Upgrading the Graphical User Interface of a Commercial Software used for Low Power Chip Design

Submitted By

Anshul Dadhich

14mcec02

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2016

Upgrading the Graphical User Interface of a Commercial Software used for Low Power Chip Design

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Anshul Dadhich

(14mcec02)

Guided By

Prof. Dhaval S Jha

तमसो मा ज्योतिर्गमय

NIRMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

NAAC ACCREDITED 'A' GRADE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2016

Certificate

This is to certify that the major project entitled **"Upgrading the Graphical User Interface of a Commercial Software used for Low Power Chip Design"** submitted by **Anshul Dadhich (Roll No: 14mcec02)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by him 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 part-I, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Internal Guide:
Prof. Dhaval S Jha
Assistant Professor,
CSE Department,
Institute of Technology,
Nirma University,
Ahmedabad

External Guide:
Manish Kumar
Senior Engineering Manager,
Power Pro R&D Team,
Calypto Systems Division,
Mentor Graphics Corporation,
Noida

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. P. N. Tekwani
(I/c) Director,
Institute of Technology,
Nirma University, Ahmedabad

Certificate

This is to certify that the two major projects entitled **"Upgrading The Graphical User Interface Of A Commercial Software Used For Low Power Chip Design"** submitted by **Anshul Dadhich (Roll No: 14MCEC02)**, towards the fulfillment of the requirements for the award of degree of Master of Technology in Computer Science & Engineering (CSE) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Date:

Mr. Manish Kumar

Senior Engineering Manager

PowerPro R&D Department

Calypto Systems Division

Mentor Graphics Corporation

Noida

Statement of Originality

I, **Anshul Dadhich**, Roll. No. **14mcec02**, give undertaking that the Major Project entitled "**Upgrading the Graphical User Interface of a Commercial Software used for Low Power Chip Design**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student
Date:
Place:

Endorsed by
Prof. Dhaval S Jha
(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Dhaval S Jha**, Assistant Professor, Computer Science Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

I would also like to mention my gratitude towards **Mr. Manish Kumar**, Senior Engg Manager, Calypto Systems Division, Mentor Graphics Corporation, Noida for his motivation and guidance in completing the project.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. P. N. Tekwani**, Hon'ble (I/c) Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Anshul Dadhich
14mcec02

Abstract

The Graphical User Interface or GUI [1], is a type of interface that allows users to interact with electronics devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation.

Electronic design automation (EDA) is the category of tools for designing and producing electronic systems ranging from printed circuit boards (PCBs) to integrated circuits. As device sizes are shrinking and chip capacity is increasing, designing these 100 million transistors SoC is becoming a challenge. This is causing a paradigm shift in the EDA design flow causing the designers to move up in the abstraction level. This has warranted new EDA tools in the system-level design and verification arena. The goal of Calypto is to provide a suite of system-level tools. All these tools will be built on a robust and common infrastructure.

Calyptos tool PowerPro is being used for the project. It automates power reduction in RTL designs. Power consumption is a design requirement for all electronic applications. PowerPro is an automated RTL power optimization solution that dynamically reduces power with little or no impact on timing or area. In this project we propose a transformation so that the circuit can be represented in a format where a single object would be able to represent information about different bits.

The GUI of PowerPro is developed using Qt which is a cross-platform application framework and C++ GUI toolkit created and maintained byTrolltech. It provides application developers with all the functionality needed to build applications with state-of-the-art graphical user interfaces. Qt is fully object-oriented, easily extensible, and allows true component programming.

The main objective of doing this project is to make the GUI of PowerPro available in the latest available version of Qt. We are trying to migrate from the older version of Qt to the latest one so that we can fulfill the contemporary requirements of the industry. The newer available version of Qt has some new libraries that provides the software a much faster runtime and makes it able to solve the complex problems in a much lesser span of time. This migration was indeed required for the companys upcoming releases of the software.

Project Area: Development

Keywords: Graphical User Interface, Qt, Power Optimization, cross-platform, application framework.

Contents

Certificate **iii**

Certificate **iv**

Statement of Originality **v**

Acknowledgements **vi**

Abstract **vii**

List of Figures **xi**

1 Introduction **1**
- 1.1 General Introduction to the Topic 1
- 1.2 Organisation 1
- 1.3 Products 2
  - 1.3.1 PowerPro 2
  - 1.3.2 SLEC 3
  - 1.3.3 Catapult 4
- 1.4 Area of Computer Science 4
- 1.5 Partners and Customers 5

2 Problem Definition **6**

3 Objectives **7**

4 Electronic Design Automation **8**
- 4.1 Introduction 8

5 Basics of Power Optimization **11**
- 5.1 Introduction 11
- 5.2 Introduction to Clock Gating 12
- 5.3 Combinational Clock-Gating 12
- 5.4 Sequential Clock Gating 13
- 5.5 Clock Gating Efficiency 14
- 5.6 Memory Gating Efficiency 15
- 5.7 Power Analysis 17
- 5.8 Calypto DataBase 17

6 PowerPro **18**
6.1 Introduction 18
6.2 Basic Concepts Used 19
6.3 Running PowerPro 20

7 Methodology **22**
7.1 Ramp up Session 22
7.2 Development 22
7.3 Status 23

8 Implementation Details **24**
8.1 PreRequisites 24
8.1.1 Qt 24
8.1.2 SPEF- The Standard Parasitic Exchange Format 30
8.1.3 Inverter Chain 35
8.1.4 User Characteristics 38
8.1.5 System Requirements 38
8.1.6 Build/Debug Environment of the Tool 39
8.1.7 Bugzilla 40
8.1.8 Software Requirements 41
8.1.9 Tools Used 46
8.2 Project Work 47

9 Conclusion **49**
9.1 Path Followed 49
9.2 Work Completed 50

10 References **51**

Bibliography **53**

List of Figures

1.1 PowerPro Basic Flow 2
1.2 Slec Working Flow 3
1.3 Catapult Flow 4
1.4 Partners and Customers 5

4.1 EDA STages 9

5.1 Example:-Combinational Clock Gating 12
5.2 Example of Sequential Clock Gating 13
5.3 Clock Gating Efficiency 14
5.4 Clock Gated Registers/Duration 15
5.5 Dynamic Power And Leakage Power 16
5.6 Light Sleep Memory Gating 17

6.1 PowerPro flow 18
6.2 Running PowerPro 21

8.1 Integration Overview 29
8.2 SPEF Header 33
8.3 Name Map Section 33

Chapter 1

Introduction

1.1 General Introduction to the Topic

Electronic Design Automation categorises software tool for the design of electronic systems such PCBs and ICs. The tools work together in a design flow that chip designers use to design and analyze entire semiconductor chips. In designing digital circuits various design parameter are considered for both high-performance and portable applications. The main design parameters are:-

- Power
- Area
- Timing

Here power and area are inter-related to each other, if area is less that means few components are used to design an application then it will take less power. The circuit description is transformed and manipulated on different levels of abstraction to optimize power.

1.2 Organisation

Calypto Design Systems, headquartered in Santa Clara, California, was founded in 2002 by an experienced group of EDA technologists who recognized a need for improvement in Integrated circuit (IC) design and verification. Calypto Design Systems leads the industry in technologies for ESL hardware design and RTL (Register-transfer level) power optimization. These technologies empower designers to create high quality and low power electronic systems for todays most innovative electronic products.

1.3 Products

1.3.1 PowerPro

The PowerPro product family is the industrys most comprehensive RTL power optimization platform and includes PowerPro CG, PowerPro MG, PowerPro Analyzer and the new Power Advisor Flow. Using PowerPro, designers can significantly lower power across a SoC design while reducing overall design time.

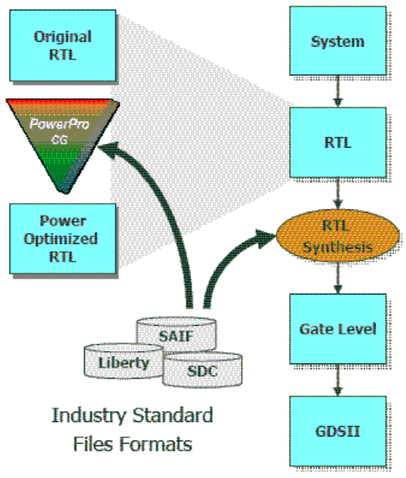


Figure 1.1: PowerPro Basic Flow

Figure 1.1 shows the basic flow of PowerPro tool. It takes RTL as input and desired optimization is run and optimized RTL is produced. PowerPro CG is an automated RTL power optimization tool that reduces power by up to 60 percent with little or no impact to timing or area.

PowerPro MG is an automated memory power optimization solution that takes advantage of the low-power control options available in todays on-chip memories to reduce both dynamic and leakage memory power with little or no impact to timing or area.

PowerPro Analyzer is the industrys most accurate register-transfer level (RTL) power analysis tool. PowerPro Analyzer also provides complete visualization of PowerPro CG and PowerPro MG optimizations, allowing users to view power optimizations in the context of RTL source code, schematic display, storable reports (ASCII, HTML, CSV, XML), and design hierarchy.

1.3.1.1 Benefits of Powerpro

- Reduces dynamic power at the RTL level
- Automatically identifies sequential clock gating opportunities
- Reduces dynamic power without impacting timing or area
- Allows users to make power, area and timing trade-offs

1.3.2 SLEC

It provides comprehensive RTL verification. Calypto delivers software products, based on unique Sequential Analysis technology, that enable designers to efficiently produce the highest quality electronic systems using automated and optimized design and verification methodologies. Unlike simulation test benches, SLEC SYSTEM verifies functionality in spite of design differences in throughput, latency and I/O. SLEC SYSTEM is ideal for verifying RTL implementations derived from system-level specifications through manual re-coding or by using behavioral synthesis.

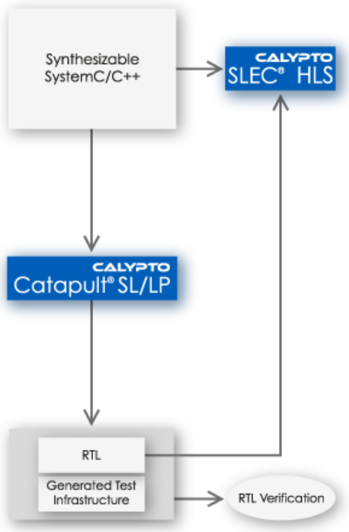


Figure 1.2: Slec Working Flow

1.3.2.1 Benefits of SLEC

- Verifies RTL blocks without the need for the entire assembled design
- Provides comprehensive functional verification without requiring additional test-benches or properties

- Shortens verification cycle by dramatically reducing the need for block level RTL simulation

1.3.3 Catapult

Catapult C Synthesis empowers designers to use industry standard ANSI C++ and System C to describe functional intent and move up to a more productive abstraction level. From these high-level descriptions, Catapult C Synthesis generates production quality RTL.

Fig 1.3 shows the flow of Catapult.

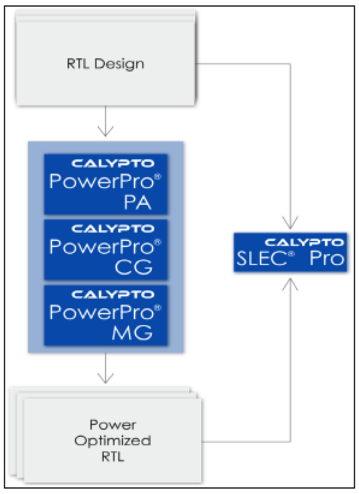


Figure 1.3: Catapult Flow

1.4 Area of Computer Science

- Electronic design automation (EDA):-

  Computer-aided design (CAD) is the use of computer systems to assist in the creation, modification, analysis, or optimization of a design. Computer-aided design is used in many fields. Its use in designing electronic systems is known as electronic design automation, or EDA.

- Memory leakage and run time
- Testing
- GUI Design and Development
- Development

- Power Analysis

1.5 Partners and Customers

Figure ?? shows all the partners and customers.

Figure 1.4: Partners and Customers

Chapter 2

Problem Definition

The field of Electronic Design Automation has a strong dependency over the tools and technologies used in it. The better and upgraded the technologies used, faster the results will be, whether it is the Graphical User Interface or the Power Analysis tools.

If we talk about GUI, it is the only way to transfer information to and from the system. The more efficient the GUI is, more reliable and faster the results will be shown and more user friendly the GUI will be.

The problem definition is therefore as follows:

The Graphical User Interface of a Commercial Software (Power Pro), used for Low Power Consuming Chip Designing is to be upgraded from the current version of application framework (Qt[2]) to a newer and better one available.

Qt is a cross platform application framework. Qt has all the functionality needed to build applications with some amazing state-of-the-art graphical user interfaces. It fully object-oriented, can be extended easily, and allows true component programming.

Chapter 3

Objectives

- The main objectives are:-
  - Main objective is to Upgrade the Graphical User Interface of PowerPro.
  - This will affect the visualization, performance, appearance, as well as the working of Graphical User Interface of PowerPro.
  - We are upgrading from Qt3 to Qt4. And the reason why we decided to upgrade the GUI from the older version of Qt to the latest one is so that we can fulfill the contemporary requirements of the industry. The newer available version of Qt has some new libraries that provides the software a much faster runtime and makes it able to solve the complex problems in a much lesser span of time. This migration was indeed required for the companys upcoming releases of the software.

Chapter 4

Electronic Design Automation

4.1 Introduction

This chapter gives a detailed description of EDA (Electronic Design Automation) .i.e. the domain in which Calypto is working.

Electronic Design Automation categories the software tools for design and production of electronic systems starting from PCBs to ICs. This is sometimes referred to as ECAD (electronic computer-aided design) or just CAD. (Printed circuit boards and wire wrap both contain specialized discussions of the EDA used for those)

It is the process of automation of steps in electronic circuit design. Traditionally, designers have relied on customized methods to design circuits. However, over past few decades the number of transistors that can be put on a single chip has increased by leaps and bounds. Hence the need for automating complex steps of circuit design.

To understand the rapidly growing, almost four billion dollar EDA industry, we need to define the words behind those three letters, "EDA":

- Electronic:

  Anything electronic from computer chips, cellular phones, pacemakers, controls for automobiles and satellites to the servers, routers and switches that run the Internet. As electronics become even more complex and pervasive, the EDA industry is more vital to the continued success of the global economy.

- Design:

  The part of the production cycle where creativity, new ideas, ingenuity and inspiration comes to the fore. This is also where designers try to model the behavior of their designs and analyze the complex interactions of millions of constituent parts in their designs to ensure completeness, correctness and manufacturability of the final product.

- Automation:

  Imagine the difference between designing a small house versus designing a mile-high skyscraper. For the skyscraper you need to design sophisticated structural, electrical, plumbing, security and environmental systems, communications and computer networks, elevators, etc. all working together. This is analogous to the dramatic increase in complexity that designers must tackle in electronics today.

Figure **??** shows different stages of Chip Fabrication

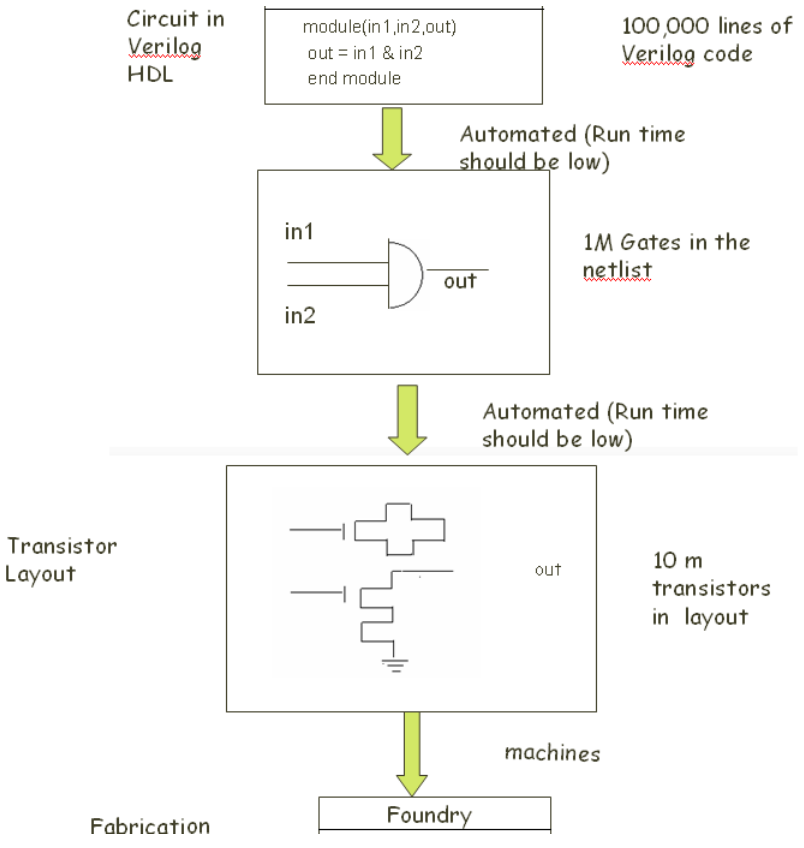


Figure 4.1: EDA STages

The electronic design process involves describing the behavioral, architectural, functional, and structural attributes of an IC or electronic system. The process is one of successive refinement where the design team adds a level of detail to the design and verifies the results of the addition before proceeding to add each next level of detail. Design teams begin with very abstract behavioral models of their system and end with a physical description of millions of transistors and their interconnections. Semiconductor foundries use the physical description to create the masks and test programs needed to manufacture the ICs. EDA tools aid the design team in capturing its design intent, creating the next level of detail, and verifying the results. Problems are often found at one level that can only be resolved by revising a previous level of description. These iterations introduce delay and risk into the design process.

Chapter 5

Basics of Power Optimization

This chapter gives a brief idea of some concepts used for power optimization in PowerPro-CG.

5.1 Introduction

As a first order approximation, the power consumed in a CMOS gate can be approximated as:

$$P = 0.5CLV^2f + VI \tag{5.1}$$

Where, V is the supply voltage, CL is the load capacitance, f is the switching frequency (or toggle rate), and I represents the static current drawn from the power supply.

The first term of this equation describes the dynamic power of the design and the second term describes the static power. To reduce the total power, it is important to reduce both components of this equation, or at least make sure that there is a cumulative reduction. To optimize for power, it is important to make sure every stage of the design (right from algorithm to transistor level) is optimized to the extent possible, at the same time keeping any negative impact on the subsequent stage to a minimum. Even at process geometries of 32 nm or below, dynamic power still constitutes a majority of the total power dissipation.

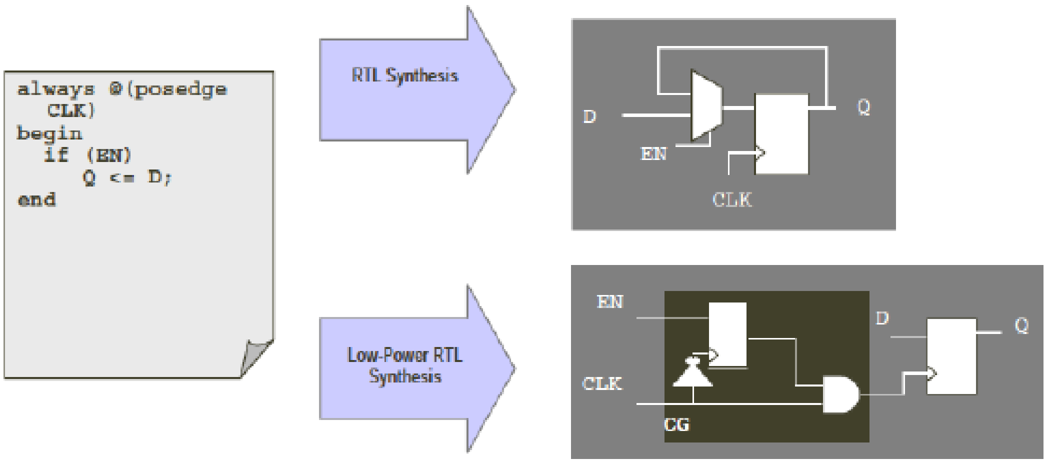
A designs dynamic power consumption is a function of the switching activity of its transistors. Power optimizations at the higher levels of design abstraction have the greatest impact. Even though every design stage offers opportunities to save power, designers have a lot more freedom at higher levels of abstraction to make design changes, in terms of design intent and constraint, without compromising on target performance or area. Clock gating is a popular design technique used to optimize dynamic power. The granularity of clock gating and the impact it has on overall power consumption depends on the phase of the design cycle.

5.2 Introduction to Clock Gating

Most clock gating is done at the Register Transfer Level (RTL). RTL clock gating algorithms can be grouped into three categories: system-level, sequential and combinational. System-level clock gating stops the clock for an entire block, effectively disabling all functionality. On the contrary, combinational and sequential clock gating selectively suspends clocking while the block continues to produce output. Clearly, not all of these are equal when it comes to reducing switching activity.

5.3 Combinational Clock-Gating

Combinational clock gating identifies the condition when the data is held in a register and shuts off the clock to the register during that period. This leads to reduction of dynamic power consumed by the register and the clock network driving the register. Consider the following example.



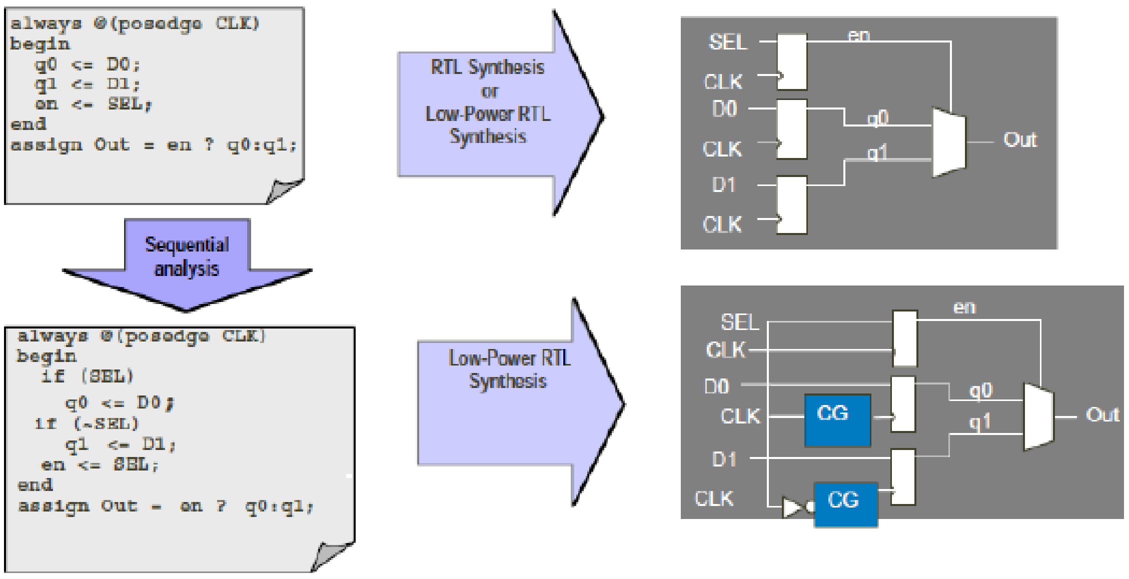
```
always @(posedge
  CLK)
begin
  if (EN)
     Q <= D;
end
```

Figure 5.1: Example:-Combinational Clock Gating

In this example, the netlist generated by the RTL synthesis tool from the Verilog code snippet is shown in top right. Register Q loads new data when EN signal is HIGH; otherwise, it holds the data. Opportunities to insert combinational clock gating can be found by looking for conditional assignments in the code. Clock gating logic is substituted when code like if (cond) out ¡= in is present. Power aware logic synthesis tools identify RTL coding patterns and make the appropriate substitution. This is shown in the circuit in the bottom right of Figure 4.1.

5.4 Sequential Clock Gating

Clock gating based on sequential analysis involves identifying new enable conditions and then using them to gate the clock. These enable conditions can be generated to hold the data if new data is not required in the downstream logic or when data is stable or invalid. We will be referring to this transformation as sequential clock gating transformation. Consider the example of such a transformation.



```
always @(posedge CLK)
begin
  q0 <= D0;
  q1 <= D1;
  en <= SEL;
end
assign Out = en ? q0:q1;
```

```
always @(posedge CLK)
begin
  if (SEL)
    q0 <= D0;
 if (~SEL)
    q1 <= D1;
  en <= SEL;
end
assign Out = en ? q0:q1;
```

Figure 5.2: Example of Sequential Clock Gating

In the RTL code snippet on the top left corner in Figure 1-2, registers q0 and q1 are latching a new data value every cycle. Hence, when taken through low-power RTL synthesis tools, they would not have clock gating. If we observe carefully, we notice that if en is HIGH, then the data latched into q1 in the previous cycle is not used. Thus, we can hold the previous data on q1 during that cycle. By performing this sequential reasoning, we can identify SEL as the new enable condition for register q1. Similar sequential analysis will identify SEL as the new enable condition for register q0 and essentially translates to the RTL code snippet in the bottom left. When this code is taken through low-power synthesis tool, it will insert appropriate clock gating logic for register q0 and q1.

System and sequential clock gating offer higher power saving potential as they tend to be more global in nature. Moreover, sequential clock-gating is the most effective way of reducing peak power. Historically, this is mainly achieved by making such changes manually. This process is often difficult and error-prone, partly due to the difficulty in recognizing such opportunities, and partly due to the difficulty in implementing the gating logic without introducing functional errors.

5.5 Clock Gating Efficiency

A typical metric used to measure the effectiveness of clock gating is the percentage of registers in the design that are clock gated. While this gives designers an indication of the clock gating in the design, it has poor correlation to power savings. Dynamic power consumption depends on the switching activity of a given node over a simulation test-bench. Clock gating efficiency, on the other hand, takes this aspect into account, making it a more telling indicator of actual dynamic power consumption.

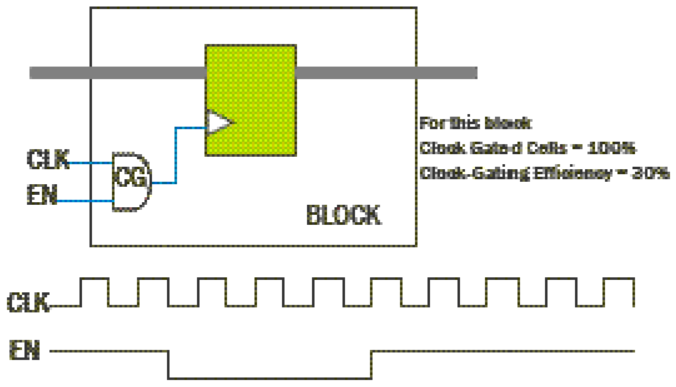


Figure 5.3: Clock Gating Efficiency

Above Fig. shows a block with a single register. Since the only register in the block is clock gated, the block is 100% clock gated. However, since the enable signal is gating the clock input to the register and the clock is inactive for 3 of the 10 cycles, the clock gating efficiency is 30% of the time. The percent of clock gated registers is not as good an indicator of power savings as is the clock gating efficiency. Estimating power depends on representative switching activity. A simulator can generate a switching activity file based on a given test-bench. This is only as representative as the test-bench itself, so selection of a representative test-bench is critical to good power estimation. Clock gating efficiency is defined as the percentage of time a register is gated for a given switching activity. When looking at an entire design, the average clock gating efficiency can be computed as the average of clock gating efficiencies for all registers in the design for a given simulation test bench.

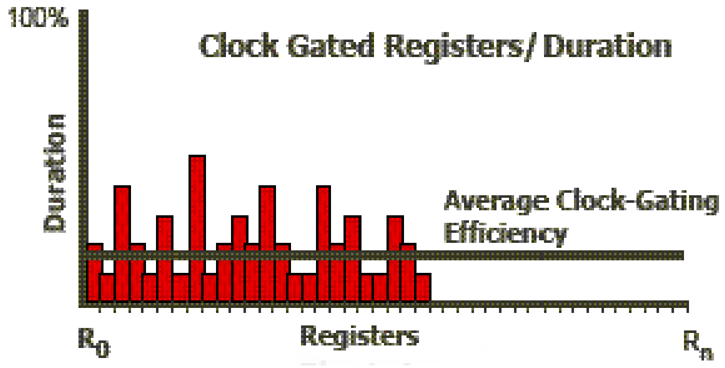


Figure 5.4: Clock Gated Registers/Duration

Above figure shows the average clock gating efficiency for the entire design over a simulation trace. Improving the clock gating efficiency in turn means reduced switching, which can save dynamic power. A designers goal is to improve the average clock gating efficiency as much as possible. It is not practical to achieve 100

5.6 Memory Gating Efficiency

The extent to which a memory has been optimized can be measured by the efficiency of the memory enables of the memory and the effectiveness of the signals controlling the sleep modes available in the memory. Since PowerPro reduces dynamic memory power via memory enable gating (hereafter referred to as memory gating) and memory leakage power via light sleep gating, the metrics of interest in the context of PowerPro are memory gating efficiency and light sleep efficiency.

Memory gating efficiency is defined as the percentage of time the memory is gated for a given switching activity. When looking at an entire design, the average memory gating efficiency can be computed as the average of memory gating efficiencies for all memories in the design for a given simulation test bench. Figure 1-5 shows a design block with a single port (1p) memory. If the original memory enable (ME) is low for 1 of the 10 cycles,

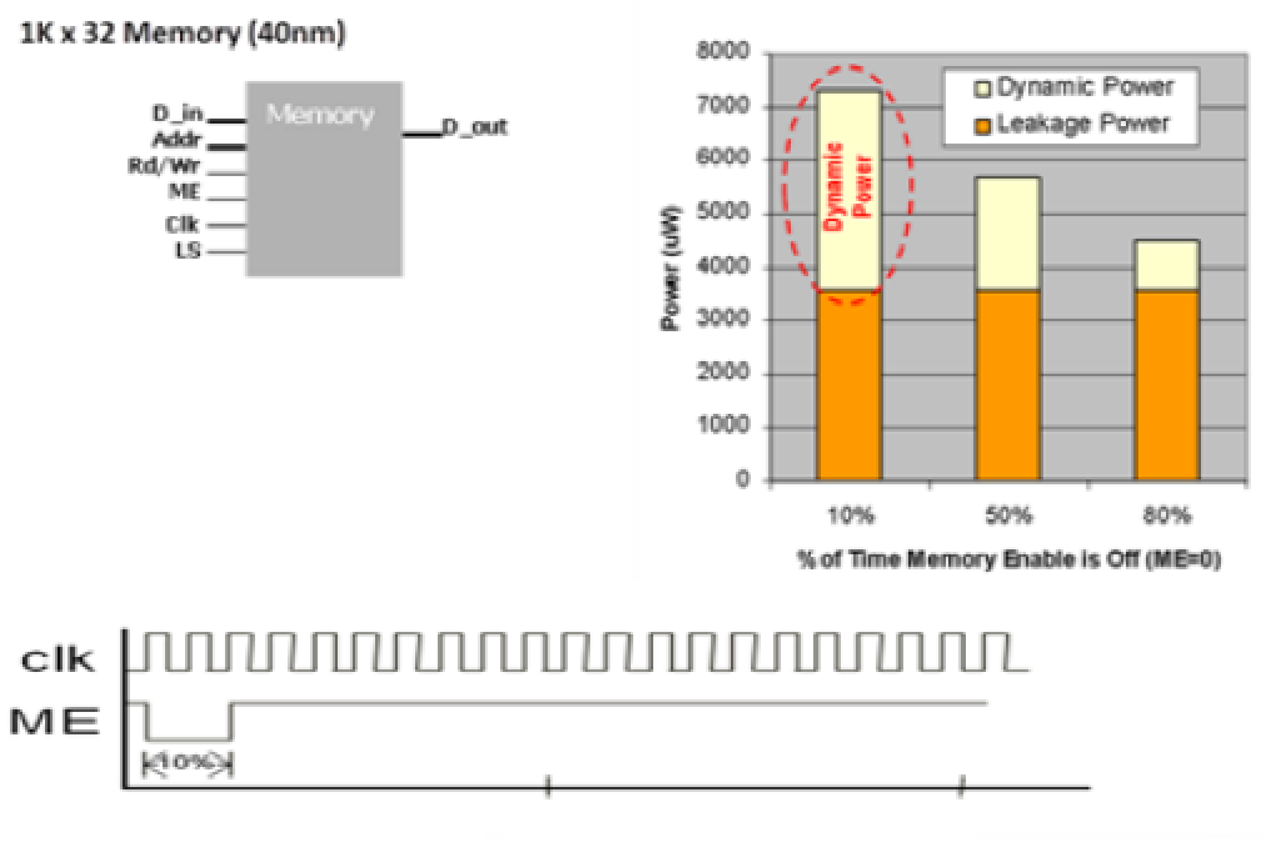


Figure 5.5: Dynamic Power And Leakage Power

the memory gating efficiency is 10%. The first bar in the power chart shows the dynamic and leakage power consumption of the design when memory enable is low for 10% of the simulation time.

PowerPro extends the amount of time the memory is inactive (ME=0), thereby saving dynamic power. Figure 1-6 shows PowerPro memory gating. The new memory enable (ME+MG_ME1) is now low for 50% of the simulation duration thus reducing the dynamic power of the memory by 43%. Further extension in the amount of time the memory is inactive correspondingly reduces the dynamic power of the memory.

Light sleep efficiency is defined as the percentage of time the light sleep control pin for a memory is active. For a design, the light sleep efficiency of the design is a weighted average of the light sleep efficiencies of the individual memories in the design, weighted by the number of bits in the memories. PowerPro implements Light Sleep Memory Gating (Figure 1-7) by extending the amount of time the memories are inactive, Light Sleep mode can then be invoked more frequently and for longer periods of time (extend the

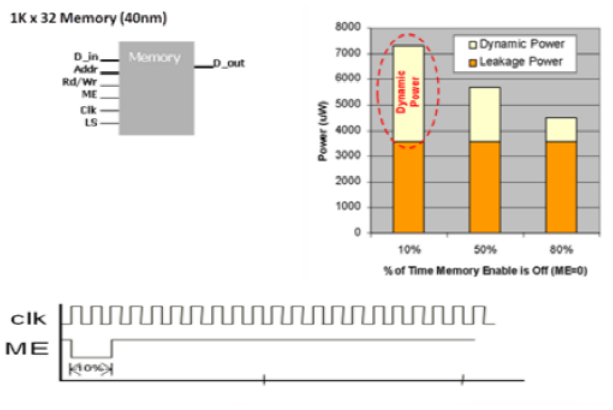


Figure 5.6: Light Sleep Memory Gating

time LS=1). Light Sleep Memory Gating can be used to reduce the leakage power consumption in the memories as depicted in the power chart below.

5.7 Power Analysis

To meet budget constraints, SoC designers need to be able to estimate power consumption in the early stages of the design cycle. However, given the lack of information about simulation vectors, technology libraries and decisions taken for synthesis and routing, inferring power consumption becomes a complex task. PowerPros Power Analysis gives a user insight into power consumption early on in the design cycle.

5.8 Calypto DataBase

As device sizes are shrinking and chip capacity is increasing, designing these 100 million transistor SoC is becoming a challenge. This is causing a paradigm shift in the EDA design flow causing the designers to move up in the abstraction level. This has warranted new EDA tools in the system-level design and verification arena. The goal of Calypto is to provide a suite of system-level tools. The first tool will be a system-level equivalence checker. The next will be the tool for the system-level virtual prototyping. All these tools will be build on a robust and common infrastructure. Figure 1 gives a top-level overview of the Calypto technologies. The front-end engine will read in the designs at the System Level (SL) or register transfer level (RTL), extract the hardware content from the description and populate the central database called Calypto DataBase(CDB).

Chapter 6

PowerPro

This chapter will provide the detailed information about PowerPro product designed by Calypto Design Systems. PowerPro is an automated RTL power optimization solution that dynamically reduces power with little or no impact on timing or area.

6.1 Introduction

PowerPro CG (for PowerPro Clock Gating) is an Automated RTL Power Optimization Solution. PowerPro CG has reduced power by up to 60% in customers RTL designs. By working at higher levels of abstraction PowerPro has increased opportunities to save power. Power savings from PowerPro are complimentary and cumulative to downstream tools that operate at the combinational gate level.

The following figure highlights the design flow with PowerProCG:

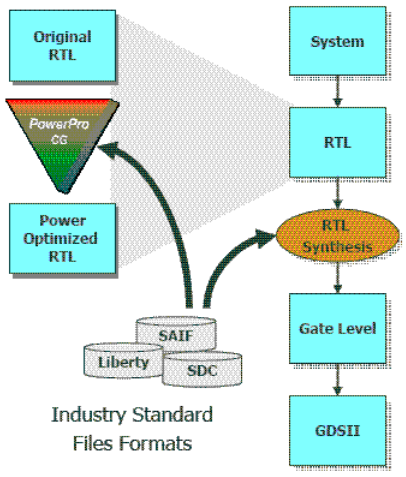


Figure 6.1: PowerPro flow

Following are the inputs to the tool:

1. A Design file.PowerPro currently supports design descriptions written in Verilog or VHDL.
2. SAIF (Switching Activity Interchange Format) which contains switching activity information.
3. SDC (Synopsys Design Constraints) which contains timing information.
4. Liberty files which contains the information about cell library.

6.2 Basic Concepts Used

PowerPro aims to gate (or shut off) activity where a clock enables data to be latched into a register or latch. By disabling clock to registers selectively, the output of the registers remain at the same logical value, and hence toggle activity is temporarily suspended. This reduces power dissipation in devices for the duration for which clock is gated not only in the registers but also in the combinational logic driven by the register outputs.

PowerPro uses clock gating efficiency to guide designers towards a lower power implementation. It ranks design registers by clock gating efficiency and accepts those transformations that result in significant improvement in clock-gating efficiency. Having such some knowledge of the design will point to areas where greater power savings maybe possible. A good example of this is low efficiency data-path registers. The intrinsic limit for improving clock gating efficiency is a function of each design block. A high-speed Ethernet IPv6 interface block for example, may have little opportunity to for clock gating. An efficiency of 10-15% may be near optimal under typical traffic scenarios. The actual power saved by clock gating is also dependent on the implementation of the logic. The power reduction is directly effected by the size of the combinational logic following the clock-gated register. The greater the logic fanning-out, the larger is the power savings.

PowerPro generates the enable signal by automatically analyzing the design for sequential clock gating opportunities. It creates the related logic (enable logic) for the concerned enable signal generated, thereby eliminating the effort required by the designer in manually identifying sequential clock gating opportunities and also reducing the possibility of introducing functional errors. The sequential enable signals along with the corresponding enable logic can be automatically patched into the original RTL by PowerPro, thereby reducing the manual effort needed in patching them into the original RTL. The functional correctness of the PowerPro optimized RTL can be verified using SLEC which verifies the sequential optimizations by comparing the clock gated RTL to the corresponding original designs.

PowerPro CG delivers functionally correct power optimized RTL using a flow which needs minimal effort and intervention from the user. Further, PowerPro CG fits into a synthesis based design flow easily, thereby providing maximum dynamic power reduction with minimum pain for the user.

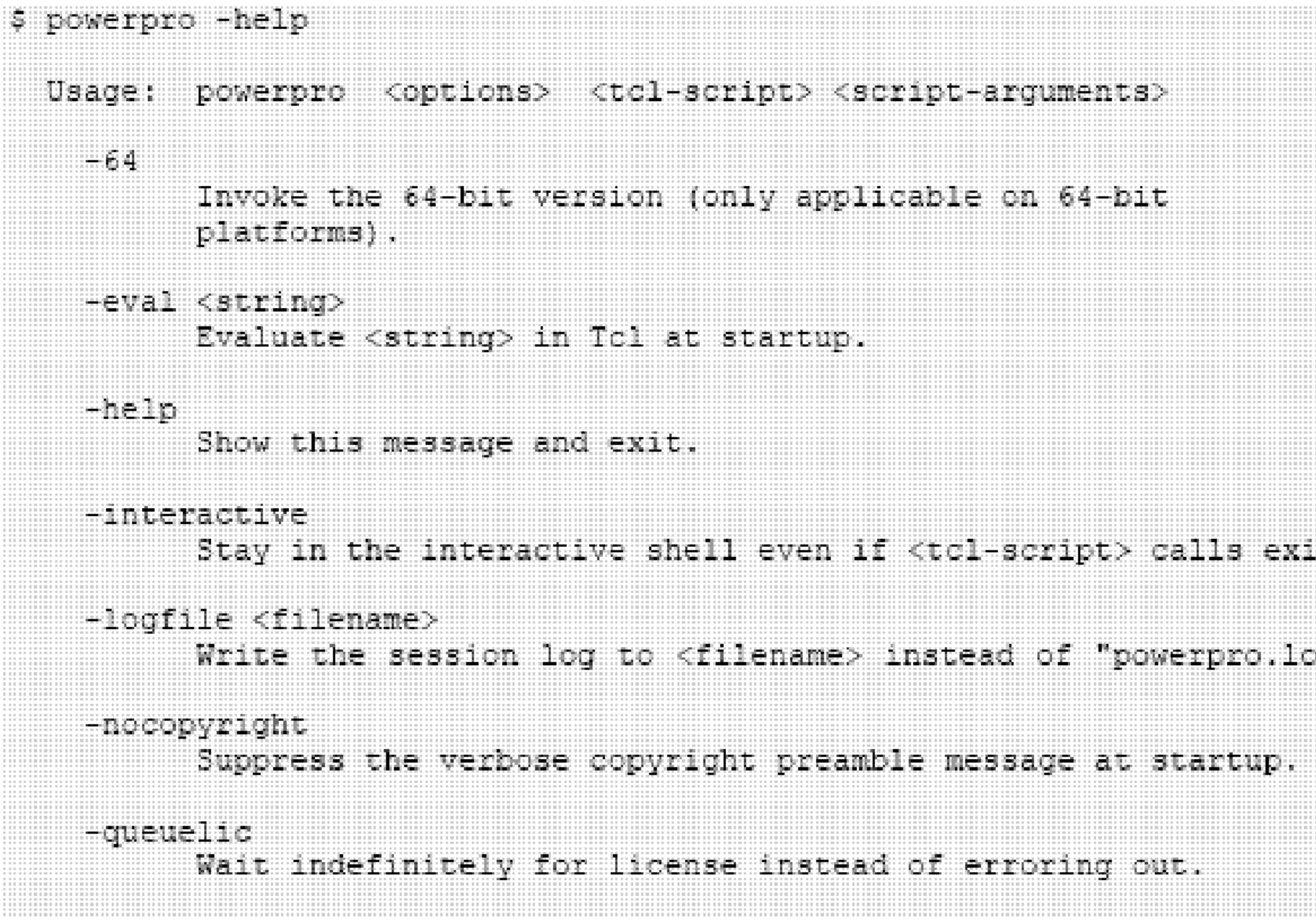
6.3 Running PowerPro

PowerPro is controlled through a Tcl setup file, typically run from the command line:

powerpro run.tcl

Running PowerPro with no setup file places PowerPro in to interactive mode, where commands may be entered within the tcl shell. Within PowerPro, the help command can be used to get detailed help on any of the commands or globals. In PowerPro, the term globals is used to refer to the global Tcl variables used to control PowerPro operation: help globals list all global, get_global ⟨ *name*⟩ returns the value of the global called ⟨ *name*⟩ and report_global lists the value of all globals. Both exit and quit can be used to end the PowerPro session. A complete list of PowerPro options is displayed when PowerPro is invoked with the help option.

Figure **??** tells the different running help options of PowerPro.



```
$ powerpro -help

  Usage:  powerpro  <options>  <tcl-script> <script-arguments>

    -64
          Invoke the 64-bit version (only applicable on 64-bit
          platforms).

    -eval <string>
          Evaluate <string> in Tcl at startup.

    -help
          Show this message and exit.

    -interactive
          Stay in the interactive shell even if <tcl-script> calls exi

    -logfile <filename>
          Write the session log to <filename> instead of "powerpro.lo

    -nocopyright
          Suppress the verbose copyright preamble message at startup.

    -queuelic
          Wait indefinitely for license instead of erroring out.
```

Figure 6.2: Running PowerPro

Chapter 7

Methodology

7.1 Ramp up Session

- Ramp up Session:
  - Overview of PowerPro tool.
  - Tutorials for:
    - Perl
    - Verilog
    - C++
    - TCL
    - Vim and
    - Sed/Awk
  - Introduction to CVS.
- QT
- Makefile
- hell scripting

7.2 Development

- This phase comprises of all the work that is done to upgrade the PowerPro GUI.
  - Building the source code of Qt4.

  - Altering the source code to make it Qt4 compatible.
  - Using the GNU debugger to find out the reason behind the error.

- Building the third party libraries which are needed to add some functionalities.
- Building the GUI source code again to see whether it is working or not.
- Similar kind of procedure was followed for the second project.

7.3 Status

- Daily stand up meeting is conducted for the progress report for 15-20min.
- Weekly status meeting of entire PowerPro team is held where everybody provide the status of work done for the week.

Chapter 8

Implementation Details

8.1 PreRequisites

8.1.1 Qt

Qt is a cross platform application framework. Qt has all the functionality needed to build applications with some amazing state-of-the-art graphical user interfaces. It fully object-oriented, can be extended easily, and allows true component programming. [5]

1996 marks the commercial introduction of Qt, from that point of time, it is the basis of many thousands of successful applications worldwide. The popular KDE Linux based desktop environment is also based on Qt .[5]

Qt is supported on the following platforms:

- MS/Windows – 95, 98, NT 4.0, ME, 2000, and XP
- Unix/X11 – Linux, Sun Solaris, HP-UX, Compaq Tru64 UNIX, IBM
- AIX, SGI IRIX and a wide range of others.
- Macintosh – Mac OS X
- Embedded – Linux platforms with framebuffer support. [5]

Qt is released in two editions:
Qt Enterprise Edition and the Qt Professional Edition are provided for commercial software development. They permit traditional commercial software distribution and include

free upgrades and Technical Support. The Enterprise Edition offers additional modules compared to the Professional Edition. [5]

The Qt Free Edition is available for Unix/X11, Macintosh and Embedded Linux. The Free Edition is for the development of Free and Open Source software only. It is provided free of charge under the terms of both the Q Public License and the GNU General Public License.

Different classes of QT used are

- QString:

  The QString class provides an abstraction of Unicode text and the classic C ”-terminated char array.

  In all of the QString methods that take const char * parameters, the const char * is interpreted as a classic C-style ”terminated ASCII string. It is legal for the const char * parameter to be 0. If the const char * is not ”terminated, the results are undefined. Functions that copy classic C strings into a QString will not copy the terminating ” character. The QChar array of the QString (as returned by unicode()) is generally not terminated by a ”. If you need to pass a QString to a function that requires a C ”-terminated string use latin1().

  A QString that has not been assigned to anything is null, i.e. both the length and data pointer is 0. A QString that references the empty string (””, a single ” char) is empty. Both null and empty QStrings are legal parameters to the methods. Assigning (const char *) 0 to QString gives a null QString. For convenience, QString::null is a null QString. When sorting, empty strings come first, followed by non-empty strings, followed by null strings. We recommend using if (!str.isNull()) to check for a non-null string rather than if (!str); see operator!() for an explanation. [5]

- QMessageBox:

  The QMessageBox class provides a modal dialog with a short message, an icon, and some buttons. Message boxes are used to provide informative messages and to ask simple questions.

QMessageBox provides a range of different messages, arranged roughly along two axes: severity and complexity.

For message boxes that ask a question as part of normal operation. Some style guides recommend using Information for this purpose.

- Information:

  For message boxes that are part of normal operation.

- Warning:

  For message boxes that tell the user about unusual errors.

- Critical:

  For message boxes that tell the user about critical errors. The message box has a different icon for each of the severity levels.

- QPushButton:

  The QPushButton widget provides a command button.

  The push button, or command button, is perhaps the most commonly used widget in any graphical user interface. Push (click) a button to command the computer to perform some action, or to answer a question. Typical buttons are OK, Apply, Cancel, Close, Yes, No and Help.

  A command button is rectangular and typically displays a text label describing its action. An underlined character in the label (signified by preceding it with an ampersand in the text) indicates an accelerator key, e.g.

  $$QPushButton * pb = newQPushButton("\&Download", this); \tag{8.1}$$

  In this example the accelerator is Alt+D, and the label text will be displayed as Download. [5]

- QRadioButton:

The QRadioButton widget provides a radio button with a text or pixmap label.

QRadioButton and QCheckBox are both option buttons. That is, they can be switched on (checked) or off (unchecked). The classes differ in how the choices for the user are restricted. Check boxes define ”many of many” choices, whereas radio buttons provide a ”one of many” choice. In a group of radio buttons only one radio button at a time can be checked; if the user selects another button, the previously selected button is switched off.

The easiest way to implement a ”one of many” choice is simply to put the radio buttons into QButtonGroup.

Whenever a button is switched on or off it emits the signal toggled(). Connect to this signal if you want to trigger an action each time the button changes state. Otherwise, use isChecked() to see if a particular button is selected.

Just like QPushButton, a radio button can display text or a pixmap. The text can be set in the constructor or with setText(); the pixmap is set with setPixmap(). [5]

- QCheckBox:

  The QCheckBox widget provides a checkbox with a text label.

  QCheckBox and QRadioButton are both option buttons. That is, they can be switched on (checked) or off (unchecked). The classes differ in how the choices for the user are restricted. Radio buttons define a ”one of many” choice, whereas checkboxes provide ”many of many” choices.

  A QButtonGroup can be used to group check buttons visually.

  Whenever a checkbox is checked or cleared it emits the signal toggled(). Connect to this signal if you want to trigger an action each time the checkbox changes state.

You can use isChecked() to query whether or not a checkbox is checked.

Warning: The toggled() signal can not be trusted for tristate checkboxes.

In addition to the usual checked and unchecked states, QCheckBox optionally provides a third state to indicate "no change". This is useful whenever you need to give the user the option of neither checking nor unchecking a checkbox. If you need this third state, enable it with setTristate() and use state() to query the current toggle state. When a tristate checkbox changes state, it emits the stateChanged() signal.

Just like QPushButton, a checkbox can display text or a pixmap. The text can be set in the constructor or with setText(); the pixmap is set with setPixmap(). [5]

- QWidget:

  The QWidget class is the base class of all user interface objects.

  The widget is the atom of the user interface: it receives mouse, keyboard and other events from the window system, and paints a representation of itself on the screen. Every widget is rectangular, and they are sorted in a Z-order. A widget is clipped by its parent and by the widgets in front of it.

  A widget that isn't embedded in a parent widget is called a top-level widget. Usually, top-level widgets are windows with a frame and a title bar (although it is also possible to create top-level widgets without such decoration if suitable widget flags are used). In Qt, QMainWindow and the various subclasses of QDialog are the most common top-level windows.

  A widget without a parent widget is always a top-level widget.

  Non-top-level widgets are child widgets. These are child windows in their parent widgets. You cannot usually distinguish a child widget from its parent visually. Most other widgets in Qt are useful only as child widgets. (It is possible to make, say, a button into a top-level widget, but most people prefer to put their buttons inside other widgets, e.g. QDialog). [5]

8.1.1.1 Nlview

The Nlview widget is a GUI building block that generates and displays schematics. The input data is a netlist-level description of the connectivity and optional engineering data like critical paths, time or signal values, etc. [6]

The Interface is a C-interface.

If you want to use Nlview, you need to write some "glue code" that implements all the functions declared in the interface between database and schematic. This manual explains to do steps.

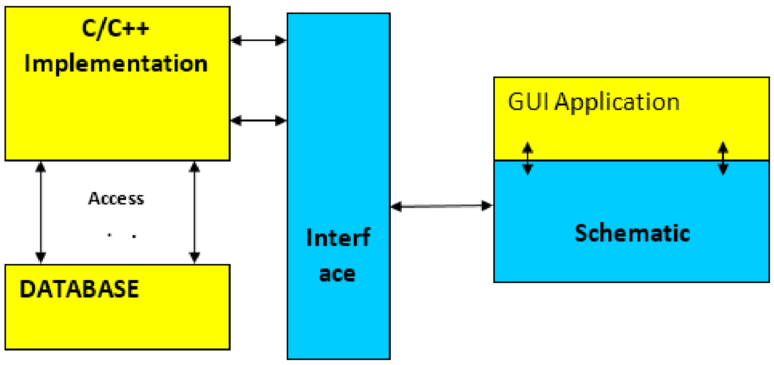


Figure 8.1: Integration Overview

The picture shows interface's part in the call chain: The GUI controls the Nlview widget by sending set of tcl commands, like more ..., less ..., loadmodule, etc. This will trigger certain interface C functions to be called, because Nlview must query the connectivity of the underlying Database. The interface implementation must respond to all the C functions defined in the interface implementation. After Nlview has finished querying the Database, it will respond to the tcl command by loading/unloading the requested components into/from the Nlview widget automatically.[6]

8.1.1.2 QScintilla

QScintilla is a port to Qt. With the features found in standard text editing components, QScintilla has features especially useful when editing and debugging source code. These features include support for syntax styling call tips, error indicators, code completion as well as syntax styling. The selection margin can contain markers like those used in debuggers to indicate breakpoints and the current line. Styling choices are more open than with many editors, allowing the use of proportional fonts, bold and italics, multiple foreground and background colors and multiple fonts.

8.1.1.3 Qwt

Qwt is library accessible for Qt. It gadgets for the Technical Applications and is an arrangement of custom Qt gadgets, GUI Components and utility classes which are basically helpful for projects with a specialized foundation. Adjacent to a 2D plot gadget it gives scales, sliders, dials, compasses, thermometers, haggles to control or show values, exhibits, or scopes of sort twofold.

8.1.2 SPEF- The Standard Parasitic Exchange Format

This chapter briefly discusses the project and the various domains in which work has been done. Detailed description of the work done in each of the domain follows in subsequent chapters.

8.1.2.1 Overview Of Project

The Title of the second project is: Reduced Parasitics Model from SPEF. Power management has emerged as a major design challenge in chip design today. The new importance of power consumption is not only coming from the increased proliferation of mobile devices, where extending battery life offers a huge competitive edge, but also from databanks and servers, where the infrastructure for providing power, cooling, and heat dissipation is getting very costly and outright prohibitive. Traditional chip design methodologies where power optimization is usually performed during physical design are not acceptable anymore, as they result in only a 10 15 percent improvement in power consumption.

Early power analysis/estimation is essential for meeting the project requirements and schedule. Below 28nm Liberty files, we have seen that there is no WLM (Wire-load) defined in the library. In that case PowerPro Gate level accuracy struggles. To meet this expectation, we need to support the physical design process node information to extract the required data to get correct CAPACITANCE .

Standard Parasitic Exchange Format (SPEF) is one of the key inputs to RTL level Power Analysis tool to maintain better Gate level Correlation. Resistance, capacitance and inductance of wires in a chip are known as parasitic data. SPEF is used for delay calculation and ensuring signal integrity of a chip which eventually determines its speed of operation.

SPEF files are usually very large and reading in large SPEF file can have considerable runtime/memory penalty. Reading large SPEF file to extract information about creating SPEF model is thus not desirable One way to solve this problem is to read SPEF file once and extract all information required by PowerPro and store it as a file. In subsequent runs PowerPro can read in the file representing extracted information instead of large SPEF file.The file representing extracted information from SPEF file would be an encrypted file so that nobody can know from the file about its content.

The platform used is Linux with C++ as the programming language. The tool has a Tcl interface.

In addition to this project Reporting of indented Hierarchies using the Tree Data Structure from the SPEF,FSDB and SAIF Files was also accomplished.

8.1.2.2 Introduction to Parasitics Data

Resistance, inductance and capacitance of wires in a chip are known as parasitic data.

It is used for delay calculation and ensuring signal integrity of a chip which eventually determines its speed of operation.

8.1.2.2.1 Different Formats of Parasitics Data

8.1.2.2.1.1 SPF–Standard Parasitic Format SPF is a Cadence Design Systems standard for defining net list parasitic data. DSPF and RSPF are the two forms of SPF; the term SPF itself is sometimes used (or misused) to represent parasitic data in general. DSPF and RSPF both represent parasitic information as an RC network.

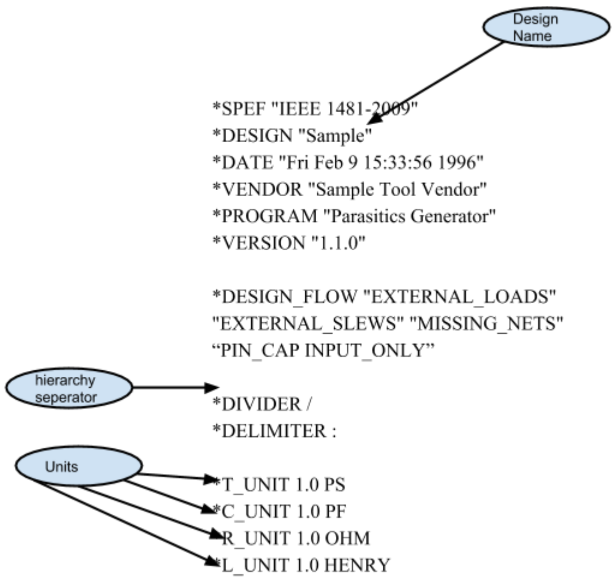
8.1.2.2.1.2 RSPF–Reduced Standard Parasitic Format RSPF represents each net as an RC "pi" model, which consists of an equivalent near" capacitance at the driver of the net, an equivalent "far" capacitance for the net, and an equivalent resistance connecting these two capacitance's. Each net has a single "pi" network for the network, regardless of how many pins are on the net. In addition to the pi network, RSPF causes the Prime Time tool to calculate Elmo re delay for every pin-to-pin interconnect delay.

8.1.2.2.1.3 SPEF–Standard Parasitic Exchange Format SPEF is an Open Verilog Initiative (OVI)–and now IEEE–format for defining net list parasitic. SPEF is NOT identical to the SPF format, although it is used in a similar manner. Like the SPF format, SPEF includes resistance and capacitance parasitic. Also like the SPF format, SPEF can represent parasitic in detailed or reduced (pi-model) forms, with the reduced form probably being more commonly used. SPEF also has a syntax that allows the modeling of capacitance between different nets, so it is used by the Prime Time SI (crosstalk) analysis tool. SPEF is smaller than SPF and DSPF because the names are mapped to integers to reduce file size.

8.1.2.2.1.4 SBPF – Synopsis Binary Parasitic Format SBPF is a Synopsis binary format supported by Prime Time. Parasitic data converted to this format occupies less disk space and can be read much faster than the same data stored in SPEF format. You can convert parasitic to SBPF, by reading them in and then writing them out with the write_parasitic -format SBPF command.

8.1.2.2.2 Introduction About SPEF Syntax General Syntax A typical SPEF file will have 4 main sections: a header section, a name map section, a top level port section and the main parasitic description section. Generally, SPEF keywords are preceded with a *. For example, *R_UNIT, *NAME_MAP and *D_NET. Comments start anywhere on a line with // and run to the end of the line. Each line in a block of comments must start with //.

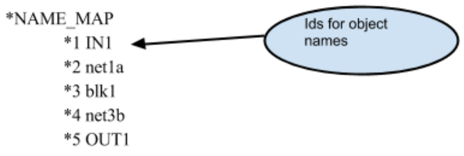
8.1.2.2.2.1 Header Information The header section is 14 lines containing information about: the design name, the parasitic extraction tool, naming styles and units. When reading SPEF, it is important to check the header for units as they vary across tools. By default, SPEF from Astor will be in pf and k Ohm while SPEF from Star-RCXT



```
*SPEF "IEEE 1481-2009"
*DESIGN "Sample"
*DATE "Fri Feb 9 15:33:56 1996"
*VENDOR "Sample Tool Vendor"
*PROGRAM "Parasitics Generator"
*VERSION "1.1.0"

*DESIGN_FLOW "EXTERNAL_LOADS"
"EXTERNAL_SLEWS" "MISSING_NETS"
"PIN_CAP INPUT_ONLY"

*DIVIDER /
*DELIMITER :

*T_UNIT 1.0 PS
*C_UNIT 1.0 PF
*R_UNIT 1.0 OHM
*L_UNIT 1.0 HENRY
```

Figure 8.2: SPEF Header

will be in ff and Ohm.



```
*NAME_MAP
        *1 IN1
        *2 net1a
        *3 blk1
        *4 net3b
        *5 OUT1
```

Figure 8.3: Name Map Section

8.1.2.2.2.2 Name Map Section To reduce file size, SPEF allows long names to be mapped to shorter numbers preceded by a *. This mapping is defined in the name map section. For example:

- *NAME_MAP
- *509 F_C_EP2
- *510 F_C_EP3
- *511 F_C_EP4
- *512 F_C_EP5
- *513 TOP/BUF_ZCLK_2_pin_Z_1

- *514 TOP/BUF_ZCLK_3_pin_Z_1
- *515 TOP/BUF_ZCLK_4_pin_Z_1

Later in the file, F_C_EP2 can be referred to by its name or by *509. Name mapping in SPEF is not required. Also, mapped and non-mapped names can appear in the same file. Typically, short names such as a pin named A will not be mapped as mapping would not reduce file size. You can write a script will map the numbers back into names. This will make SPEF easier to read, but greatly increase file size.

8.1.2.2.2.3 Port Section The port section is simply a list of the top level ports in a design. They are also annotated as input, output or bi-direct with an I, O or B. For example:

- *PORTS
- *1 I
- *2 I
- *3 O
- *4 O
- *5 O
- *6 O
- *7 O
- *8 B
- *9 B

8.1.2.2.2.4 Internal Section Each extracted net will have a *D_NET section. This will usually consist of a *D_NET line, a *CONN section, a *CAP section, *RES section and a *END line. Single pin nets will not have a *RES section. Nets connected by abutting pins will not have a *CAP section.

*D_NET reg control_top/GRC/n13345 1.94482

*CONN

```
*I reg control_top/GRC/U9743:E I *C 537.855 9150.11 *L 3.70000

*I reg control_top/GRC/U9409:A I *C 540.735 9146.02 *L 5.40000

*I reg control_top/GRC/U9407:Z O *C 549.370 9149.88 *D OR2M1P

*CAP

1 reg control_top/GRC/U9743:E 0.936057

2 reg control_top/GRC/U9409:A reg control_top/GRC/U10716:Z 0.622675

3 reg control_top/GRC/U9407:Z 0.386093

*RES

1 reg control_top/GRC/U9743:E reg control_top/GRC/U9407:Z 10.7916

2 reg control_top/GRC/U9743:E reg control_top/GRC/U9409:A 8.07710

3 reg control_top/GRC/U9409:A reg control_top/GRC/U9407:Z 11.9156

*END
```

The *D_NET line tells the net name and the net's total capacitance. This capacitance will be the sum of all the capacitance s in the *CAP section.

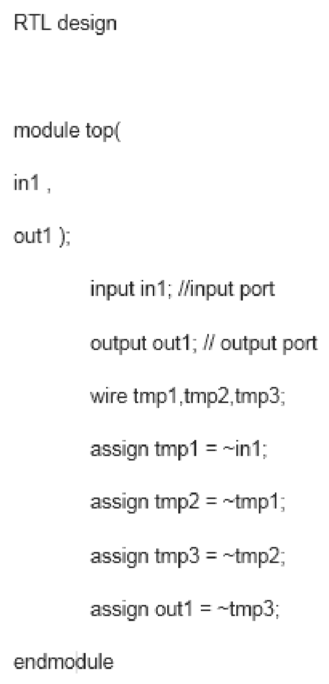
8.1.2.2.2.5 Internal Section Values The above examples show a single parasitic value for each capacitor or resistor. It is up to the parasitic extraction and delay calculation flow to decide which corner this value represents. SPEF also allows for min:typ:max values to be reported:

```
1 reg control_top/GRC/U9743:E 0.936057:1.02342:1.31343
```

The IEEE standard requires either 1 or 3 values to be reported. However, some tools will report min:max pairs and it is expected that tools may report many corners (corner1:corner2:corner3:corner4) in the future.

8.1.3 Inverter Chain

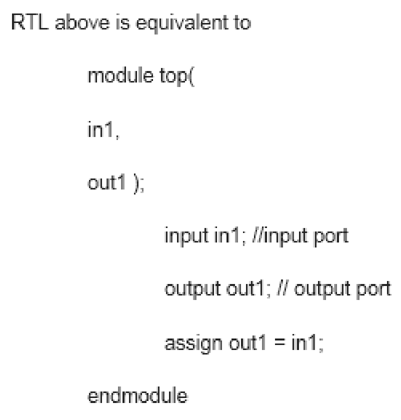
The net list might have series of inverters. The goal is to get rid of inverter pairs. Since goal is to generate smallest functionally correct net list, buffers and inverter pairs are redundant.



```
RTL design

module top(
in1 ,
out1 );
        input in1; //input port
        output out1; // output port
        wire tmp1,tmp2,tmp3;
        assign tmp1 = ~in1;
        assign tmp2 = ~tmp1;
        assign tmp3 = ~tmp2;
        assign out1 = ~tmp3;
endmodule
```

Example:

In this example all the four inverters are redundant. Because in1 and out1 are same. So removing all the four inverters will save the power. RTL above is equivalent to:



```
RTL above is equivalent to
        module top(
        in1,
        out1 );
                input in1; //input port
                output out1; // output port
                assign out1 = in1;
        endmodule
```

So the purpose of inverter chain tutorial is find all the redundant buffers and inverters and remove them.

8.1.3.1 Need

- To have understanding of database, its implementation and its application interfaces. It is for internal use by the R&D of Calypto only.
- To have understanding of traversals of database (DFS,BFS).

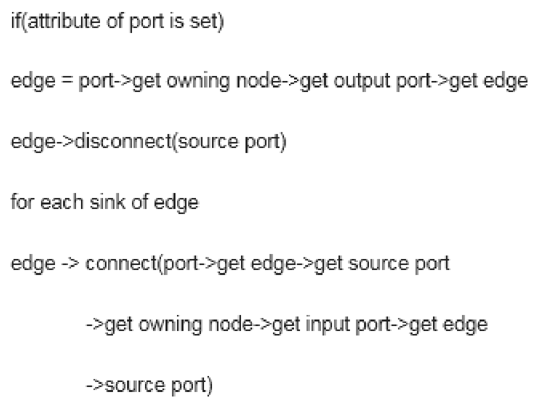
- To have knowledge of different APIs which can be used to retrieve data from database.
- To have basic understanding of port, edge, instance ,nodes and different APIs related to them and relationship among them.
- To have knowledge about node types.
- To have knowledge about callbacks.

8.1.3.2 Steps

1. Analysis
   (a) Perform DFS on nodes.
   (b) While doing DFS check type of each node.
   If(inverter node)
   Create and set attribute of node.
   (c) Again perform DFS on ports

   ```
   If(input Port)
   If(attribute of owning node is set)
   If(previous input port is not marked)
   Create and set the attribute for port.
   ```

2. Transformation
   (a) for each input port in view



```
if(attribute of port is set)
edge = port->get owning node->get output port->get edge
edge->disconnect(source port)
for each sink of edge
edge -> connect(port->get edge->get source port
          ->get owning node->get input port->get edge
          ->source port)
```

3. Cleanup
    (a) Remove attribute which are created for nodes and ports.
    (b) Remove the deadcode.

```
testbench
    testbench.dut
        testbench.dut.ct
    testbench.read_input_file
    testbench.simulate
testbench.fopen
```

8.1.4 User Characteristics

The end user is expected to be Linux/Unix literate and be able to use Linux commands. The user is expected to be aware of Object Oriented Programming language C++ so that he is able to write the code and can run the tool and can verify the results. The user is expected to be aware of integrated circuits and power optimization.

8.1.5 System Requirements

Supported Platforms and Operating Systems

- Support Classifications: ==========

  "Certified" means that the platform is extensively tested by Calypto for each and every release. "Binary-compatible" means that the platform satisfies the operating requirements.

- Operating Systems: ==========
  - Red Hat Linux 9 Binary-compatible
  - Red Hat Enterprise Linux 3 Certified
  - Red Hat Enterprise Linux 5 Binary-compatible

- Platforms: =========
  - Intel Pentium III Binary-compatible
  - Intel Centrino Binary-compatible
  - Intel Xeon (32-bit) Certified
  - Intel Xeon (64-bit) Certified
  - Other XBG(IA32) Binary-compatible
  - AMD Athlon/Duron Binary-compatible
  - AMD Opteron (32-bit) Binary-compatible
  - AMD Opteron (64-bit) Binary-compatible.

8.1.6 Build/Debug Environment of the Tool

8.1.6.1 Concurrent Version Systems

CVS is a version control system. It is used to record the history of the source files. Bugs can creep in when software is modified, and may not be detected until a long time after the modification is made. With CVS, you can retrieve older versions to find which change caused the bug.

CVS can also help when a project is being worked on by multiple people, where overwriting each others changes is easy to do. CVS solves this problem by having each developer work in his/her own directory and then instructing CVS to merge the work when each developer is done.

While CVS stores individual file history in the same format as RCS, it offers the following significant advantages over RCS:

- It can run scripts which you can supply to log CVS operations or enforce site-specific policies.
- Client/server CVS enables developers scattered geographically or slow modems to function as a single team. The version history is stored on a single central server and the client machines have a copy of all the files that the developers are working on. Therefore, the network between the client and the server must be up, to

perform CVS operations (such as check-ins or updates) but need not be up to edit or manipulate the current versions of the files. Clients can perform all the same operations which are available locally.

- In cases where several developers or teams want to maintain their own versions of the files due to geographical locations and/or policies, CVSs vendor branches can import a version from another team (even if they don't use CVS), and then CVS can merge the changes from the vendor branch with the latest files if that is what is desired.

- Unreserved checkouts, allowing more than one developer to work on the same files at the same time.

- CVS provides a flexible module database that provides a symbolic mapping of names to components of a larger software distribution. It applies names to collections of directories and files. A single command can manipulate the entire collection.

- CVS servers run on most UNIX variants, and clients for Windows NT/95, OS/2 and VMS are also available. CVS will also operate in what is sometimes called server mode against local repositories on Windows 95/NT [3]

8.1.7 Bugzilla

It is an open source, commercial "Bug-Tracking System" which allows an individual or groups of developers to keep track of bugs in the products code they are developing. Generally, commercially available defect-tracking software vendors charge enormous licensing fees. But, Bugzilla is free of cost. Bugzilla system is used to track the issues with the external customer products. This is also used to track the internal products and processes e.g. licensing, training courses, cds tools. Doc tools, etc.

This is a repository for filing the product change requests. The product change requests can be of two types:

- Bug tracking

- Enhancement request

8.1.8 Software Requirements

8.1.8.1 Object Oriented Concepts

- Objects:

  Objects are the basic run-time entities in an object-oriented system. Programming problem is analyzed in terms of objects and nature of communication between them. When a program is executed, objects interact with each other by sending messages. Different objects can also interact with each other without knowing the details of their data or code.[4]

- Classes:

  A class is a collection of objects of similar type. Once a class is defined, any number of objects can be created which belong to that class.[4]

- Data Abstraction:

  Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes.[4]

- Encapsulation:

  Storing data and functions in a single unit (class) is encapsulation. Data cannot be accessible to the outside world and only those functions which are stored in the class can access it.[4]

- Polymorphism:

  It is the ability of objects belonging to different types to respond to method calls of the same name, each one according to an appropriate type-specific behavior. The programmer (and the program) does not have to know the exact type of the object in advance, so this behavior can be implemented at run time. This is called late binding or dynamic binding.[4]

- Inheritance:

  Inheritance is the process by which objects can acquire the properties of objects of other class. In OOP, inheritance provides reusability, like, adding additional features to an existing class without modifying it. This is achieved by deriving a new class from the existing one. The new class will have combined features of both the classes.[4]

8.1.8.2 Makefile

It is simply a way of associating short names (called targets) with a series of commands to execute when the action is requested. For instance, a common makefile target is "clean," which generally performs actions that clean up after the compiler–removing object files and the resulting executable.

Make, when invoked from the command line, reads a makefile for its configuration. If not specified by the user, make will default to reading the file "Makefile" in the current directory. Generally, make is either invoked alone, which results in the default target, or with an explicit target. (In all of the below examples, % will be used to indicate the prompt.)

To execute the default target:
% make

To execute a particular target, such as clean:
% make clean

Besides giving you short build commands, make can check the timestamps on files and determine which ones need to be recompiled; we'll look at this in more detail in the section on targets and dependencies. Just be aware that by using make, you can considerably reduce the number of times you recompile.

8.1.8.3 Tcl

Tcl (originally "Tool Command Language", but nonetheless conventionally rendered as "Tcl" rather than "TCL"; and pronounced "tickle") is a scripting language created by John Ousterhout.Tcl provides generic programming facilities, such as variables and loops and procedures that are useful in a variety of applications. Furthermore, Tcl is embeddable. Its interpreter is a library of C procedures that can easily be incorporated into applications, and each application can extend the core Tcl features with additional commands for that application.

Some important features of Tcl are:

- Everything is a command, including language structures.
- Everything can be dynamically redefined and overridden.
- All data types can be manipulated as strings, including code.
- Extremely simple syntactic rules.
- Simple exception handling using exception code returned by all command executions.
- Readily extensible, via C, C++, Java, and Tcl.
- Close integration with windowing (GUI) interface.
- Easy to maintain code. Tcl scripts are often more compact and readable than functionally equivalent code in other languages. [8] Functional programming can easily be done in Tcl, as higher-order functions or functional abstractions are built into the language, though it is not widely used for this purpose. [8]

The most important advantages of using Tcl include the following:

- Fast Development:

  Tcl makes the applications to be implemented 5-10x faster than with other languages, specifically if the application involves graphical user interfaces, integration or integration. If an application is built in Tcl, it can be easily evolved rapidly to meet changing needs.

- Graphical User Interfaces:

Tcl provides facilities for creating graphical user interfaces which are incredibly simple as well as remarkably powerful with its Tk toolkit. For example, Tk canvas widget keeps it easy to make displays with graphics, also, it provides facilities which are powerful such as bindings and tags. Tk has been designed from the ground up for the rapid development inherent in dynamic programming languages like Tcl.

- Easy to Learn:

  Being a simple language, it is very simple to learn. Programmers can learn Tcl and produce some interesting applications in just couple of few hours or days. For casual programmers also, Tcl is quick to learn.

- Deployment:

  Dynamic languages make deployment hard as the need is to get both the application scripts and interpreter onto the target machine. Many dynamic languages do provide tools to "compile" every line into a single executable (which Tcl is already having from 1993).

- Testing:

  Being an ideal language to use for automated software and hardware testing, it is the dominant language used for the purpose. Using Tcl, we can easily make connection to internal APIs of an application or testing hardware, invoking test functions, checking the results, and reporting errors. Using Tcl's interpreted implementation, creation of testscan be done very rapidly, and these tests can be saved easily as Tcl script files that can be reused for testing regression.

8.1.8.4 SED/AWK

SED" stands for Stream EDitor. Sed is a non-interactive editor, written by the late Lee E. McMahon in 1973 or 1974.[9]

Instead of altering a file interactively by moving the cursor on the screen (as with a word processor), the user sends a script of editing instructions to sed, plus the name of the file to edit (or the text to be edited may come as output from a pipe). In this sense, sed works like a filter – deleting, inserting and changing characters, words, and lines of text. Its range of activity goes from small, simple changes to very complex ones.[9]

However, SED lacks relative addressing (the ability to specify a line number to work on relative to the current line number) because it processes a file one line at a time and never backs up. Also, sed gives you no immediate verification that a command has altered your text in the way you actually intended.

AWKis an interpreted programming language that is included in most versions of UNIX. The name is derived from the initials of its creators – Alfred Aho, Peter Weinberger, and Brian Kernighan – who developed the language in 1977 and 1978. The language is
particularly designed for filtering and manipulating textual data.

It was developed from grep, C, and sed syntax, a combination that allows complex programs to be developed quickly. awk is frequently used for prototyping. [9]

awk (also written as Awk and AWK) is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line. awk comes with most Unix-based operating systems such as Linux, and also with some other operating systems, such as Windows 95/98/NT. [9]

An awk program is made up of patterns and actions to be performed when a pattern match is found. awk scans input lines sequentially and examines each one to determine whether it contains a pattern matching one specified by the user. When the matching pattern is found, awk carries out the instructions in the program. For example, awk could scan text for a critical portion and reformat the text contained in it according to the user's command. If no pattern is specified, the program will carry out the command on all of the input data.[9]

8.1.9 Tools Used

8.1.9.1 GDB

GDB is a debugger. A debugger is a tool which can help you find bugs in your code. It will allow you to follow your program as it executes to see what happens at each step. The program can be stopped on any line or at the start of any function and various types of information can be displayed, such as the values of variables and the sequence of function calls that got you where you are. If your program causes a segmentation fault, GDB will show you where it happened. Advanced users can alter the values of variables to experiment with temporary bug fixes and view the contents of the stack. [10]

The greatest advantage to using GDB within Emacs is that it will work with source code (.c file). Whenever execution of the program is stopped (by a breakpoint, a segmentation fault, or some other signal), Emacs will display the source code in a window and will mark the line on which it has stopped with a $= \rangle$ *symbol*.

We can also use Emacs to cut and paste commands, scan through GDB output and save it as a text file.[10]

It is a source-level debugger, capable of breaking programs at any specific line, displaying variable values, and determining where errors occurred. Currently, it works for C, C++, Fortran Modula 2 and Java programs. It allows you to see what is going on 'inside' another program while it executes – or what another program was doing at the moment it crashed. [10]

GDB can thus do four main kinds of things (plus other things in support of these) to help you catch bugs in the act: Start your program, specifying anything that might affect its behavior.

- Make your program stop on specified conditions.

- Examine what has happened, when your program has stopped.
- Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

The program being debugged can be written in Ada, C, C++, Objective-C, Pascal (and many other languages). Those programs might be executing on the same machine as gdb (native) or on another machine (remote). gdb can run on most popular UNIX and Microsoft Windows variants. [10]

8.2 Project Work

This project can be divided into several stages:

- Setting up the environment: cshrc, aliases, CVS, creating the sandbox.
- Compiling the GUI source code with Qt4 and make changes required in the code.
- Compile the third parties to integrate them with the GUI source code.
- Add the binaries of third party libraries after compilation.
- Try to achieve the new schematic, reports and remaining parts of GUI one by one.

Following steps are used to accomplish these stages:

- Used the utility qt3to4 to make changes needed in the PowerPro GUI source code.
- Compiled the source code of PowerPro GUI using Qt4.7.4 which was written for Qt3.3.6
- Found the errors in the source code and made the changes in the code according to the needs in Qt4.
- Used GNU debugger to sort out the issues in building.
- Repeated all the same methodology for 64 bit compilation.

PowerPros GUI is developed using Qt. So, there is a need of building the source code of Qt first.

When Qt is compiled successfully, we try to compile the GUI source code using the qmake entity which is available in the directory qtbase of the directory in which qt is built.

If the source code of GUI is built successfully, it will make the binary libdesignview.so available in work1derived_objreleaseplatformlinuxlib.

There are some third party libraries that provide some functionalities and are needed to be built.

Use GNU debugger to debug the error.

When we make changes in the GUI source code, it gives some undefined symbols while we try to build the GUI source code.

C++filt is the utility that gives the information about the undefined symbol in the code. It gives the function as well as the file where the problem exists.

Chapter 9

Conclusion

This project helped me in having a better understanding of the various Object Oriented concepts and practically implementing them in code. This project also helped me understand why there is necessity of following several coding guidelines in the organization. These several coding guidelines helps organization to achieve better performance of products and also help developers to learn and develop new standards. I also got a chance to work in scripting languages like Perl and TCL.

I learnt the practical usage of LEX and YACC in parsing of a SAIF file. I also learnt the usages of gdb (debugger) and RTL Compiler. This project helped me to improve my technical skills and also helped me to gain an on-hand experience on a live project.

After working on the tool during the product release I have realized practical importance of how quality must be ensured by following the Software Development Life Cycle. I also understand the importance of Documentation for the product.

9.1 Path Followed

- Tutorials on Linux shell, VIM, Tcl commands, sed, awk and Verilog.
- Setting up eclipse and workspace. Also getting familiar with the concepts of CVS (Concurrent Versions System).
- Understanding technologies involved in Calyptos products e.g. Power Analysis, Power optimization, etc.
- Understanding concepts of PowerPro using tutorials on PowerPro

– Compilation of GUI source code with the Qt4.

– Making changes in the code to make it comply with Qt4.

– Html Help Pages are now available in the reports available with PowerPro as well as PowerPro Designer.

The work of the two projects which had been assigned to me is complete.

9.2 Work Completed

– The Qt miggration has been successfully completed which resulted in better Usability of the GUI.

– The second project on SPEF has also been successfully completed.

Chapter 10

References

1. Wikipedia for Graphical User Interface, QT, QT for Windows, SPEF.
2. OOP Concepts: Tutorials Point
3. NLView Tutorial: www.concept.de/nlview.html
4. TCl Tutorial: www.tcl.tk
5. SED Tutorial: www.grymoire.com/Unix/Sed.html
6. GDB Tutorial: www.gnu.org/s/gdb/
7. Calypto :: http://calypto.com/en/
8. Qt Website :: http://www.qt.io/
9. Migrating Qt :: http://doc.qt.io/qt-4.8/porting4.html
10. The Qt Blog :: https://blog.qt.io/blog/2015/09/08/qt-5-6-alpha-released/
11. Makefile Tutorial http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/
12. Verilog tutorial :: www.asic-world.com/verilog/veritut.html
13. Vim Tutorial :: http://www.openvim.com/
14. Tcl Tutorial :: https://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html
15. Html, Java Script and J Query tutorial: https://www.codecademy.com/
16. SPEF Files Explaination: http://vlsi.pro/readingspeffiles/
17. Reading a SPEF File: http://www.vlsi-expert.com/2010/08/howtoread-spef.html
18. Knowledge -based User Interface Migration by Melody Moore, Spencer Rugaber, and Phil Seaver.

19. Cross-Platform Development: Software that Lasts by Judith Bishop, Nigel Horspool.

Bibliography