power budget. It helps to know immediately whether the RTL optimizations indeed save power.

It is well known that power consumed in a CMOS circuit can be calculated as the summation of dynamic power, leakage power and a small amount of instantaneous or short circuit power. Short circuit power is the power dissipation when both NMOS and PMOS transistors are ON at the same time. By controlling the slew rate, this power is minimized. Dynamic power consumed by a circuit is given as :

$$P_{dynamic} = \frac{1}{2} \times C \times V^2 \times f \times a \tag{1.1}$$

where "C" is the capacitance, "V" is the operating voltage, "f" is the clock frequency, and "a" is the activity or percentage of the circuit toggling in a clock cycle. Even though the equation seems easy,using this equation at the RTL is tricky. RTL designers need a lot of information regarding how the RTL will be implemented to estimate power correctly. Leakage power can be calculated as:

$$P_{leak} = V \times I_{leak} \tag{1.2}$$

where "V" is the operating voltage and "I_{leak}" is the leakage current. Leakage current is calculated for nodes that are not switching. Calculating which nodes are static can be tricky at the RTL. Total Power consumed in a circuit can be given as :

$$P = C_L V_{dd} V_{swing} f + V_{dd} Q_{sc} f + V_{dd} I_{leak} + V_{dd} I_{through} \tag{1.3}$$

V_{dd} represents the supply voltage, I_{dd} represents the static current drawn from the supply, C_L represents the load capacitance, and f represents the switching frequency. The VI term represents the static, or DC power consumption, while the CV^2f term represents the dynamic power consumption. In the more detailed version, V_{swing} represents the signal voltage swing (which for CMOS is usually equal to V_{dd}),Q_{sc} represents the charge consumed due to the short - circuit momentary current (also known as crowbar current) drawn from the supply during switching events, I_{leak} represents the parasitic leakage current, and $I_{through}$ represents the (by design) quiescent static current. The first two terms of this equation represent the dynamic power consumption,

while the latter two represent the static power consumption.

- **Objective :** RTL level enables power related design tradeoffs, whereas gate-level is best suited for power signoff. RTL and gate-level power analysis for the same design demonstrates how RTL enables functional visibility and debug. The adder as an example is identified as a power hotspot at RTL. In the RTL view, the designer can explore shutting off the adder by tracing upstream/downstream. In contrast, the same adder is lost in the hundreds of gates after synthesis.

- **Performance :** RTL power runs an order of magnitude faster versus gates. For example, a mobile graphics processor team took 22 minutes to run RTL power analysis for a design block that took 20 hours to get to gate-level power numbers, dominated by synthesis and gate-level simulation overhead.

- **Activity :** Activity has a first-order impact on power, and power consumption management must adequately represent this. RTL simulations are easy to bring up and can provide wide coverage across multiple modes of operation. Gate simulations on the contrary are harder to bring up and are often available too late - sometimes after tapeout.

- **Accuracy :** RTL power is typically within 15% to 20% of post-layout power numbers. It is possible to model synthesis and physical effects adequately at RTL for consistent accuracy versus gates, without compromising the runtime benefit. For instance, clock network and wire capacitance both have a large impact on power with very little information available at RTL. However, calibrating and characterizing representative layouts to generate RTL models for these can significantly improve RTL power predictability.

	RTL Power	Gate-level Power
Goal	Design trade-offs, budgets	Sign-off
Performance	Typically minutes	>10X slower
Stimulus	Easier setup, early	Complex setup, too late
Root cause analysis	Ease of power debug	Difficult to view functional intent

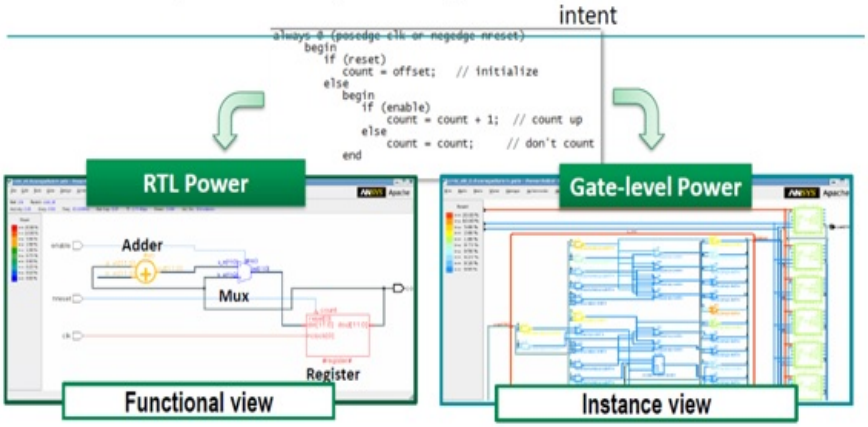

Figure 1.2: RTL vs Gate-level Power

(Image Courtsy: *PowerArtist-Enabling RTL Design for Power, ANSYS)*

1.3 Peak Power and Average Power

Designers need to analyze their designs for peak power and average power. Average power consumption should be below the power requirement set by the market. The power requirement is often specified as the Thermal Design Power (Thermal Design Power (TDP)) constraint. Package selection is also based on the average power consumption. So designers want to get this value early in their design cycle and make sure it is correlated with the final implementation. Peak power typically happens when the design is subject to maximum activity. The peak power value is used to design the power supply. Peak power is also used to determine the power integrity. Higher peak power can cause excessive an IR drop, which will result in poor performance. Peak power can be handled by adding decoupling capacitances to the design. These capacitances store a charge during normal operation and supply the additional demand during peak operation. Designers also need to pay attention to the difference between peak power and average power. Too large

a difference will require them to add a number of decoupling capacitances, which can increase the area and the leakage current. Too small a difference may point to missed optimization opportunities.

Today, System-On-Chip (SoC)s operate across multiple modes with different IPs active at different modes. As a result, the difference between peak power and average power can be very large. Hence, power analysis at the RTL is important to identify and fix many of these issues.

1.4 Outline of thesis

This thesis is organised as follows.

In chapter 2, Low Power SoC Design Issues is described.

In chapter 3, Methodology of RTL level Power Analysis forSoC using PowerArtist tool is described.

In chapter 4, RTL Power Analysis flow setup is described.

In chapter 5, Analyzing Power Reports and Power reduction opportunities is described.

In chapter 6, Power Analysis Challenges for Complex SoC Design is described.

Chapter 2

Low Power SoC Design Issues

2.1 Integration of complex blocks on SoC

Design complexity of SoC is increasing with technology as more functionality is being added on SoC.At same time, pressure is building up to reduce operating and standby power. Today the market is focused on reducing power in wide spectrum of SoCs from CPUs, GPUs and Mobile not just Internet of Things (IOT)/wearable SoCs. Battery powered SoCs require much more aggressive power reduction techniques. All of these SoCs today have a large number of power and clock domains and they have hundreds of IPs and memories.

These types of multi-power domain SoCs are complex and present new integration challenges because many blocks have different operating modes at different voltages, different clock period and duty cycles of each block being awake, asleep or in shutdown mode. We must pay more attention to applications that dictate modes of operations, power and battery life requirements.

While power gating and other techniques are effective today, these techniques may not be enough for many IOT low-power SoCs. We have to think in terms of design- for- power in terms of energy consumptions at all levels of design including overall system/ application, architectural, power management, RTL, Design for Testability (DFT), and physical implementation in addition to technology and process advancements.

Power management units (Power Management Unit (PMU)), a mixed signal block and digital controller are very important blocks of a low power SoC. In addition to power gate digital block with a switch, but for battery powered designs it is possible to power gate the analog portions. In these designs, it is important to deal with issues such as stability and the time necessary to reach a stable state. This requires an analog circuit designer and has to be done much more carefully than for digital design.

Success with multi-voltage SoCs requires careful planning and a robust methodology and toolset for all stages of the implementation. One need proper tools for the analog and digital part and one need to be conscious of their flow.

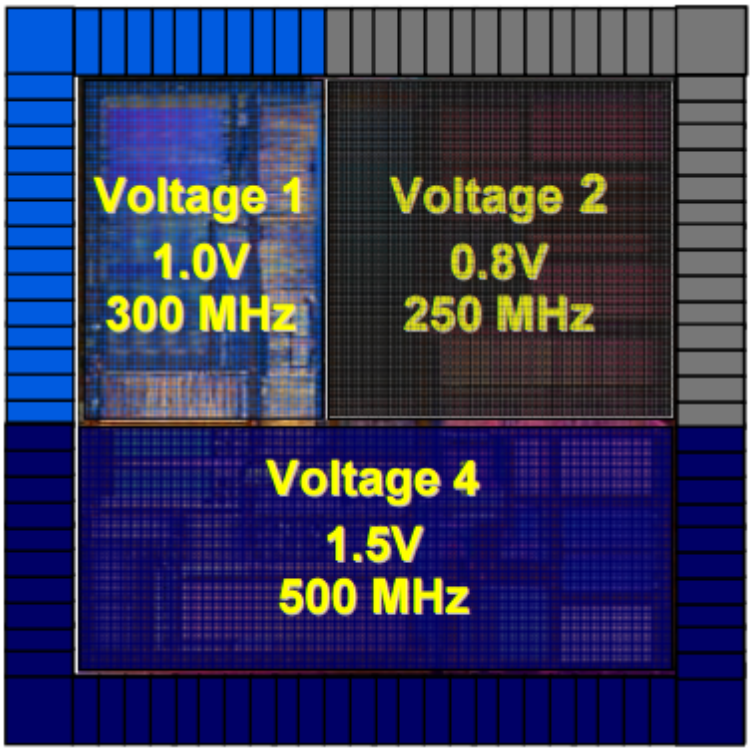

Figure 2.1: Multi-voltage blocks on SoC

2.2 System Requirements and RTL development

A detailed requirement document is needed, including: Always-on (AON) block; power domains; power gating; clocking; system level and block level power options with timing; and a description of how power domains interact with each other. This information is used by firmware/software RTL designers and by verification and physical implementation teams. Also, one must define power or ground switches for each domain. There may be a required sequence for powering up the design in order to avoid deadlock.

Also we need an explicit power sequence for each of the power domains to come up in a well-defined order that assures correct function. And in fact, some IP may require a specific power up sequence. Companies have front-end tools and flows to capture power intent using Unified Power Format (UPF)/Common Power Format (CPF) including Power State Tables (Power State Table (PST)) to define all possible power state combinations; the PST captures the valid interaction between the different power domains so that tools can determine the optimal buffering strategy [8].

Tools are being developed to address power at the RTL level. This will help identify coarse & fine grain power reduction methods for implementation. The power intent UPF is described at RTL for the logic design. The UPF file is updated during the synthesis flow, and updated again during the place and route process.

2.3 Verification

Verification of Power-Intent (UPF/ CPF) using these fine-grained techniques requires close interaction between the verification team, System designer, RTL designers and circuit/ physical implementation team members. In battery powered SoCs, all low level assumptions such as intermittent on/off timing have to be verified at the RTL level and in some cases, must use circuit simulations. Automated solutions are not satisfactory since the analysis must cross all phases - RTL design, Synthesis, and dynamic operation. The addition of low-power mode with multiple power domains creates corner

cases that are not detected in functional simulations. This is where formal approach is the one way to ensure their DUT will be free of bugs, as well as unwanted 'Xs.' Detailed SPICE simulation of the whole chip is not a viable option for these SoCs. Simulation times would be unreasonably increased due to the complexity of power controller block. A detailed simulation of this block would require cycling through multiple power transitions, which in turn requires carefully chosen input vectors for each stage of the design, exponentially increasing simulation times [2].

This is a critical step to ensure that the power constraints are defined properly and to avoid later surprises. Power-on/off is a complex issue because analog blocks require technology-dependent stabilization and lock times order tens of milliseconds. Additionally, all power domains have their own reset sequence and timing asSoCiated with that. In particular, we need to make sure that all power domains are completely powered up before issuing reset. Also, the CPU/controller may need to wait until the rest of the chip is powered up before booting.

2.4 Use of Formal Tools

Functional Verification of power sequencing is timing consuming and add to schedule. This is one task where formal verification tools will help. Formal tools (MVRC or Jasper/ Cadence) can be used for verifying properties that simulation will take long time to verify such as reset time of all blocks. Formal tool help determine the logic path that takes the longest time to reset [8].

2.5 Physical implementation

UPF/ CPF based power - intent implementation flow is well established for netlist updates which include insertion of level shifters, Isolation cells etc.

2.6 Clock tree Synthesis

The clock tree synthesis engine should use the PST-based buffering solution while expanding the clock tree network across different power domains means that they have to go through level shifters. the clock tree synthesis tools will automatically insert level shifters at the appropriate places. As a general guideline all clock sources reside in always-on blocks as it is tricky to balance skew through level- shifters.

Floorplanning, power grid planning: Multiple power domains require very careful and detailed power grid planning. The power grids become more complex.

2.7 Static Timing Analysis

Multi-level voltage scaling presents a greater challenge. The questions is: which multi-corner, multi- mode voltage library to use for synthesis, place and route, and Static Timing Analysis (STA)? With multiple voltage, libraries may not be characterized at the exact voltage we are using, timing analysis becomes much more complex. The end result is that the design should meet timing and power requirements for all the mode/corner scenarios. Designers must ensure that the correct level shifters, retention cells, and other design elements have been accurately placed for each of the different power domains, also verify bulk and well connections at the transistor level. There are tools from Electronic Design Automation (EDA) companies help automate these checks [2].

2.8 Low Power SoC Issues

There are many design methodology issues requiring automation, but today these are resolved using custom script or manual intervention.

2.8.1 Rush current management

In multi- Voltage domain designs, blocks are powered through on-chip switches or from different external voltage pins or on Chip Regulators LDOs. During power up sequence there is a large current peak for short duration which cause voltage drop that could corrupt retention registers. It is very important to reduce peak current, through decoupling Caps or adding appropriate delay (daisy chaining) in turn-on signals or clocks for these blocks. One have to carefully design size of Switches to reduce voltage drop. Over-size Switch will increase standby leakage power.

2.8.2 Power estimation issues

The biggest gap in our methodology is that we don't have accurate early power consumption estimation at the system partitioning stage. RTL power estimation tools are just starting to be used. There is little data available on correlation of RTL power with measured silicon results and micro- amp level accuracy especially for battery powered SoCs. These types of tools will have high impact on design flow improvement.

Today most of the resources are assigned to verification including Spice simulation for mixed signal blocks, functional and power-intent verification, power-intenttiming interaction and power sequencing at chip and blocks level.

Methodologies are being updated to meet demands of low-power multi-voltage designs, and using the latest tools and flow scripts that incorporate best practices can make the difference between taping out the design on time and within specification.

Chapter 3

Methodology of RTL Power Analysis for SoC using PowerArtist tool

3.1 Introduction to PowerArtist tool

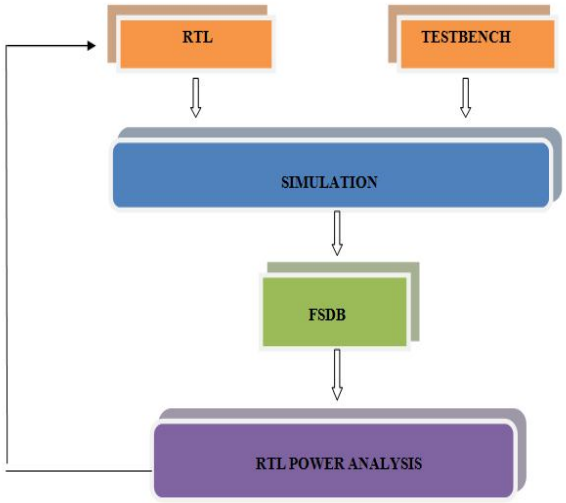

Figure 3.1: Typical SoC RTL Power Flow

PowerArtist enables RTL-to-GDS design for power methodology by providing early RTL power estimation and analysis-driven power reduction capabilities. RTL designers working on a variety of applications, from mobile to CPU to networking to automotive ICs, use PowerArtist to optimize designs throughout the development cycle.

It is TCL based tool for RTL level and Gate level power analysis. In our project, we are using PowerArtist for power estimation of various blocks on modem based SoC at RTL level. Full chip Power Analysis helps to understand the average dynamic power consumption based on appropriate usecase selected for simulation.

3.2 RTL based Power Analysis using PowerArtist

As shown in Figure 3.2, main steps in this power analysis flow are controlled by the following commands:

1. **Elaborate**

   Compiles the HDL design description into an internal binary format called the scenario file. The scenario file is a binary representation of the hierarchical micro architectural netlist for the design.

2. **GenerateActivityWaveforms**

   Analyzes the activity file and produces waveform files representing the activity inthe design.

3. **CalculateFlopClockActivity**

   Monitors the activity at the clock pins of registers.

4. **CalculatePower**

   a. analysis_type average - performs an average power analysis.
   b. analysis_typetime_based - performs a time-based power analysis.

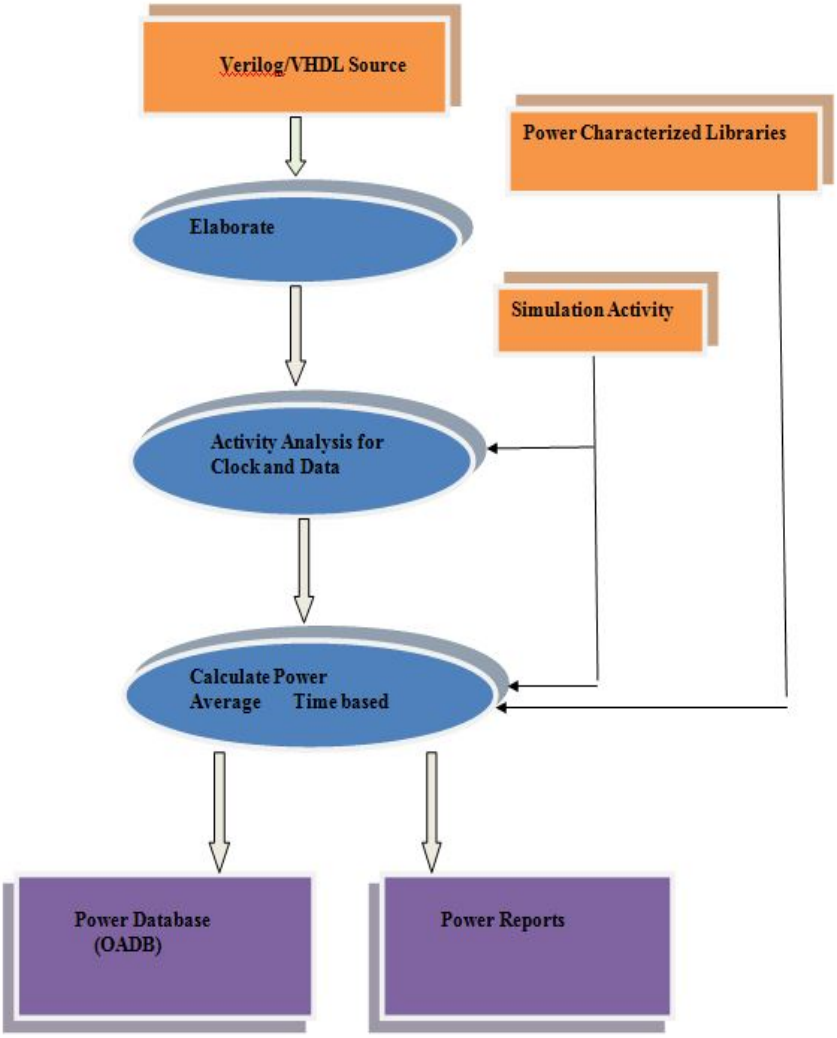

Figure 3.2: Main steps involved in RTL Power Analysis

PowerArtist is a mixed-level power analysis tool that analyzes RTL designs, gate-level designs, and combination of both [3]. It take inputs in the form of design description, technology libraries, and operating conditions such as power supply values and external loading, and toggling activities through simulation. It produces a detailed report which states the amount of power consumed by the various instances/portion of the design along with additional power wastage information which helps to detect hotspots in the design.

Architecture of Power Analysis tool PowerArtist is shown in Figure 3.3. Its front end includes a language parser and inference engine. Their function is to read the design description in Verilog, VHDL or mixed Verilog/VHDL, translate the design into an internal representation, and finally loading into the internal data-base. The power calculation engine reads the internal database, and, using the specified toggling activities through Fast Signal DataBase (FSDB) file (captured during simulation of RTL design), environmental data, and technology specific physical libraries, calculates the power for each portion/instance in the design.

The technology library describes the electrical characteristics of the circuit primitives to be employed in the design. Data, such as input capacitances, state dependent static and dynamic power consumption, logical function, and physical size, are all used during various computations leading up the final power calculation [3].

Upon encountering RTL code, PowerArtist infers the hardware that would be produced by a synthesizer for that code [3]. However, unlike a full logic synthesizer, PowerArtist does not produce a gate level netlist but it produces a micro-architectural netlist that contains inferred operators, such as multiplexers, arithmetic units, multi-bit registers and decoders which are technology unmapped. Although it is similar to the initial steps executed by a conventional gate-level synthesize, this approach results in faster execution time as it has a smaller internal database [3]. It has an ability to easily cross-reference the original RTL code with the inferred operators. By contrast, when encountering instantiated objects, such as compiled memories or gate-level primitives, PowerArtist passes these objects directly to the

internal database [3].

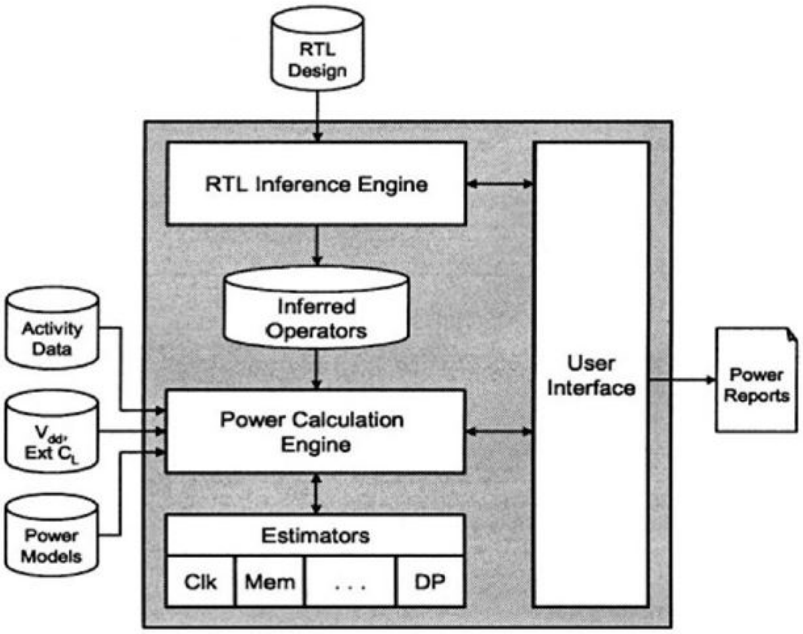

Figure 3.3: Power Analysis tool architecture

(Image Courtsy: *Massoud Pedram & Jan M. Rabaey,"Power Aware Design Methodologies", page no. 439*)

Once the inferencing step is done and the internal data-base is loaded, the design is ready for analysis [3]. The calculation engine loads the database, along with the toggling activity (which can come from a simulation trace, in the form of a value change dump (Value Change Dump (VCD)) file, Fast Signal Database (FSDB) or from a vectorless activity specification), technology data, and environmental data. For gate-level power analyses, the calculations are relatively straightforward: for each instance, PowerArtist determines the particular stimulus from the activity data, looks up the power characteristics for that stimulus in the technology specific physical library, and computes the instance's power using the specified operating conditions through corner cases (ff126125 - Fast nMOS, Fast pMOS, 1.26 V Supply voltage and 125 °C Temperature).

For RTL Power Analyses, instead of processing gate-level instances, the Power calculation engine calculates power for each inferred instance in the

design. This is accomplished by elaborating a parameterized model for each inferred instance. The elaboration process involves evaluating a built-in parameterized power equation for each instance utilizing power information [3] from the technology specific physical library along with the toggling activity and operating conditions. Each inferred operator has its own unique power equation, thus enabling PowerArtist to separately calculate power for the different design structures and objects that are found in the RTL. For example, datapath operators are inferred and evaluated separately from control and clock operators [3]. This divide and conquer approach enables faster and more accurate power calculations (more accurate than the "one size fits all" algorithm common to gate-level tools) along with a reporting format that is closely linked to the RTL source code. For example, clock power, input/output buffer (I/O) power, register power, and random logic power [3] are all included in the power report.

Various reporting styles and format are available for producing power reports in PowerArtist tool. Given the objective of writing power- efficient RTL code, effective analysis, and debugging tools, capable of pointing power problem areas, are critical for identifying power reduction opportunities.

PowerArtist is primarily used for two different purposes: power reduction and power analysis. The objective of power reduction is to minimize power consumption by writing power-efficient RTL code and by providing early visibility into the design's power characteristics. The objective of power analysis is to check or verify that the design, whether at the RTL or gate level, is within an acceptable power consumption limit [3].

Producing power- efficient RTL code requires the ability to identify the amount and source of power consumption along with its underlying causes [3]. Although many of these factors, such as V_{dd} or C_L, are fixed by either technology or environmental dictates, others, such as nodal switching frequency, can be affected by coding styles.

In a power reduction methodology, PowerArtist is used to estimate power as the RTL code is written, thus enabling the designer to quickly understand the impact of design decisions and to optimize the code during the design creation process, which is the essence of low-power design [3]. If a simula-

tion test bench is available, then the design is simulated to collect toggling activities for the subsequent power calculation. If a simulation test bench is not available, then PowerArtist's vectorless activity function is used to generate activity and state information for use in the power calculations [3]. Although the accuracy of the resulting calculations is much better with simulated data, vectorless activity specification can be used to determine which of several coding alternatives will consume the least power when simulation data is not yet available [3].

Conventional Code:
64x32 Register File using Flip-Flops

```
input web, oe,clk;
input [31:0] di;
input [5:0] aadr, badr;
output [31:0] do;

// define storage array
reg[31:0] array [63:0];
reg[31:0] do;

// Write Cycle: edge trigger
//              implies flops
always @ (posedge clk) begin
    if (web == 0) array[aadr] = di;
end

// Read Cycle - a sync read
always @ (badr or oe) begin
    if (oe == 1) do = array[badr];
end
```

Low Power Code:
64x32 Register File using Latches

```
input web, oe,clk;  // clk not needed
input [31:0] di;
input [5:0] aadr, badr;
output [31:0] do;

// define storage array
reg[31:0] array [63:0];
reg[31:0] do;

// Write Cycle: level trigger
//              implies latches
always @ (aadr or web or di) begin
    if (web == 0) array[aadr] = di;
end

// Read Cycle - a sync read
always @ (badr or oe) begin
    if (oe == 1) do = array[badr];
end
```

Figure 3.4: Register file RTL code

A comparative example of different RTL codes for a given function is listed in Figure 3.4. Consider a register array organized as 64 words by 32 bits. Such a storage function can be coded in several different ways, two of which are presented here. Although the amount of code is the same, the power

difference is substantial: the latch-based array consumes only about half as much power as the conventionally coded register array [3].

In this case, the power reduction principle at work is the use of latches instead of flip-flops - latches being more power-efficient than flip-flops [3]. This targets the $V_{dd}Q_{sc}f$ term in Equation (1.3) - latches, implemented with fewer transistors than flip-flops, consume less internal current; however, a more common method of power reduction is the minimization of effective switching frequencies, targeting the two frequency dependent terms in Equation (3), $C_L V_{dd} V_{swing} f$ and $V_{dd}Q_{sc}f$. Here, the concept is to reduce unwanted or unnecessary toggles, which in turn reduces the dynamic power consumption [3].

3.3 PowerArtist Inputs for Power Analysis

PowerArtist requires the following inputs to calculate power for an RTL or a gate level design:

- **RTL/Gate-level Design Files**

  Supported formats are Verilog, VHDL, System Verilog, or a mix of any of these three HDLs.

- **Simulation Activity File**

  Supported formats for activity information are FSDB, VCD, and SAIF.

- **Power Libraries**

  Characterized libraries in Liberty format (.lib).

- **Net Capacitance**
  a.Standard Parasitic Extraction Format (SPEF)
  b.Wire Load Models
  c.Pace(PowerArtist Calibration and Estimation)Technology File

- **Clock Definitions**

PowerArtist can infer the clock trees for RTL or pre-Clock Tree Synthesis (CTS) gate-level designs, and it can trace existing clock trees for post-CTS gate-level designs.

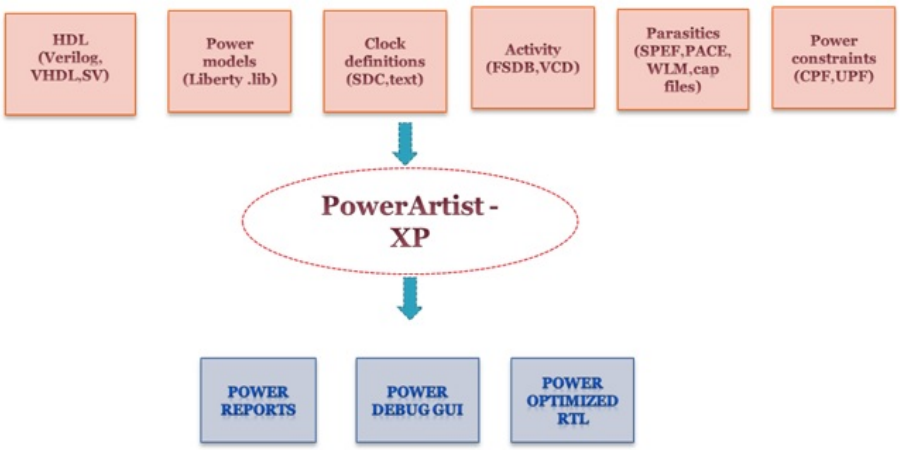

Figure 3.5: Inputs to PowerArtist for RTL Power Analysis

We need to specify the root clocks and clock buffers to infer a clock tree. To trace a clock tree, only root clock names are required.

3.4 PowerArtist shell

The PowerArtist shell (pa_shell) is a fully functional Tcl shell, from which all other PowerArtist commands must be invoked. We can use any standard Tcl command inside of this shell.

Syntax : pa_shell

This command invokes the PowerArtist shell. At the resulting pa_shell prompt, we may type any Tcl command or PowerArtist command. We will be able to execute all PowerArtist-PT commands.

pa_shell -tcl run_Elaborate.tcl

This consumes licenses at the PowerArtist level and then runs the Tcl script Elaborate.tcl. A sample Elaborate.tcl file might be:

```
source ptSourceFiles.tcl
Elaborate -scenario_file my.scn \
   -top top \
   -synlib_files {mylib.lib}
```

Figure 3.6: Sample Elaborate.tcl

Invoking the GUI from the pa_shell : pa_shell % PowerCanvas -pdb top.pdb

3.5 Preparing for Power Analysis

Before running Power Analysis, arrangements for setting net capacitance, handling designs with multiple power supplies and libraries and clock power specification must be done.

3.5.1 Estimating Net Capacitances

Accurately estimating dynamic power in the design requires good estimates of the net capacitances; therefore, before we begin our power analysis, we need to determine how we will provide net capacitance information to PowerArtist. We can use any of the following methods:

- Back-annotate capacitances using SPEF (Standard Parasitic Extraction Format) file.
- Specify a default output load capacitance value using the -default_output _load option to CalculatePower command.

- Specify a PowerArtist Calibrator and Estimator (PACE) technology file (using the - power_tech_file option) to estimate net capacitance for RTL, mixed RTL/gate or gate-level designs. PACE files are also used for clock distribution network modeling.

- Allow PowerArtist to use wire load models to estimate signal capacitances. This can be used for either RTL and mixed-RTL and gate or for pure gate-level netlists. We can set the wire load models using Tcl commands.

SetWireLoadModel: This command defines the wire load models for instances and nets.

***SetWireLoadModel* -name model_name -library lib_name -instance inst_name(s) -net net_name(s) -scaling_factor factor**

Arguments :

-name : Specifies the name of the wire load model.

-library lib_name : The logical library name which should be searched for the wire load model given with the -name option. If we do not specify a library name, the estimators will use the library specified using the -wireload_library command-line option. If that is not found, then it will search all of libraries specified using the -synlib_files option for a wire load model.

-instance inst_name(s) : Specifies a hierarchical instance name or a Tcl list of instances. We may use wildcards. For each hierarchical instance you specify, the power analyzers will set theuser wire load model on all of the instances that are in that hierarchy. All of thenets that are fully covered by this hierarchical instance will be given the samevalue.

-net net_name(s) : Sets the wire load model on the specified nets only. We

can specify a single hierarchical net name or a Tcl list of nets. We can use wild cards.

-scaling_factor factor : The factor by which the computed capacitance value returned from the wire load model should be scaled.

3.5.2 Handling Multiple Libraries with different No-minal Voltages

If we specify multiple libraries and if there are conflicting values of voltage in the first library rail of the different libraries, PowerArtist will generate a critical warning, if it identifies conflicting values of nominal voltage of different libraries.

3.5.3 Handling Designs with Multiple Power Supplies

Many designs use more than one power supply (or voltage rail). Common examples include:

- A design with two voltage islands: one supply for the core of the chip and the other for the I/Os.
- A design that uses power gating. The supplies to the power domains will need to be explicitly turned on and off due to the use of sleep signals to put various power domains of chip in a standby mode.

PowerArtist provides general support for multiple power supplies by:

- Allowing to define additional power supplies, which are referred to as virtual supplies.
- Allowing to set virtual power supplies on an instance by instance basis.

For a library with nom_voltage and operating conditions, the characterization voltage is considered as the nom_voltage and the estimation voltage is picked up from the default operating conditions. Similarly, if a library

defines multiple supplies using the power supply attribute as:

```
nom_voltage : 1;
power_supply()
default_power_rail :VDDlow;
power_rail(VDDhigh, 1.32) ;
power_rail(VDDlow, 1.32) ;
power_rail(VSShigh, 0.00) ;
power_rail(VSSlow, 0.00) ;

operating_conditions("BEST")
process : 0.70;
temperature : 110;
tree_type : balanced_tree;
power_rail(VDDhigh, 1.00) ;
power_rail(VDDlow, 1.00) ;
power_rail(VSShigh, 0.00) ;
power_rail(VSSlow, 0.00) ;

default_operating_conditions : "Best";
```

then by the fact that the default_operating_conditions have selected "Best", the estimation voltages for the supplies are taken as:

VDDhigh - 1V
VDDlow - 1V
VSShigh - 0V
VSSlow - 0V

and the power numbers are derated accordingly, based on the supply voltages defined in the power_supply construct. To get the dynamic power, PowerArtist derates the dynamic energy number by the square of the estimation voltage divided by square of the characterization voltage. Similarly,

PowerArtist derates leakage power by the estimation voltage divided by the characterization voltage.

The Liberty rail_connection attribute is used to specify the power rails to which a cell is tied[7]. Power for an instance of the cell will be derived from those power rails only.

3.5.4 Leakage Power Modeling

Accurate leakage power estimation requires the Liberty leakage power model to be associated with power and ground pin.
The following is an example of such a leakage power model:

```
library (...)
...
voltage_map(VDD2, 0.9);
voltage_map(VDD1, 1.2);
voltage_map(VSS, 0.0);
cell (XYZ)
...
pg_pin (PVDD)
voltage_name : VDD1;
pg_type :primary_power;

pg_pin (PVSS)
voltage_name : VSS;
pg_type :primary_ground;

...
leakage_power ()
value : P;
related_pg_pin : PVDD;
```

...

If the power and ground pin association is not specified, then such leakage power models get associated with the first power supply of the library. In the above example, in the absence of related_pg_pin attribute, the leakage_power model will get associated with the first power supply, which is VDD2.

3.5.5 Technology Libraries for Power Analysis

A technology library is a text file that contains four kinds of information about the ASIC technology:

- **Structural information**
  Describes each cell's connectivity to the outside world, including cell, bus, and pin descriptions.

- **Functional information**
  Describes the logical function of every output pin of every cell so that the tool can map the logic of a design to the actual ASIC technology you are using. Cells that do not have a function described in the technology library are left untouched during optimization.

- **Timing information**
  Describes the parameters for pin-to-pin timing relationships and delay calculation for each cell in the library. This information ensures accurate timing analysis and timing optimization of a design.

- **Environmental information**
  Describes the manufacturing process, operating temperature, supply voltage variations, and design layout, all of which directly affect the efficiency of every design. These variables and their effects are defined in the environment descriptions of the technology library. Environment

descriptions include interconnect wire areas; wire capacitance and resistance; and the scaling factors for variations in process, temperature, and voltage.

Generally ff126125, ff118125 and tt110025 corner cases are used for worst case and typical operating conditions. So physical libraries for standard cells, macros or memories corresponding to these corner cases should be choosen and given as input to PowerArtist tool.

3.5.6 Running RTL Mixed-Vt Power Analysis

Using cells characterized for different threshold voltages (Vt) is a critical way to control leakage power in low-power designs. Many gate-level synthesis and physical design tools have the ability to optimize the design to reduce leakage power by replacing regular threshold voltages cells with cells that have a higher threshold voltage. This optimization is typically done to cells on paths that have positive timing slack. To perform this optimization we must either have multiple libraries characterized at a single threshold level or libraries characterized for multiple thresholds. PowerArtist allows to perform power analysis that takes mixed-Vt libraries into consideration and supports both library methodologies.

The SetVT command is used to assign different thresholds to hierarchical instances in design.

***SetVT* -mode percentage -instance top.block1 top.block2 -vt_group HVT:70 LVT:30**

In this example, all of the inferred elements that are children of top.block1 and top.block2 will have their default cells chosen from the libraries that are to be used for power analysis such that 70% of the default cells have a vt_group with the value High Voltage Threshold (HVT) and 30% have the value Low Voltage Thresold (LVT).

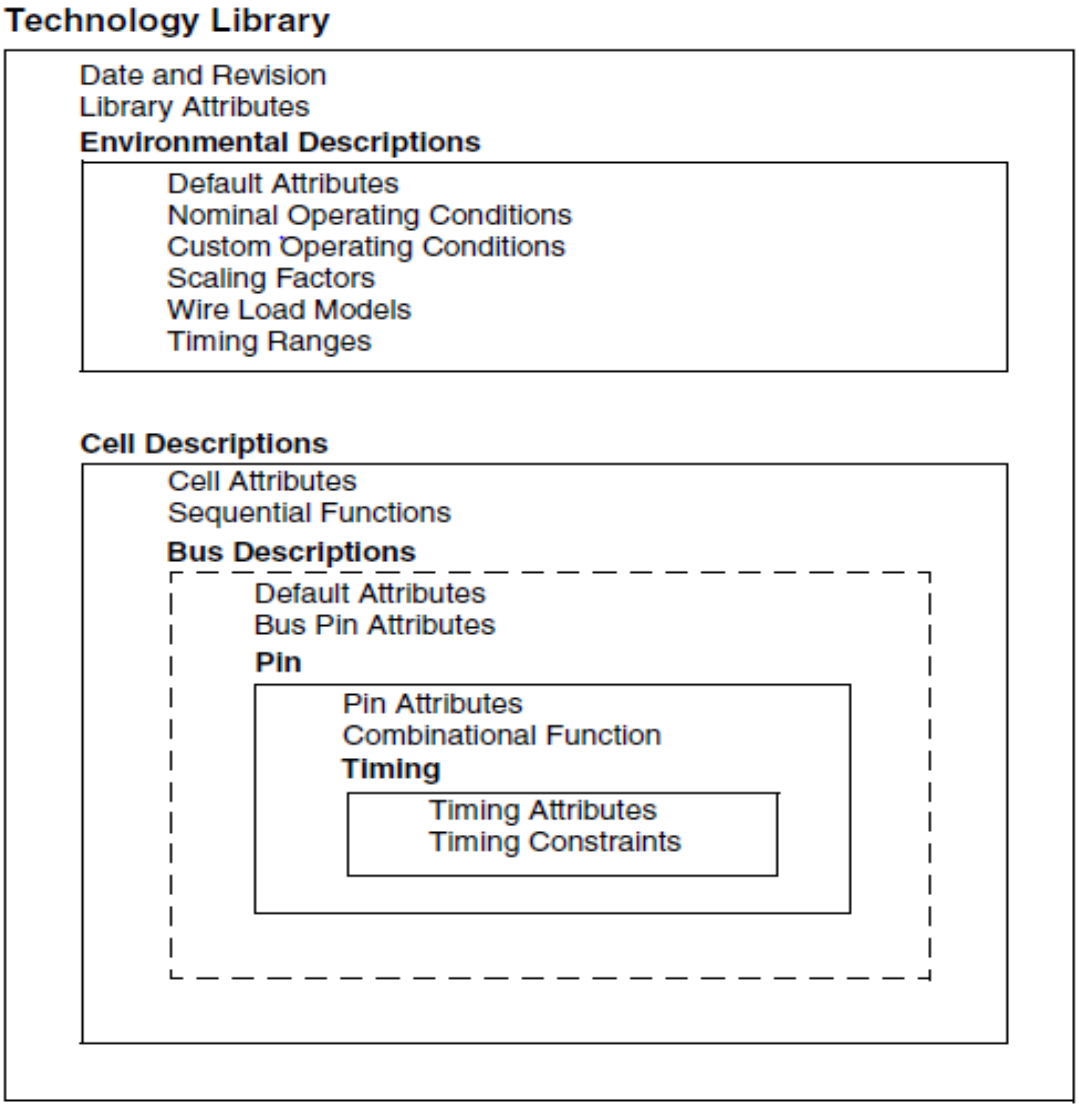

Figure 3.7: Structure of Physical library

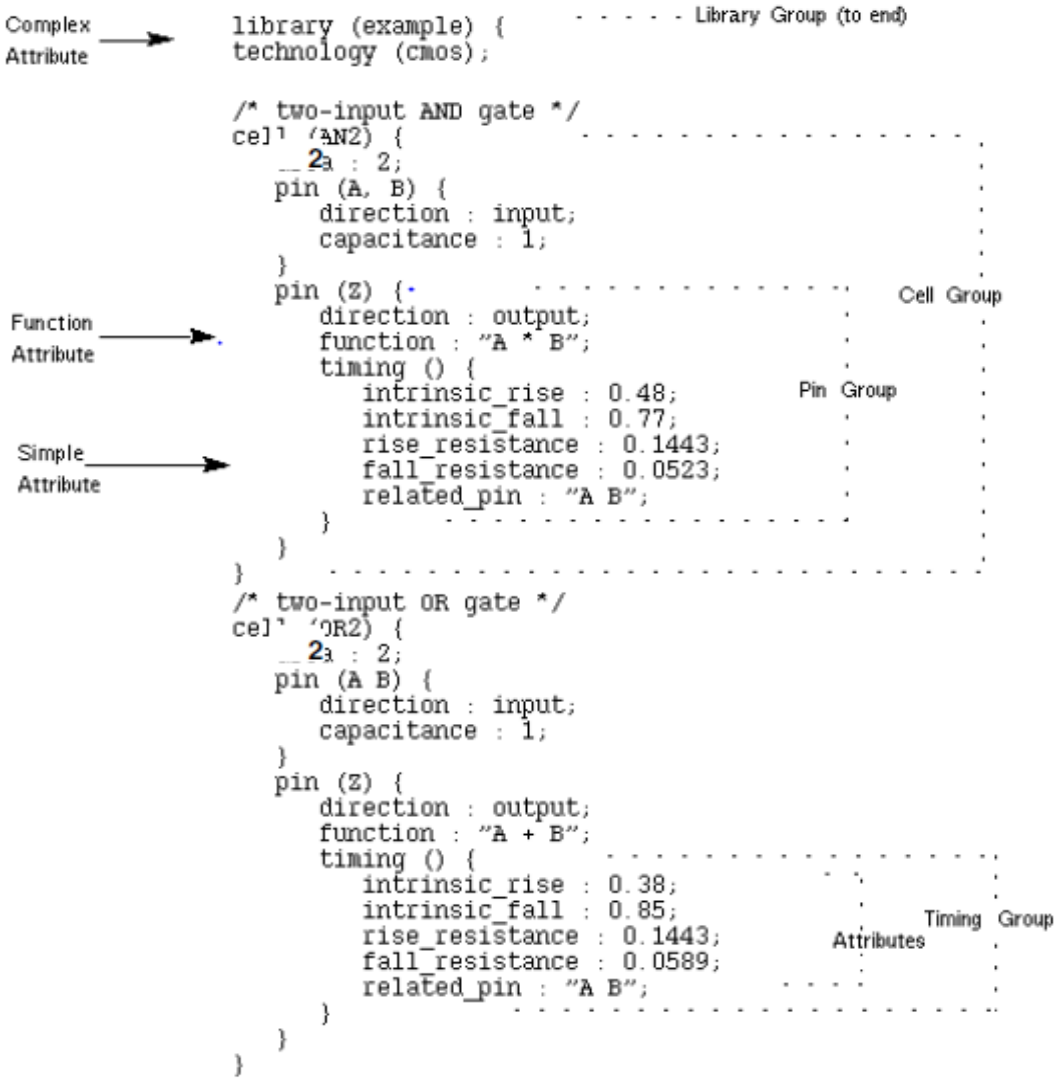

Figure 3.8: CMOS technology library file

3.6 Vector Analysis

Vector Analysis helps to understand whether the testbench is exercising the design completely. It helps to identify the appropriate timing window for power estimation.

- Simulate the design to create a simulation activity file in any of the supported formats. We are using FSDB (Fast Signal Database) format to capture simulation activities as it is most compact.
- Elaborate the design to create a PowerArtist scenario file.
- Perform vector analysis to create analysis graphs of activity over time for specific groups of instances.

Before running the power analysis step, we should verify that our testbenches have good power coverage. Often, testbenches are designed for functional verification or fault coverage, and might not represent typical operation or worst-case power situations. PowerArtist's vector analysis tool can show portions of the design where testbenches do not produce enough activity. We can choose appropriate tests or exercising more blocks in each test, so that testbenches adequately cover all situations.

There are two different types of vector analysis: clock-cycle mode and time-based mode[7]. The type of analysis is determined by the value of the following option:

-activity_waveform_graph_type activity_per_cycle|freq_per_inter

Choose activity_per_cycle for clock-cycle mode or freq_per_inter for time based mode. When we perform a vector analysis, we should be aligning the intervals so that the dominant clock edge of our choice is at the first interval start time and that each interval is some multiple of clock period.

Syntax for Vector Analysis:

GenerateActivityWaveforms -activity_file file_name
-scenario_file file_name -top_instance inst_name
-activity_waveform_clock_name clock_name
-activity_waveform_clock_edge (pos | neg | auto)
-activity_waveform_cycles_per_intervalint
-activity_waveform_graph_type activity_per_cycle
-activity_waveform_group_list group_list
-activity_waveform_log file_name
-activity_waveform_number_of_intervals int | all
-activity_waveform_start_clock_cycle int
-ptcl_output_file | -FSDB_output_file
-use_rtl_sim_data true | false

3.7 Simulation based Average Power Analysis

1. Simulate the design using appropriate testcase so that maximum of the RTL design components are covered and generate simulation data in the FSDB, IAF, or VCD formats.

2. Build a scenario file using the Elaborate command

3. Run vector analysis using the GenerateActivityWaveforms command to select appropriate timing window for Power estimation.

4. Run the CalculatePower-analysis_type average command for Power analysis.

5. Anlayze the Power reports generated in html or text format.

Running power analysis in full simulation mode is done entirely by the CalculatePower command. This command first converts simulation activity data into a global activity file (Global Activity File (GAF)) and then runs the power analysis[7].

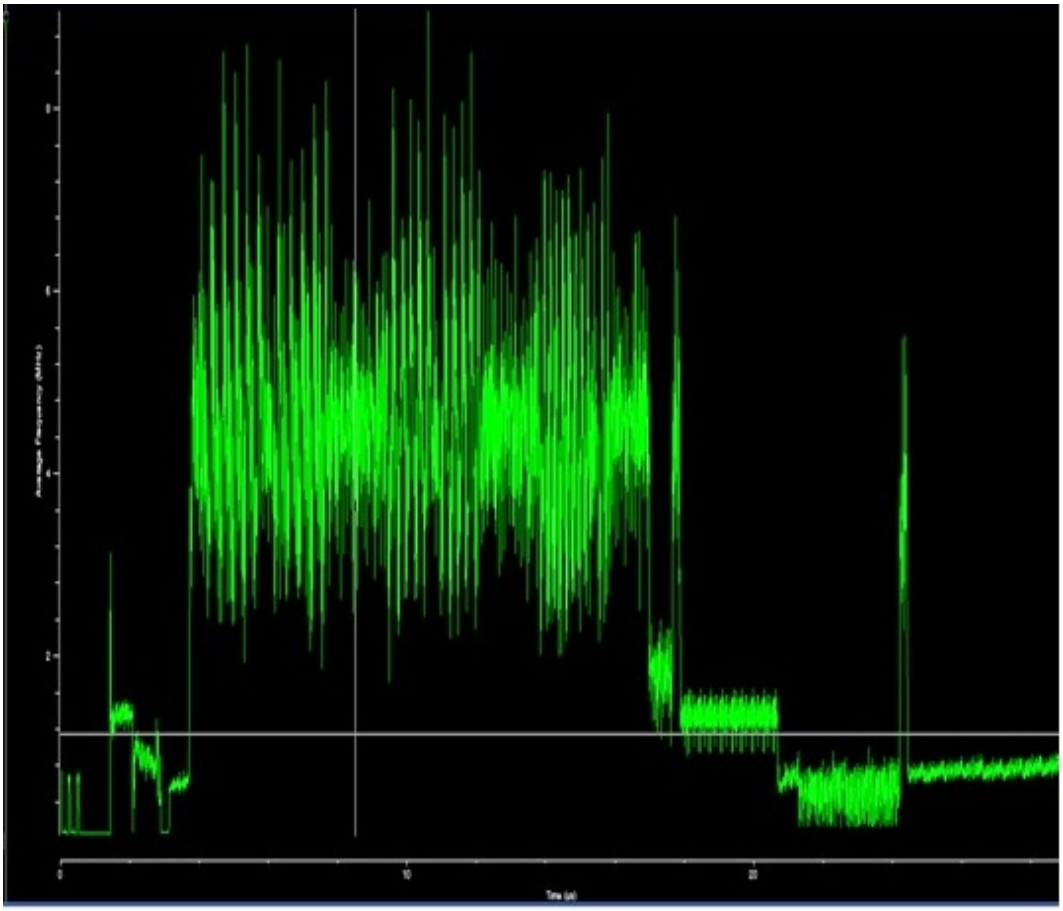

Figure 3.9: Vector Analysis graph of frequency versus time

We can control the operation of power analysis by specifying the appropriate commands in PowerArtist command file.

-activity_file simulation_file
This option specifies an input stimulus file generated by to a functional simulator run. The file may be in FSDB, VCD or IAF format. Specify either -activity_file or -vectorless_input_file for an average power analysis. CalculatePower automatically determines the type of the simulation activity file.

-GAF_file GAF_file
Specifies the name for generated GAF file.

-scenario_file scenario_file
This option specifies the scenario file you generated using the Elaborate command.

-synlib_files file_name1 file_name2 ...
Adds the specified file or Tcl list of files to the list of Liberty technology files

-top_instance top_module_name
This option specifies the instance in the simulation hierarchy that corresponds to the top-level instance in scenario file. The -top_instance is specified as a dotted name in which a dot is used as a hierarchical separator regardless of the simulator that was used (for example, testbench.corelogic_0).

-start_time & -finish_time
These options are used to set appropriate timing window for power analysis.

```
set design top
CalculatePower -analysis_type average \
   -activity_file  ../design_data/rtl_sim/activities.vcd \
   -average_report_file $design.AverageBatch.rpt \
   -average_report_options agip \
   -average_write_power_db true \
   -compress_gaf true \
   -default_output_load  3.9e-11 \
   -detailed_vertical_report true \
   -finish_time 12135580ps \
   -gaf_file $design.AverageBatch.gaf \
   -calculate_log CalculatePowerAverageBatch.log \
   -mode_file txrx.mode \
   -power_db_name $design.AverageBatch.pdb \
   -scenario_file $design.Batch.scn \
   -start_time 6014730ps \
   -top_instance txrx_tst.top1 \
```

Figure 3.10: CalculatePower sample inputs

Chapter 4

RTL Power Analysis flow setup

4.1 Flow directory structure

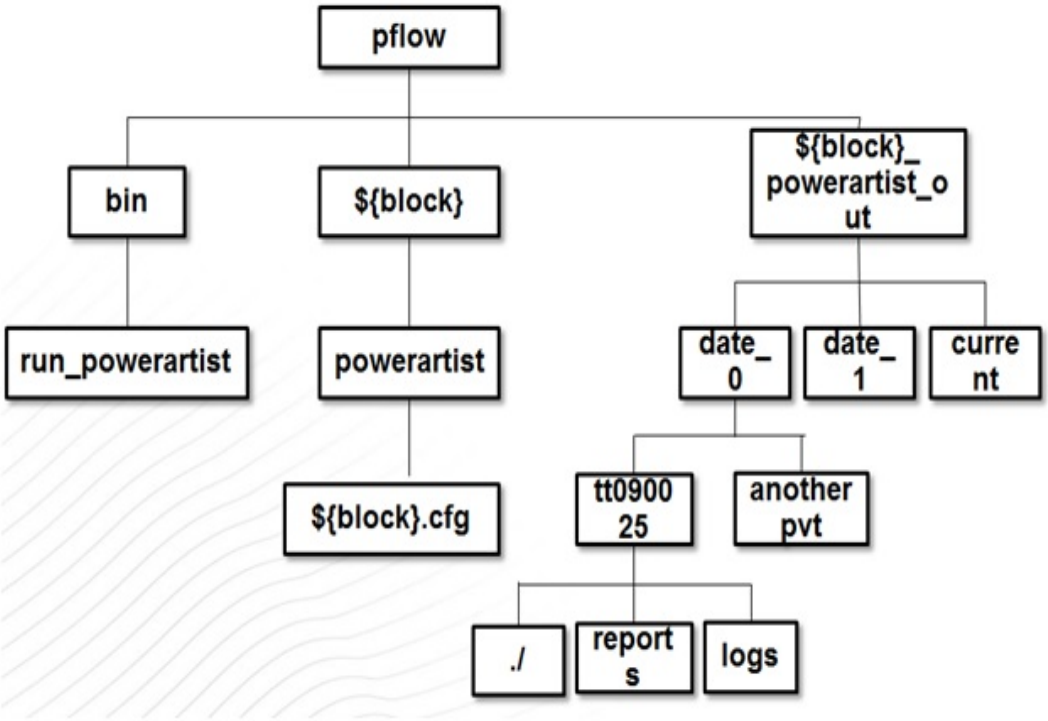

Figure 4.1: General flow for Power Analysis Setup for any block

A specific flow for Power Analysis has been setup. This flow is uniform across some of the tools used by the company which includes PowerArtist tool. Figure 4.1 shows the general directory structure for any module/block whose Power Estimation is to be done. For Power Analysis of any block, inputs required (as discussed in chapter 3) for Power Anlaysis has to be set in this flow in the form of run_powerartist, makefile & various other tcl scripts.

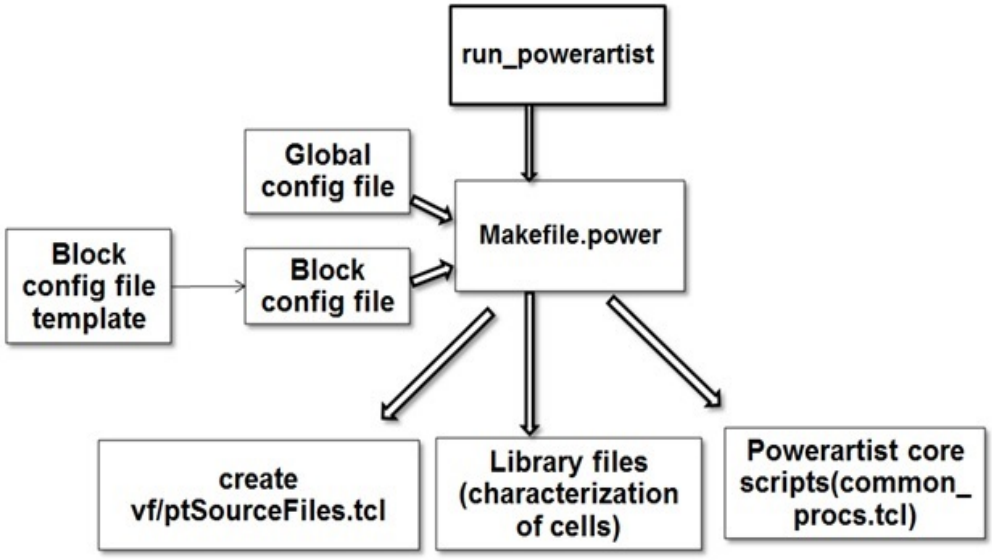

Figure 4.2: Script Configuration for the flow

4.2 RTL Design file list generation

RTL Design lib of a block has to be elaborated for Power Analysis. If RTL Design is in Verilog, Verilog startup file has to be created. For VHDL Design, PowerArtist needs ptSourceFiles.tcl for elaboration.

```
+define+macro1
+define+macro2
../data/design/rtl/txchan.sv
../data/design/rtl/txfsm.v
../data/design/rtl/upi.sv
../data/design/rtl/mem_wrap_half.v
../data/design/rtl/RR2P.v
../data/design/rtl/RR2P_half.v
../data/design/rtl/IBUFDR.v
../data/design/rtl/DDRVO.v
+libext+.v+.sv
```

Figure 4.3: Sample Verilog Startup file

For generation of ptSourceFiles.tcl, perl script has been made. This script will make list of VHDL Design files in the format acceptable to PowerArtist tool i.e ptSourceFiles.tcl format.

The ptSourceFils.tcl file is a Tcl file that contains the following sections:

- Definitions of all of the standard libraries and the source files that define them.
- AddLibrary commands that map logical library names to physical libraries. A typical AddLibrary command would be:
  AddLibrary WORK/system/u/demouser/work
- CompileFile commands, providing file names, libraries, and VHDL standards (87/93) to apply when compiling.

A typical CompileFile command would be:

CompileFile -file test.vhdl -library /system/u/demouser/work -87 yes

We also need to specify VHDL libraries. These commands are written so that they support the legacy flow of pre-compiled libraries. CompileFile can

```
# 1) Library Files

AddLibrary SYNOPSYS ww_synopsys
AddLibrary STD ww_std
AddLibrary IEEE ww_ieee
AddLibrary 'mcm_lib' /projects/XYZ/HW/mcm_lib
AddLibrary 'synchronizerr2_lib' /projects/HW/synchronizerr2_lib
AddLibrary 'clock_r1_lib' /projects/XYZ/clock_r1_lib

# 2) VHDL Packages

CompileFile -type vhdl -file /tools/sequence/powertheater/2015.1.1p1/pthdl_src/standard_93.vhd -93 no -library ww_std
CompileFile -type vhdl -file /tools/sequence/powertheater/2015.1.1p1/pthdl_src/std_1164.vhd -87 no -library ww_ieee
CompileFile -type vhdl -file /tools/sequence/powertheater/2015.1.1p1/pthdl_src/syn_arit.vhd -87 no -library ww_ieee

# 3) RTL Design files

CompileFile -type vhdl -file /projects/XYZ/HW/mcm_lib/rtl/mcm_lib_tech_cell_wrapper_pkg.vhdl -93 yes -library mcm_lib
CompileFile -type vhdl -file /projects/XYZ/HW/mcm_lib/rtl/ent_and2.vhdl -93 yes -library mcm_lib
CompileFile -type vhdl -file /projects/XYZ/HW/mcm_lib/rtl/arc_and2_rtl.vhdl -93 yes -library mcm_lib
CompileFile -type vhdl -file /projects/XYZ/HW/mcm_lib/rtl/ent_cgb.vhdl -93 yes -library mcm_lib
CompileFile -type vhdl -file /projects/XYZ/HW/synchronizerr2_lib/rtl/mod-sync-rtl.vhdl -93 yes -library synchronizerr2_lib
CompileFile -type vhdl -file /projects/XYZ/HW/synchronizerr2_lib/rtl/synchronizer.vhdl -93 yes -library synchronizerr2_lib
CompileFile -type vhdl -file /projects/XYZ/HW/clock_r1_lib/rtl/div2_fix_bp_cfg.vhdl -93 yes -library clock_r1_lib
CompileFile -type vhdl -file /projects/XYZ/HW/clock_r1_lib/rtl/div4_fix.vhdl -93 yes -library clock_r1_lib
```

Figure 4.4: Sample ptSourceFiles.tcl for VHDL Design

simply skip the physical-to-logical mapping and accept a logical library name:

CompileFile -file test.vhdl -library WORK -87 yes

4.3 FSDB generation during simulation

Once appropriate testcase is selected for simulation of RTL design, Verdi tool can be used for Fast Signal Database (FSDB) generation during simulation. FSDB format has the following advantages over the standard VCD file format:

- An FSDB file is more compact than a standard VCD file. Typically, an FSDB file is about 5 to 50 times smaller than a VCD file.
- Using FSDB files, the Verdi system displays waveform and back-annotated signal values faster.
- SpringSoft provides a set of object and support files in the Novas package that can be linked with popular simulators to extend the sim-

ulator command set to support dumping FSDB files directly during simulation[6].

4.3.1 Linking FSDB Dumper for FSDB Dump commands

If we invoke any Verilog FSDB dumping commands in SystemVerilog code, we must use the PLI table file novas.tab when linking the archive library file pli.a with the VCS-MX final simulation file[6]. For SystemVerilog top-level or VHDL-top-level designs, we must use VCS to elaborate the design.We can link the FSDB dumper as follows:

+vcsd P $NOVAS_INST_DIR/share/PLI/VCS/$PLATFORM/novas.tab
$NOVAS_INST_DIR/share/PLI/VCS/$PLATFORM/pli.a

If we invoke any FSDB foreign functions in the VHDL code, we must compile and include a VHDL package in design.

Use the following steps:

- Use VCS-MX analyzer, vhdlan, to compile the Novas provided VHDL file novas.vhd to the same directory as the design is saved.

  vhdlan $NOVAS_INST_DIR/share/PLI/VCS/$PLATFORM /novas.vhd

  The VHDL file novas.vhd contains the definitions of the FSDB foreign functions.

- Use the novas package in any VHDL design files that invoke FSDB foreign functions. For example :

library IEEE ;
use IEEE.std_1164.all ;
use work.novas.all ; - - using novas package.

entity testbench is
end ;
architecture blktestbench is
Begin...

Process begin: dump
fsdbDumpvars(0, ":", "+fsdbfile +signal.fsdb") ; - - call VHDL procedure
wait ;
end process
end ;
then recompile the VHDL files you modified.

In the above example, *fsdbfile* command is used to specify the FSDB file name created by the Novas object files for FSDBdumping. If not specified, the default FSDB file name is " novas.fsdb". *fsdbDumpvars* command is used to dump the signals in the current and succeeding scope the design.

Generated FSDB file can be viewed in Verdi waveformviewer and appropriate timing window can be selected based on maximum toggling of signals.

4.4 Technology Libraries list

Technology specific physical libraries for standard cells and memories has to be given as one of the inputs for Power Analysis. Wireloads model for interconnect capacitance and resistance values can also be included alongwith technology libraries list.

```
# Specify the technology libraries
ReadLibrary -name ../data/libraries/hvt.lib
ReadLibrary -name ../data/libraries/lvt.lib
ReadLibrary -name ../data/libraries/iopad.lib
ReadLibrary -name ../data/libraries/mem_DP256x32.lib
ReadLibrary -name ../data/libraries/mem_DP512x32.lib
ReadLibrary -name ../data/libraries/retention.lib

# Compile them into a binary library database
WriteLibraryDatabase -write_library_database_dir ./$DB_DIR/library_db -write_library_database_log ./
$LOG_DIR/write_lib_db.log
```

Figure 4.5: Sample list of Technology libraries

```
library (my_library) {
technology ( cmos ) ;
delay_model : table_lookup;
...
cell(inv0d0) {
area : 0.75;
mode_definition(rw) {
mode_value(read) {
when : "I";
sdf_cond : "I == 1";
}
mode_value(write) {
when : "!I";
sdf_cond : "I == 0";
}
}
leakage_power () {
mode(rw, read);
value : 2;
}
leakage_power () {
mode(rw, write);
value : 2.2;
}
pin(I) {
direction : input;
max_transition : 2100.0;
capacitance : 0.002000;
fanout_load : 1;
...
}
...
} /* cell(inv0d0) */
} /* library
```

Figure 4.6: Sample Technology library for Invertor cell for Power model

Chapter 5

Analyzing Power Reports and Power reduction opportunities

5.1 Basics of detailed Power report

The detailed power report is stored in an ASCII file (or as an HTML file if requested using the -html_report_title option) [7]. Major sections include the following:

1. Header includes :
    - Date and time report file was generated.
    - Version of PowerArtist used.
    - Library file used for the power calculation.
    - Assumed clock frequency.
    - Voltage values for the power supplies used.
    - Name of the activity file.
    - Names of any capacitance files.
    - How the power consumed in driving inter-module nets is reported.
    - Design and simulation prefix.
2. Total Power Consumption: summarizes the results of the analysis.

3. Internal Power Consumption: lists the power consumed by each individual leaf module in the design.

4. Clock Power Consumption: lists the following information:

    - The hierarchical net name, the type of analysis, and the total power for the clock tree. The analysis type will be either Instantiated, which means that it was traced, or Inferred meaning clock inferencing was done.
    - Area occupied by the net.
    - Clock senses.
    - Frequency of the clock net
    - Transition time of the clock net
    - Fanout capacitance of the clock net (divided into contributions from wire and pin capacitances).
    - Power consumed as the net toggles (divided as wire and pin components).
    - The wire load model applied to the net (the first entry is the net name and thesecond entry is the wire load model applied to that net)
    - Descriptions of the instances and net that were traced as part of the clock tree.
    - The clock tree inferencing that occurred including buffers and integrated clock gating cells that were inferenced.
    - The clock gating summary that provides statistics on the number of gated and ungated registers and the number of integrated clock gating cells either instantiated or inferenced in design.

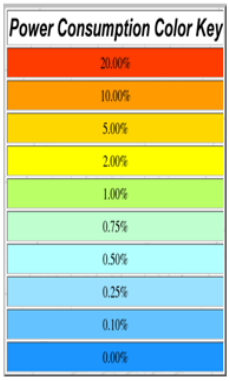

Figure 5.1: Power Consumption Color Key

Power Consumption

	Static	Dynamic	Total
Total	1.43mW	34.9mW	36.4mW
Internal	1.39mW	13.3mW	14.7mW
IP Core	0W	0W	0W
Pads	0W	0W	0W
Clock	21.9uW	20.4mW	20.4mW
Clock Domain	1.4mW	14.5mW	15.9mW
Inferred Buffer	17.2uW	1.27mW	1.29mW

Design Area

	Total-area	Net-area	Registers	Gates
Area	9.27M	0	201839	23734125

Figure 5.2: Sample Power report

Component	Cell Count	Power (Watts)		
		Static	Dynamic	Total
lte_active_asic	810247	1.43mW	34.9mW	36.4mW
Clock Power	13691	21.9uW	20.4mW	20.4mW
Memory Power	111	433uW	6.92mW	7.35mW
Other Power	473246	813uW	5.72mW	6.53mW
Inferred Buffer Power	316138	17.2uW	1.27mW	1.29mW
Register Power	6756	145uW	652uW	796uW
Latch Power	305	113nW	476nW	589nW
IP Core Power	0	0W	0W	0W
Pad Power	0	0W	0W	0W

Figure 5.3: Vertical Power report stating component power

Net	Estimator	Frequency (Hz)	Clock Power (Watts)		
			Static	Dynamic	Total
lte_active_asic.RefClk	Inferred	246M	12uW	6.22mW	6.24mW
lte_active_asic.Bus_ReqClk	Inferred	21.7M	87.7nW	7.26uW	7.35uW
lte_active_asic.VASIP_RefClk	Inferred	307M	9.07uW	14.1mW	14.2mW
lte_active_asic.VASIP_Slave_ReqClk	Inferred	0	87.6nW	0W	87.6nW
lte_active_asic.VASIP_Master_RespClk	Inferred	0	83.2nW	0W	83.2nW
lte_active_asic.CTDEC_UDDEC_ReqClk	Inferred	0	83.8nW	0W	83.8nW

Figure 5.4: Clock Power

LTE_ACTIVE_ASIC.INST_LOGIC_RAMS.INST_LOGIC Components	Width	Height	Registers	Gates
I_CG [illegible] A	1.53	1.53	0	3
I_CG [illegible] B	1.53	1.53	0	3
I_CG [illegible] A	1.53	1.53	0	3
I_CG [illegible] B	1.53	1.53	0	3
INST_[illegible]	34.6	34.6	194	1992
INST_[illegible]	85.1	85.1	521	10415
INST_[illegible]P	185	185	3305	50127
INST_[illegible]	183	183	3861	58604
INST_[illegible]	488	488	18738	392429
INST_[illegible]	57.1	57.1	339	5297

Figure 5.5: Area of each Instance in the Design

Clock Gating Summary

```
Instance: LTE_ACTIVE_ASIC
Number of inferred clock gating cells: 3946
Number of registers gated by inferred clock gating cells: 178829
Number of instantiated clock gating cells: 0
Number of registers gated by instantiated clock gating cells: 192251
Total number of gated registers: 201829
Total number of ungated registers: 171
```

Figure 5.6: Clock Gating Summary

5.2 Power reduction opportunities based on Power reports

Predictable power profiling throughout the project execution has become an important need in designs. Designers struggle to identify inefficient clock gating and power hungry sub blocks early in the design cycle. Traditionally

power estimation is done post synthesis using synthesized gate level netlist resulting in significant lag time before power issue is discovered. The objective of early power estimation at RTL stage is to identify areas in the RTL that are not power efficient. The tool Power Artist of ANSYS is used for RTL Power Estimation for LTE Modem. Few important testcases are used for generating FSDB and appropriate timing window selection is done using Vector Analysis step. Alongwith Power reports, other useful power debug reports are also generated. These reports being : BAR (Block Activity Ranking) report, Clock Gating Efficiency report, report of clock and data activity at each Flop as well as list of ungated registers in the design. The power bugs in the design are analyzed by looking into the BAR report which states clock and data activity at each internal sub-blocks/instances in the block. A few power reduction opportunities are observed at some instances in the design where clock activity was zero yet data activity was observed. Few scenarios where data activity was zero yet high clock activity is present are also observed. RTL power estimation brings in an added advantage of over 20x faster compared to the post layout power estimation tools and can be run 1-3 months earlier in the design cycle before the layout collaterals are available for the power estimation. It is also very easy to analyze design in RTL stage and bring in small changes that can give big power savings.

Few commands useful for RTL Power Wastage debugging are as follows :

reportCGEfficiency : The command quantifies the efficiency of each clock gate along with the power each clock gate controls. This command generates a comprehensive report of all inferred and instantiated ICGCs in the design.

reportFlopAct : This list out all flops in the design along with their data/-clock activities. One can easily identify power wasted in when either clock or data has less activity compared to the other.

PowerArtist contains power reduction modules called "PowerBots". These PowerBots scan the design looking for a specific design feature, such as an enable signal that can be constructed to turn off unobserved register toggles[7].

A PowerBot performs an analysis of the topology of the design and activities on nets and then makes recommendations on potential changes which can be made to the design[7]. It reports either power savings numbers or power wastage numbers to help you make decision. If it can determine an example change to make, it can provide code snippets as well. Once RTL optimization is done for power reduction, it is essential to do functional verification to check the functionality of design is not impaired.

Block Name	Hier. Level	Total Power (W)	% of Top Total Power	Total Clock Power (W)	Clock Activity	Data Activity
XYZ	0	4.15236e-01	100.00000	2.41075e-02	0.35	8.26148e-04
XYZ.INST_LOGIC_RAMS	1	4.00675e-01	96.49329	2.40947e-02	0.35	5.61447e-03
XYZ.INST_LOGIC_RAMS.INST_LOG IC	2	3.67414e-01	88.48312	2.40877e-02	0.35	2.21801e-03
XYZ.INST_LOGIC_RAMS.INST_LOG IC.PROCESSOR	3	1.61852e-01	38.97836	9.02372e-03	2.00	9.79814e-03
XYZ.INST_LOGIC_RAMS.INST_LOG IC.ABC	3	9.50530e-02	22.89134	1.33190e-03	0.06	9.10635e-03

Figure 5.7: Block Activity Ranking report

(Image Courtsy: *PowerArtist - RTL Design for Power Methodology, 2010, Apache Design Solutions, Inc*)

5.3 RTL Power Wastage Debug

Once Power Analysis of a block at RTL level is done, it is of utmost importance to verify the power reports being generated. By analyzing the Power reports, RTL designer can come to know the dynamic and static power associated with each instance in the RTL from hierarchical power reports. Once the power hotspots in the RTL design are detected, RTL designer must check for the presence of any power bugs in the design. These power bugs may cause increase in dynamic power numbers due to uncessary toggling of signals. PowerCanvas is the Graphical User Interface of PowerArtist which proves very helpful for tracing the signals in the internals of the block and observing any unexpected activity on the signals.

Purpose of Power debug is to uncover and reduce wasted power within the block

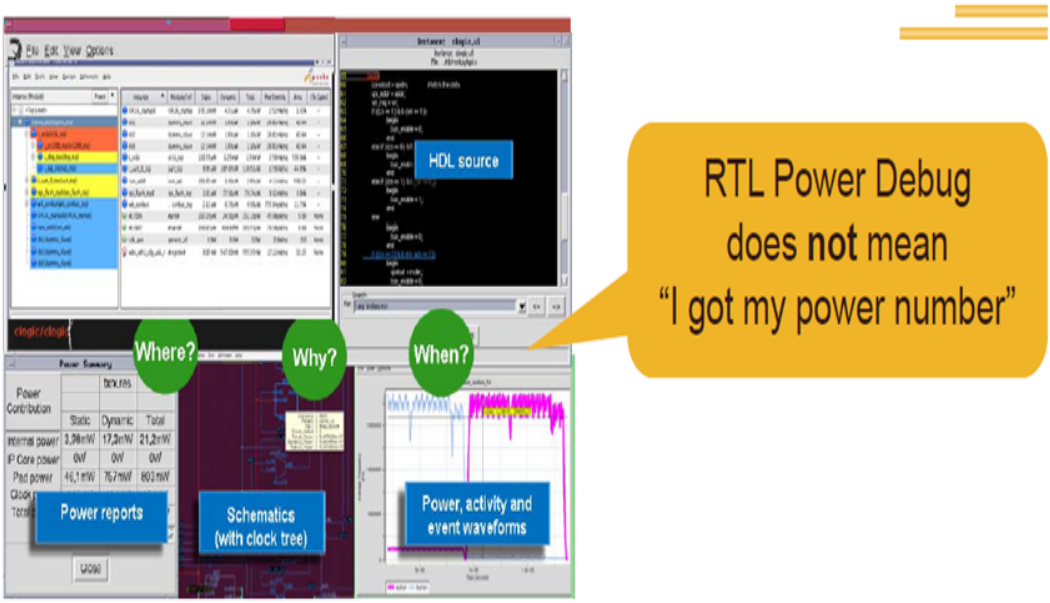

Figure 5.8: RTL power debug with PowerCanvas

(Image Courtsy: *PowerArtist - RTL Design for Power Methodology, 2010, Apache Design Solutions, Inc*)

Power debug is successful when one is able to answer following questions :

- Which sections of the design can be shut off? When?
- Which sections of the design are wasting power? Why?

"Power bugs" may not be functional bugs and they may prove more difficult to detect. Most "power bugs" are due to redundant activity at clock pins or data pins at different instances in the design. PowerArtist helps to identify "Power bugs" early in the design cycle. PowerCanvas is extremely useful for tracing the nets in the design which helps to identify any clock gating or data gating opportunities in the design. While tracing the nets in the design using PowerCanvas GUI of PowerArtist, we can detect any ungated register in the design which is contributing more power due to unnecessary activities on the nets.

5.4 Examples of Power Bugs in the Design

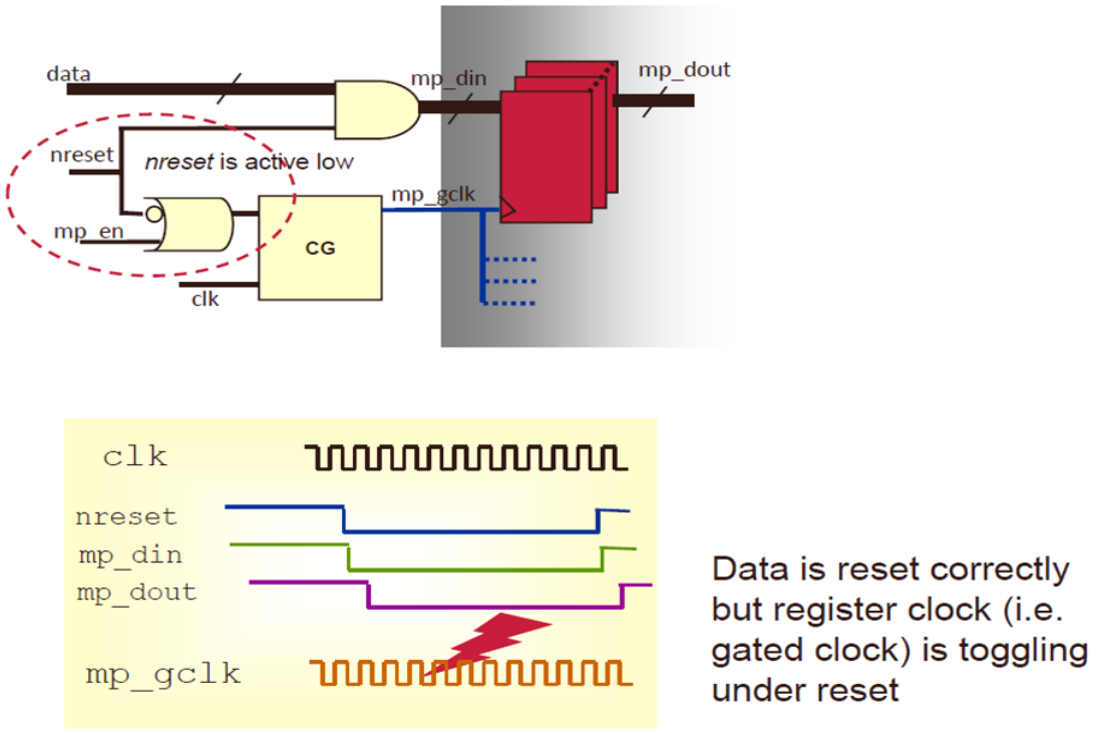

Figure 5.9: Active Clock During Reset

(Image Courtsy: *PowerArtist - RTL Design for Power Methodology, 2010, Apache Design Solutions, Inc*)

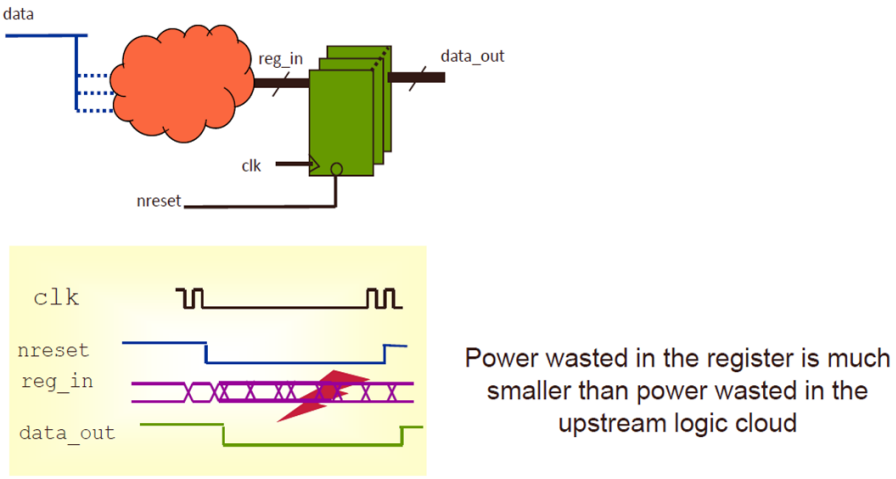

Figure 5.10: Active Data During Reset

(Image Courtsy: *PowerArtist - RTL Design for Power Methodology, 2010, Apache Design Solutions, Inc*)

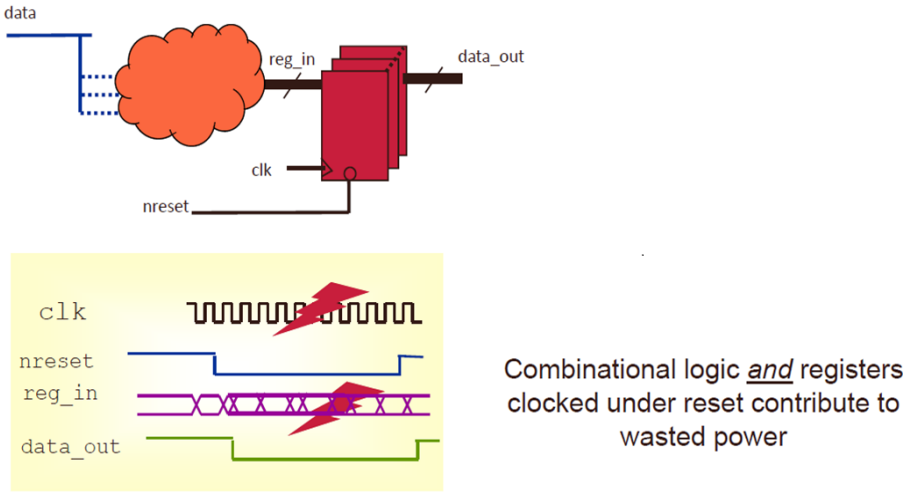

Figure 5.11: Active Data and Clock During Reset

(Image Courtsy: *PowerArtist - RTL Design for Power Methodology, 2010, Apache Design Solutions, Inc*)

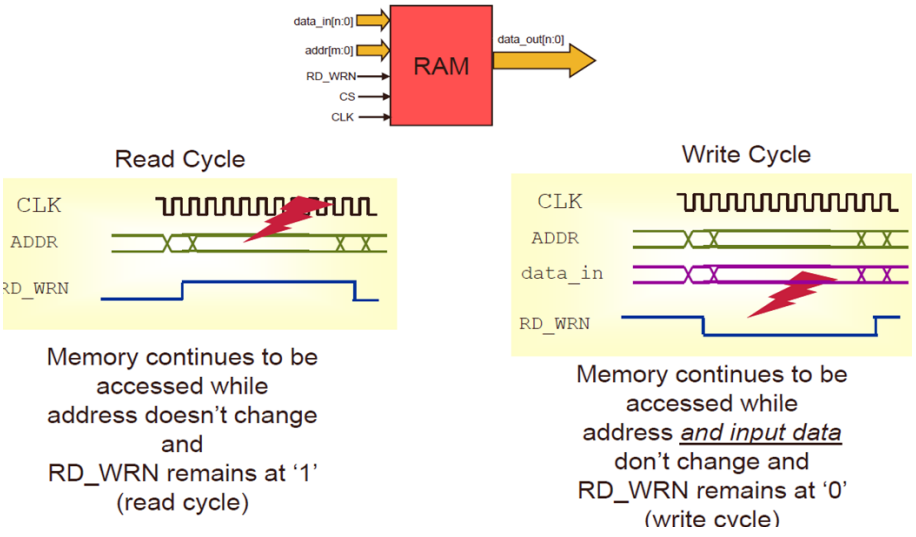

Figure 5.12: Pattern-Dependent Redundant Memory Cycles

(Image Courtsy: *PowerArtist - RTL Design for Power Methodology, 2010, Apache Design Solutions, Inc*)

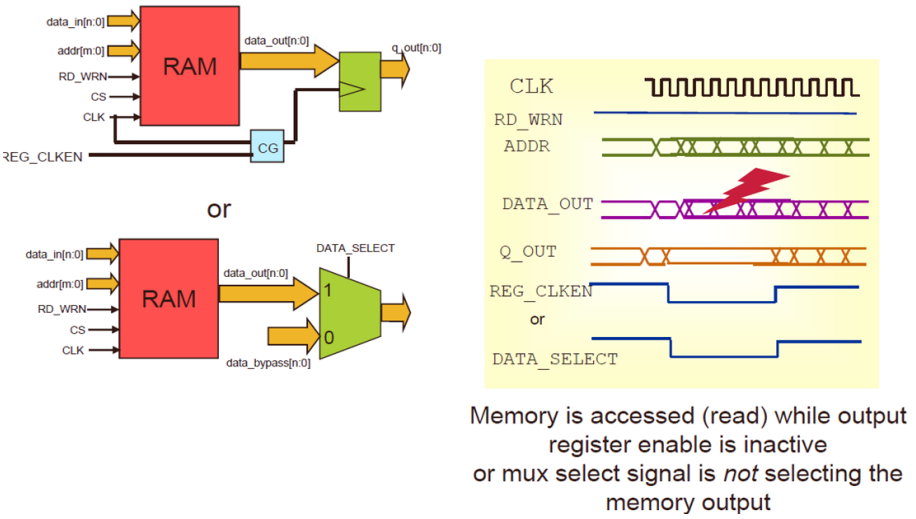

Figure 5.13: Redundant Memory Read to Downstream Register or Mux

(Image Courtsy: *PowerArtist - RTL Design for Power Methodology, 2010, Apache Design Solutions, Inc*)

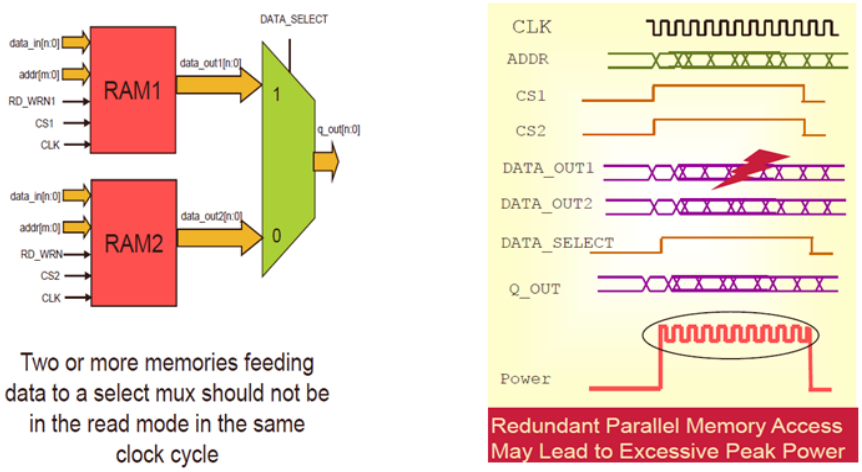

Figure 5.14: Redundant Parallel Memory Access

(Image Courtsy: *PowerArtist - RTL Design for Power Methodology, 2010, Apache Design Solutions, Inc*)

5.5 Critical Warnings affecting Power

To get the accurate power numbers, it is essential to solve all the critical warnings in the power analysis run. Critical warnings are those which has more impact on dynamic power and are observed while tracing a path from particular pin of the microarchitectural sub-block to the output pin of the block.

```
wwengine: Warning 8517: Following sequential elements are not part of any user defined clock domain
        top.core1.a1.#1187
        top.core1.a1.#8852
        ........
```

Figure 5.15: Critical warning showing untraced instances of the block

(Image Courtsy: *PowerArtist - RTL Design for Power Methodology, 2010, Apache Design Solutions, Inc*)

Warning in figure 5.13 shows the elements or instances which are left untraced using defined clock-domains. It means there are few more clock pins in the

design which were not specified and are missed. Dynamic Power associated with those pins will not be calculated if this critical warning is not solved.

How does this message impact power?

- At RTL and gate level Pre-CTS : i.e. for inferred clock trees.
- Clock buffer tree power will not be calculated for these clock nets.
- Clock gating (at RTL) will not be accounted for if sequential elements are registers.

Warning 8517 (Power Analysis):*"The following sequential elements are not part of any user defined clock domain".*
This warning is accompanied with Note which provides details on the percentile of clock tracing coverage and recommended nets to increase the coverage. In addition to specifying clock nets as recommended by PowerArtist, one can also improve clock tracing by using TraceThruCell command if the design contains special clock control cells like clock divider.

Warning 2176 (Vector Analysis):*"Simulation data missing for the following power critical net(s)".*
The user should examine this warning, see if the missing net count is higher than 1significant amount of multidimensional arrays it is recommended to use FSDB). For best accuracy, it is also recommended to use a - level 0 dump from simulation, containing all hierarchical ports, declared nets, and register outputs.

Warning 2818 (RTL Import):*"<verilog.v:line_ no>, Module <module_ name> is marked with 'celldefine. Recommended replacing with definition from technology library".*
This warning usually appears when the RTL contains simulation models of memories but not the .lib files. This can greatly affect accuracy and performance of the simulation and should be addressed immediately by providing

.lib files that contain memory power models. If these LIB files are not available, such memories or other special cells flagged by this warning should be black-boxed as part of the elaboration process.

Chapter 6

Power Analysis Challenges for Complex SoC Design

6.1 Challenges for RTL Power Analysis

1. Power consumption is dependent on both the physical structures on the chip and the mode of operation. With today's multi-mode SoCs, determining the correct stimulus to verify average and peak power across a variety of modes is increasingly challenging.

2. Generally, designers will want to obtain early estimates of power based on available stimulus. For more accurate power estimates, switching activity data is obtained by simulating test cases with real system stimulus. Often, such simulation is not available until later in the design cycle. Designers need to use the most accurate testbench available at any given point in the design flow and revise their estimate as new stimulus becomes available. If switching activity data is not available from simulation, designers should estimate the switching activity on the chip's primary inputs and apply that estimate within the power analysis tool. Transient switching power can be estimated based on the number of flip-flops, combinatorial gates, and clock speed.

3. The functional simulations are Verilog or VHDL simulations. The func-

tional simulation is carried out to generate the FSDB or SAIF file by running the testbench on the RTL or synthesized gate-level netlist. The FSDB generated by running simulation on the RTL is used as an input for RTL power analysis. Selection of appropriate testcase is very important which can give actual functionality of the module and maximum toggling of signals. Testbench should be robust and extremely optimized to capture necessary activities of intented signals [8].

4. One should use physical libraries that represent the worst-case power. Synthesis is done using worstcase timing libraries to optimize for area, timing, and power concurrently, but they do not necessarily represent the worst-case power. Dynamic power is usually the highest in fast conditions, which can be represented by the best-case timing libraries.

5. Use accurate wire modeling. Every designer knows about the inaccuracies of wire load models when it comes to timing closure. Yet, many design teams use a "zero" wire load model for synthesis, resulting in inaccurate power estimation. Use a reasonable wire load model or one of the "physical based" wire-modeling technologies available in today's synthesis tools. Physical layout estimation is a physical modeling technique that bypasses wire loads for RTL synthesis optimization. This may take the form of an equation to model the wire delay. Physical layout estimation uses actual design and physical library information and dynamically calculates wire delays for different logic structures in the design. In most cases, physical layout estimation-synthesized designs correlate better with place-and-route tools.

6. One can compare the RTL Power analysis results with gate-level Power analysis result to check the correctness of Power numbers. Gate level Power anaysis would produce more accurate power numbers compare to RTL Power Analysis approach but would be more time-consuming. RTL Power analysis is considered better approach as it gives fast results which is extremely required during initial RTL design phase.

6.2 Techniques for Switching and Leakage Power Reduction used in Complex SoC

Following are the commonly used Power management techniques employed during Low Power Complex SoC Design [2].

1. **Clock tree optimization and clock gating**

   Portions of the clock tree(s) that aren't being used at any particular time are disabled.

2. **Operand isolation**

   Reduce power dissipation in datapath blocks controlled by an enable signal; when the datapath element is not active, prevent it from switching.

3. **Logic restructuring**

   Move high switching operations up in the logic cone, and low switching operations back in the logic cone; a gate-level dynamic power optimization technique.

4. **Logic resizing (transistor resizing)**

   Upsizing improves slew times, reducing dynamic current. Downsizing reduces leakage current. To be effective, sizing operations must include accurate switching information.

5. **Transition rate buffering**

   Buffer manipulation reduces dynamic power by minimizing switching times.

6. **Pin swapping**

   By swapping gate pins, switching occurs at gates/pins with lower capacitive loads.

7. **Multi-Vth**

   With the use of multi-threshold libraries, individual logic gates use transistors with low switching thresholds (faster with higher leakage) or high switching thresholds (slower with lower leakage).

8. **Multi-supply voltage (MSV or voltage islands)**

   Selected functional blocks are run at different supply voltages.

9. **Dynamic voltage scaling (DVS)**

   In this subset of DVFS, selected portions of the device are dynamically set to run at different voltages on the fly while the chip is running.

10. **Dynamic voltage and frequency scaling (DVFS)**

    Selected portions of the device are dynamically set to run at different voltages and frequencies on the fly while the chip is running. Used for dynamic power reduction [2].

11. **Adaptive voltage and frequency scaling (AVFS)**

    In this variation of DVFS, a wider variety of voltages are set dynamically, based on adaptive feedback from a control loop; involves analog circuitry.

12. **Power shut-off (PSO) [or power gating]**

    When not in use, selected functional blocks are individually powered down. Memory splitting If the software and/or data are persistent in one portion of a memory but not in another, it may be appropriate to split that block of memory into two or more portions. One can then selectively power down those portions that aren't in use.

13. **Substrate biasing (bodybiasing or back-biasing)**

    Substrate biasing in PMOS biases the body of the transistor to a voltagehigher than Vdd; in NMOS, to a voltage lower than Vss.

Chapter 7

Conclusion

Methodology for Power Analysis for SoC described in this thesis is a part of Low Power SoC Design and is based on a unique Power Analysis flow developed and used by Broadcom Limited to estimate the static and dynamic power using PowerArtist tool. Static power is leakage power and is looked from energy table in the technology library. Dynamic power comprises of internal power and switching power. Internal power is calculated based on the rise power and fall power numbers referred in energy table in technology library. Switching power is estimated from toggling of signals in FSDB file. RTL Power analysis methodology described in this thesis depends on the usecase selected for simulation of RTL design, technology specific physical libraries for standard cells and macros (indicating leakage and internal power for various operating conditions and input pin values), wireload model used for interconnect resistance and capacitance modeling and time interval selected for Power estimation in FSDB file.

References

[1] Anantha P. Chandrakasan, Samuel Sheng, and Robert W. Brodersen, "Low-Power CMOS Digital Design" , *IEEE* JOURNAL OF SOLID-STATE CIRCUITS. VOL. 27. NO. 1, APRIL 1992.

[2] Dr. Chi-Ping Hsu, Corporate Vice President, IC Digital and Power Forward, Cadence Design Systems, "A Practical Guide to Low-Power Design", 2005

[3] Christian Piguet, "Low Power CMOS Circuits, Technology, Logic Design and CAD Tools", 2006

[4] Michael Keating, David Flynn, Robert Aitken, Alan Gibbons, Kaijian Shi, "Low Power Methodology Manual for System-on-Chip Design" , Synopsys Inc & Arm Limited, 2007

[5] Messoud Pedram & Jan M. Rabaey, "Power Aware Design Methodologies" by Springer Publications, 2007

[6] "Linking Novas Files with Simulators and Enabling FSDB Dumping" by SpringSoft Inc , version 2010.04

[7] PowerArtist User guide, Apache Design Solutions Inc, 2015

[8] Preeti Gupta,"Be Early with Power", Chip Design Magazine. www.chipdesignmag.com/display.php?articleId=613