LUND UNIVERSITY

Energy efficient Ericsson Many-Core Architecture (EMCA) IP blocks for 5G ASIC

ZILIN ZHANG
zi6171zh-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor:
Steffen Malkowsky
steffen.malkowsky@eit.lth.se
Ioannis Savvidis (Ericsson)
ioannis.savvidis@ericsson.com

Examiner:
Erik Larsson
erik.larsson@eit.lth.se

August 30, 2021

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Power consumption has become a leading concern for SoC aimed at 5G products that demand increased functionality, smaller form factors, and low energy footprint. For some EMCA IP blocks a hierarchical clock gating mechanism ensures coarse-grained power savings based on actual processing need but for many blocks this approach cannot be employed. This implies that these blocks have to rely on a high local clock gating efficiency to meet the set power requirements. For this purpose, designers have to manually analyze and optimize the blocks to improve the combinational and sequential clock gating. But this legacy flow is error-prone and time-consuming as it requires running long simulations to ensure the RTL changes have not introduced functional errors. The scope of this thesis is to evaluate and deploy a novel flow that features automatic power optimization along with integrated formal verification guarantees for bug-free RTL. The flow is applied on a set of EMCA IP blocks to reduce design efforts and produce energy efficient IPs even when time to market is the highest priority for a project. The thesis demonstrates that the researched flow can be easily integrated into the existing front end IP design process for production. For this purpose, several IP blocks have been tested and optimized to collect empirical data. The power optimizations have been verified all the way down to the pre-layout netlist level. On average, a reduction of 20% has been achieved for the dynamic power (observed range: 5.2% - 59.3%) with very low effort and minimal impact on area and timing.

Acknowledgments

This project is supported by Ericsson and oriented to the low-power Application Specific Integrated Circuit (ASIC) design. During the time I have been working on my thesis, a lot of people have given me endless help. Especially my supervisor at Ericsson Ioannis Savvidis, and I want to thank him for his constant guidance, positive motivation and tireless support. He has given me many important advice that has motivated me to complete this project. He has led me through all the technical difficulties all the way to the finalized result. Also, I want to thank Alok Kapoor Jain who has provided me continuous help with debugging problems and with technical issues. I want to thank the team in Mentor: Richard Langridge and Bo Janfalk, for their weekly presence at the feedback meetings. They have provided important advice at different stages of the project. I want to thank the people from Ericsson: Neerajnayan Balachandran, Enrique Cobo Jiménez, Roger Engberg, Venkata Jagannadha, Ionut Tolea and Dragos Dospinescu, who have patiently provided me technical supports. I want to thank the manager at Ericsson, Pierre Rohdin G, who has helped me to get the resources that I needed to start the project. His support has created a very conducive environment for the development of the project. I want to thank my supervisor at Lund University, Steffen Malkowsky, for his valuable input and dedication to the project. Thanks to Prof. Erik Larsson for his guidance in defining the project and finalizing the documents. Finally, I want to thank Fredrik Johansson who has given me valuable suggestions on writing.

Popular Science Summary

Fifth Generation (5G) technology, as a newly flourishing technology, has quickly taken over the market in the past five years. Many survey reports by the telecom companies have shown that the global mobile data traffic is growing significantly in each year. And this data traffic increase is largely due to the expansion of 5G networks. The power consumption introduced by this 5G expansion has shown a rapid increase that leads to a large carbon dioxide dissipation. Therefore, power optimization for 5G products has become one of the major hotspots in recent years. It is also an urgent concern to keep the operating costs and electricity bills under control for the telecom vendors.

For a generic 5G product, the clock-driven power is dominated because of high performance requirements of the product. EMCA Intellectual Property (IP) block is a block that consists of Ericsson developed digital signal processors. To improve the power efficiency of generic EMCA IP blocks, Clock Gating (CG) is the most commonly used method. CG is a low-power design technique that reduces dynamic power dissipation by removing redundant clock toggles. Ericsson has developed a flow for CG boost. This flow depends on Spyglass power analysis and manual clock gates insertion. This process is normally error-prone and time-consuming. The verification of the optimization needs to run many regressions to cleanup the functional bugs introduced during the optimization. Hence, we need to provide a competent flow that assures a functional error-free optimization with small efforts at the industrial level. Another challenge is that the optimization for generic EMCA IP blocks is usually usecase-specific. This means that the testcase that is used to optimize an IP block should be more common to the block.

This thesis proposed a flow that can bring together analysis, high-efficient optimization, and formally-verified automatic RTL generation to boost the clock gating efficiency at the late stages of the IP design process. This new flow introduced an efficient Electronic Design Automation (EDA) tool that can automatically implement clock gating in an ASIC design. This EDA tool in-built a

functionality equivalence checker that assures error-free optimization. With this tool, the proposed approach guarantees a safe optimization with very little efforts in optimization and verification. In the new flow, we proposed three optional strategies for the testcase selection procedure, which is decisive to the quality of power optimization for an IP block.

Four EMCA IP blocks have been tested and optimized by this new approach. The case study is carried out based on the three strategies developed in the flow. An average reduction of 20% has been achieved for the dynamic power for all tested blocks, which shows that the new flow has performed very well on analyzing and optimizing generic EMCA IP blocks. And, the flow is proved to be very competent for low-power IP design process at the industrial level.

List of Abbreviations

4G Fourth Generation.

5G Fifth Generation.

ASA Average Switching Activity.

ASIC Application Specific Integrated Circuit.

CC Clock Cycle.

CCG Combinational Clock Gating.

CG Clock Gating.

CGE Clock Gating Efficiency.

CPU Central Processing Unit.

CS Chip Select.

CTS Clock Tree Synthesis.

DCGE Dynamic Clock Gating Efficiency.

EDA Electronic Design Automation.

EMCA Ericsson Many-Core Architecture.

FF Flip Flop.

FIFO First-In, First-Out.

FinFET Fin Field Effect Transistor.

FSDB Fast Signal Data Base.

GCT Gated Clock Toggles.

GPU Graphics Processing Unit.

HDL Hardware Description Language.

IO Input Outout.

IP Intellectual Property.

MC Memory Count.

MOSFET Metal-Oxide Semiconductor Field-Effect Transistor.

MRPU Memory Read Port Utilization.

MWPU Memory Write Port Utilization.

OBCG Observability Based Clock Gating.

RTL Register Transfer Level.

RUC Read Utilization Count.

SA Switching Activity.

SAIF Switching Activity Interchange format.

SBCG Stability Based Clock Gating.

SCG Sequential Clock Gating.

SCGE Static Clock Gating Efficiency.

SDC Synopsys Design Constraints.

SLEC Sequential Logic Equivalence Checker.

SPEF Standard Parasitic Exchange Format.

TCL Tool Command Language.

TCT Total Clock Toggles.

UVM Universal Verification Methodology.

VCD Value Change Dump.

VHDL Very High-speed Integrated Circuit Hardware Description Language.

WUC Write Utilization Count.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This Chapter provides an overview and the general plan of this thesis. 5G technology has appeared on the horizon in recent years and will flourish in the market in the near future. A general 5G station consumes around three times more energy than a Fourth Generation (4G) station [4]. The report from Environmental Health Trust [4] predicted the total installed base of 5G small cells to be more than 13 million in 2050, and the expansion of 5G network expects 60 times increase on energy consumption between 2020 and 2030. This drastic power increase brought by 5G technology could add between 2.7 to 6.7 million tonnes of carbon dioxide equivalents per year by 2030. Hence, the goal to decrease the carbon footprint of 5G technology becomes significant in the era of information expansion. 5G products need to meet various requirements for different usages. For example the low-rate services are more low-power and low-cost prone, and the high-rate services usually cannot avoid high expense on power to satisfy the high-speed and reliability requirements [5]. It has been pointed out in a newly published report by Ericsson that with the exponentially growing data traffic, the 5G development is facing a crucial problem [1]. As we can see in figure 1.1, the predicted data traffic in 2024 will approximately double the global mobile data traffic in 2020. The power efficiency of 5G networks should be prioritized in order to reduce operational cost and the effect on the environment. The research on low-power ASIC design has recently been referred to as an important factor towards the challenges brought by the ever increasing performance on 5G products.

1.1 Overview

The ASIC design technology has undergone a series of innovations from the 1980s and forward. It started with the language-based design revolution in the 1980s, for example, Very High-speed Integrated Circuit Hardware Description Languages

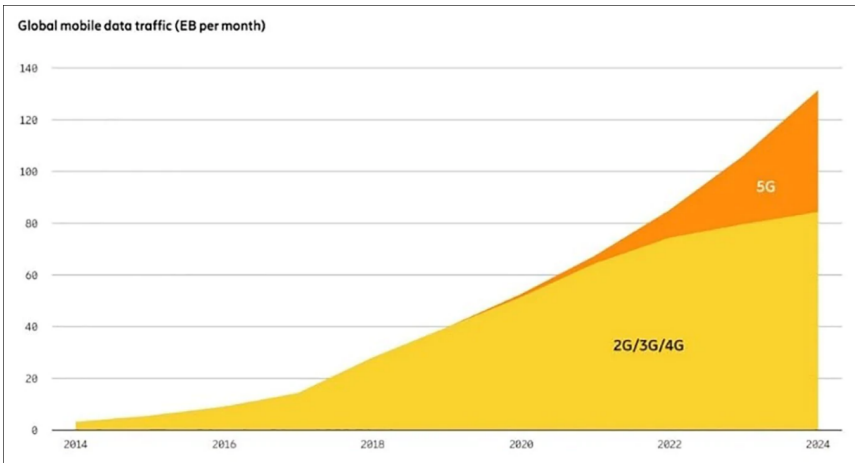


Figure 1.1: Global mobile data traffic outlook presented in Ericsson Mobility Report, June 2019 [1]

(VHDLs) development commenced in 1983, and continued with the development of reusing common IP block technology in the 1990s. In recent years, the technology of high-efficient designs with a compact structure has become mature and reliable. The attention has then shifted to the carbon footprint and life span of the design [6]. As the density of ASIC increases at an exponential rate according to Moore's law, it becomes even harder for the power saving techniques to catch up. Especially for technology nodes below 14nm, clock power is the most prominent contribution to power consumption because of the high clock frequency required in the high performance operation mode [7]. Clock power can be defined as the dynamic power related to clock signals. Therefore, clock power reduction can result in significant dynamic power savings. Dynamic power in turn can be classified as switching power and internal power. The static power is not the topic of interest in this thesis because, the latest technology nodes of inherently reduced leakage power are typically used in advanced 5G devices. For example, the Fin Field Effect Transistor (FinFET) technology has become the major solution that is adopted by integrated circuit manufacturers to guarantee a low leakage design [8].

Power optimization can be conducted on four levels which are architectural level, micro architectural level, gate level and transistor level. This thesis primarily focuses on the low power techniques at the micro architectural level to better budget power and to discover power bugs at the Register Transfer Level (RTL) stage. It is more convenient and fruitful to save power by means of coding modifications at the front end than by means of power optimization techniques at the back end.

EMCA can be defined as a growing number of ASICs that consist of both general purpose processors and Ericsson's own developed digital signal processors. For most EMCA IP blocks, a hierarchical clock gating mechanism ensures power savings. But this approach is not fit for all the blocks. For the blocks that cannot

be employed with hierarchical clock gating, a high local clock gating efficiency is needed to meet the power requirements. On this purpose, Ericsson has proposed a precise flow to characterize and optimize IP blocks [9]. Among other approaches, this flow uses Differential Energy Analysis technique to uncover power inefficiencies. It has been applied on IP blocks in Ericsson and has been verified as precise and reliable. But current flow necessitates that IP designers have to manually analyze and optimize the blocks to improve the combinational and sequential clock gating. This part of the flow is error-prone and time-consuming as it requires running long simulations to ensure that the RTL changes have not introduced functional errors. This can lead to a chasm between flow development and deployment particularly when time to market is the highest priority for a project, which prevents it from becoming a competitive and prolific solution in the industry.

The scope of this thesis is to evaluate and deploy a novel flow that features automatic power optimization along with integrated formal verification guarantees for bug-free RTL. The flow will be applied on a set of EMCA IP blocks to reduce design efforts and produce energy efficient IPs with minimal effort. To support this flow, four EMCA IP blocks are discussed and experimented using the automatic power analysis and optimization techniques provided in the flow. This flow uses a commercial tool named PowerPro together with Ericsson in-house tools to automatically analyze and optimize IP blocks on RTL level. PrimeTimePX is used as a signoff tool in the verification phase.

1.2 Problem description

5G is introducing a challenge on power consumption in System on a Chip designs. The power density in a chip is increasing exponentially as the design is becoming more compact. The life span and the power consumption of 5G products become vital as the high bandwidth wireless appliances evolve. Many low power protocols have been proposed to solve these issues, such as Narrow Band Internet of Things and Long Term Evolution for Machines. They provided solutions, such as operation at near-threshold voltages, voltage scaling and intelligent CG methodologies [10]. Because of the high CG frequency that 5G products are operating at, this thesis focuses on the CG methodology to reduce high clock power consumption in high performance devices. The previous flow focuses on discovering the CG opportunities in a design. Based on the uncovered power inefficiencies, design teams manually insert clock gates in the design. However, the redundancies passing through more than five clock cycles are very difficult to discover and hence to minimize. There need to be a high-efficient tool to uncover those intricate CG opportunities automatically.

This flow helps to optimize the design based on representative testcases. This flow is expected to facilitate the power quality of generic ASIC designs. Moreover, this flow does not require the design team to grasp the knowledge about the power

saving potential on their design.

1.3 Goals

The goal of this thesis is to propose how this flow can bring together analysis, optimization, and formally-verified automatic RTL generation to boost the clock gating efficiency at the late stages of the IP design process. Several case studies are implemented on the exsited IP blocks in Ericsson. Some of the sub-modules of those blocks are analyzed, and automatically optimized in the front end stage. The pre Clock Tree Synthesis (CTS) netlists, pre layout netlists are used in this evaluation. Power metrics are extracted to evaluate and verify the improvement of the optimized designs. The low power front end consists of four steps.

1. The first step is to choose the right testcase to conduct power optimization on a design by quickly visualizing the input stimuli.
2. The second step is to evaluate the activity profile of the design based on the chosen testcase. This step helps to narrow down the potential submodules and time windows for optimization.
3. The third step is to estimate the post optimization power metrics of the design at RTL level. This provides an initial forecast of the power saving potential in the design. Subsequently, the design is optimized by using observability CG insertion and stability CG insertion methods [11]. Thereafter, the functionality equivalence of the modified design is examined.
4. The fourth step is to synthesize the modified design and to extract the post optimization power metrics of the design at gate level(pre-layout, pre-CTS netlist). Comparing the metrics of the initial design with those of the modified design at gate level provides a precise the precise evaluation of the improvement on power savings, which in turn facilitates the front end design "signoff" decisions.

1.4 Thesis plan

In order to achieve this project in a systematic way, the work started with checking the Ericsson in-house materials and online reference materials to choose analytical methods and to sketch an initial flow. The author was trained in using the tools before fitting the tools into the designed flow. After setting the experimental environment for the flow, the approach was tested and evaluated on EMCA IP blocks to prove that this approach is generally applicable. The IP blocks that are chosen for experiments were discussed and selected before starting the case study.

In the meantime, testcases were discussed by the team to find the best testcase to optimize each block.

1.5 Thesis outline

Chapter 2 outlines the previous research on low power design. This, firstly, includes a explanation of the fundamental ideas, such as power terminologies and power metrics. Secondly, it includes a description to the analytical framework, optimization techniques. Thirdly, it includes an introduction of different tools that are used. Chapter 3 provides the framework of the design, from defining the testcases according to the analytical theories, to presenting the optimization methodologies. In chapter 4, there are provided some case studies based on the theoretical flow developed in Chapter 3. In chapter 5 the conclusions are presented and suggestions for needed future research is presented as well.

Chapter 2

Low power ASIC design background

This chapter provides an introduction to power dissipation and to elementary theories for power saving at RTL level. We start from organising the basic ideas behind this flow. At first, a brief introduction of the common RTL power saving techniques is provided. As we know, power dissipation is divided into dynamic dissipation and static dissipation. Therefore, the methods for saving power are explained separately by reference to dynamic dissipation and static dissipation. Then, power metrics are described aiming at evaluating the quality of power savings of a design. The tools used to analyze these metrics are brought up in the following sections. In the last section of this chapter, the optimization tool is introduced and explored. The quality of the optimization tool is examined in chapter 4.

2.1 Power vs Energy

The ultimate goal of saving power is to save energy. The difference between power and energy is very distinct. Energy is related to the operating power and the runtime of a design. In order to achieve low energy consumption, the design should be working in a short time frame with low power output. In other words, low power output does not necessarily mean low energy requirement. For example, the energy efficiency of GPU applications gets improved by adjusting the operating mode for the current workload. In figure 2.1, GPU works periodically and has constant idle times if they complete a task faster than the cycle periode [2]. The high frequency operating mode of the GPU releases energy contributing to a high temperature of the device. Decreasing the GPU clock frequency ideally saves power. In the reality, this is probably not true because of the redundant toggles during operations. This may cause more energy consumption when running GPU for a longer period. Therefore, both power and energy should be considered in

this thesis.

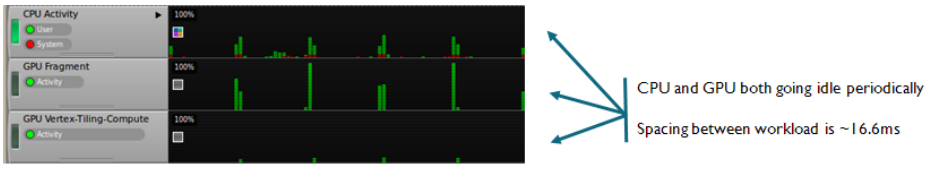


Figure 2.1: Screenshot from Streamline of a GPU and CPU idling each frame when the Dynamic Voltage and Frequency Scaling frequency selected is too high [2]

From a performance perspective, power dissipation can be classified into functional power and standby power [12]. Standby power dissipates when the design is operating in the idling mode. The functional power dissipates when the design is performing tasks. Power redundancies exist in both of the dissipation modes. The usage-specific testcases should be charaterized by the reference to those two dissipation modes to discover all the redundancy types in a design. The characterized testcases are, for example, idling the powered-on design to analyze the idle redundancy, and running in the worst operating condition to uncover the violating power bugs. This will be discussed in detail in the next chapter.

Figure 2.2 illustrates a conceptual power consumption waveform for a design. The curve plots the instantaneous power over time. The area under the curve illustrates the accumulated energy. In this figure, the curve initially exhibits a drastic spike. This spike is normally caused by initialization of modules in the design. After that, the curve is in a period of stable state which usually matches with idle state. The small spike during the idling time could be caused by glitches. The next part of the curve shows the functioning period, in which peak power occasionally occurs. Instantaneous power cannot reveal the complete picture of the energy expenditure while the average power does not display the instantaneous situation. If there are a lot peaking situations happening, the life span and the performance of the design will be affected. The occurrence of peak power is likely a result of some violating situations with error operation or bad components. Peak power also occurs in the devices with pulsing behavior, such as wireless sensors that periodically transmit information across a long distance. This power should be reduced by the designer to uncover the functional bugs from the RTL side, and handled in each specific case. In general, there are many measures to reduce peak power, such as, introducing large capacitors near the load circuit, careful selection of application components, and slowly ramping up the load [13]. This problem is normally solved in well-designed IP blocks. In this thesis, our interest is limited to shortening the average power for general cases and limiting the energy in total.

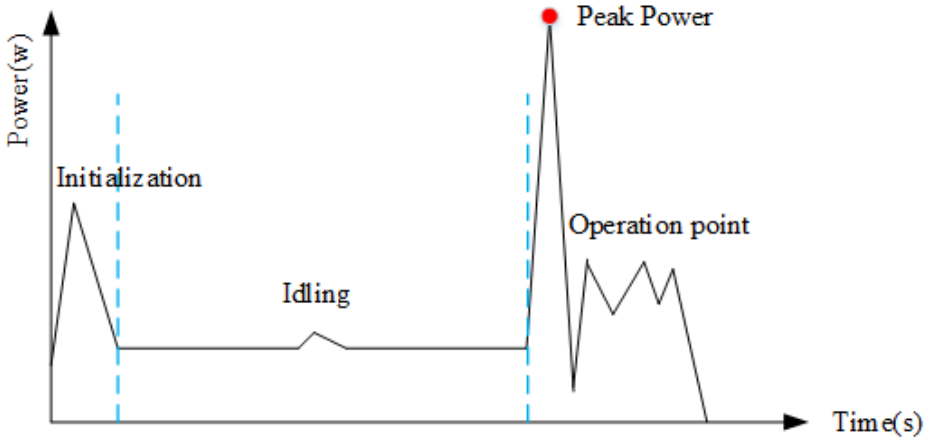


Figure 2.2: Power vs Energy plot of a running design

2.2 Net switching power

As methioned above, power consumption was divided into dynamic power consumption and static power consumption. As an example, a pull-up and pull-down NOT-AND gate for the input of a combinational/sequential logic gate is shown in figure 2.3. Two arrows in the figure illustrate the charging and discharging behaviours of capacitance C. The Voltage waveform shows the voltage variation corresponding to these two behaviours. The curve follows the exponential upward/downward trend during charging and discharging. The energy provided by the voltage supply is $C * V_{\mathbf{DD}}^2$. It is apparent in this figure that half of the energy is dissipated in charging and half is dissipated in discharging.

Assuming clock frequency is $f_{\mathbf{clk}}$, the switching power can be defined as the power dissipated in upward or downward transition. The switching rate is normally smaller than $f_{\mathbf{clk}}$. We use a scaling factor a to denote the switching frequency that is $a * f_{\mathbf{clk}}$. The scaling factor a is a factor number between 0 and 1 [9]. Based on these definitions, the net switching power can be given by eqaution (2.1).

$$P_{\mathbf{sw}} = C * V_{\mathbf{DD}}^2 * a * f_{\mathbf{clk}} \tag{2.1}$$

The tools used in the thesis, PowerPro and PrimeTimePX, estimate dynamic power by calculating the switching power of the net and the internal power of all the cells instantiated in the netlist. The net switching power depends on the wire parasitics that are usually provided by Standard Parasitic Exchange Format (SPEF) file, pin capacitances that are provided by library files, and the activity data of the net that is provided by the simulation data.

Except from the net switching power introduced above, imperfect switching also

causes other types of dynamic power inside of cells. For example, the switching power dissipates in the parasitic capacitances inside of a cell, and the short current power appears at the non-ideal gate transitions. There is a short period when a current path is active from the pull up structure to the pull down structure. This instantaneous access will lead to peak power that is called internal short current power. There is another non-ideal power release that occurs when there are no gate transitions, i.e. the pull up and pull down structures are off. There are still small capacitances connected between the power rails. The power wasted on this current path is related to as leakage power.

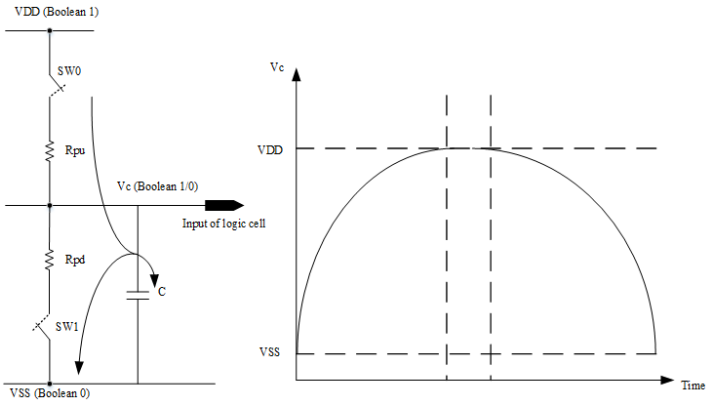


Figure 2.3: Switching power in a capacitor

2.3 Cell internal power

Cell internal power is defined as the dynamic power dissipated within the boundary of a cell [14]. There are two contributors to this dynamic power, i.e. internal short current power and internal switching power. The internal short current is, as mentioned above, the result of a short current current through the short circuit path during transitions. The short current is related to the technology node scaling, the cell type, the fanout of the gate, and the transition time of the input voltage [15]. The internal switching power is due to the charging and discharging of the capacitances within the bundary of the cell. The power estimation tools calculate internal power based on cell switching activity, cell type, output load and slew rate. Those information are provided in the library files and the simulation data. The best way to shorten the short current power is to increase the fanout and decrease the input transition time properly. To shorten the internal power, high threshold voltage is often used in the high power supply domain. But this method would lead to significant leakage power. Therefore, it should be used together with other semiconductor processing techniques. Nowadays, this short current can be reduced by technology node scaling and by assuring bigger fanout

within an applicable scope. Hence, the internal switching power usually is the dominant source of Cell internal power. This power is calculated and discussed during the dynamic power analysis of the case studies in chapter 4.

2.4 Leakage power

Leakage power is dissipated on the low current path when no switching happens on nets. For high performance devices, there are many categories of leakage power in CMOS circuits, most of them are really small and can hence be ignored when considering power consumption. The main leakage currents that lead to power loss are subthreshold leakage, gate-oxide tunnelling leakage and reverse biased junction leakage. This power is mainly consumed in the small capacitances in transistors. Therefore, the technological parameters of the cells determine the reduction quality on leakage power. Nowadays, with the extensive use of the FinFET technology, the leakage power is excellently optimized and became a non-critical concern for low power designs in recent years. In contrast to planar Metal-Oxide Semiconductor Field-Effect Transistors (MOSFETs), FinFET builds three dimensional channel between drain and source. The gate electrode is wrapped around the channel to form several gate electrodes on each side, which displays superior short-channel behavior [16]. This then reduces leakage effects and enhances the current density in FinFET devices comparing to the conventional MOSFET devices [17].

2.5 Power optimization methodologies

In this section, the purpose of saving power at the architectural level is discussed briefly. Also, a few techniques to reduce both static power and dynamic power reductions are discussed in short. Finally, clock gating, which is the main technique involved in the design, will be explained in detail.

There are multiple methods proposed to save power in ASIC design. The techniques of low power are mainly based on adjusting the power related parameters, as introduced in the previous sections. For example, static power is related to the cells and macros used in design and the supply voltage, whereas dynamic power is a function of switching frequency, net capacitance and supply voltages. There are many useful techniques to reduce leakage power [18], such as power gating that temporarily shuts down idling sub-blocks in a design, body bias control that dynamically adjust the threshold voltage to efficiently trade off leakage power and performance, and multi-threshold voltage technique that uses both high threshold cells and low threshold cells in a design to keep the performance while reducing the lekage power. The techniques for dynamic power reduction are usually the means to adjust one parameter or the combination of parameters in the switching

power equation (2.1), for example, bus encoding that encodes a piece of data to reduce activities on a bus, operand isolation that selectively isolates the modules with redundant opearations, and CG that blocks the clock signal when the circuit is not in use. A previous work presented a case study, in which he drew the conclusion that CG has the biggest efficiency by saving around 25% of power dissipation out of 30% total savings, comparing to the 5% of power savings by other techniques [19]. Also, he pointed out that the changes made in the early stage of design achieves better optimization effects. The attainable power saving difference between changes in transistor level and RTL level could be up to 20%.

2.5.1 Clock gating

From a design perspective, CG can be divided into hierarchical CG or architectural CG and local or auto (inferred) CG. Architectural CG is also referred to as coarse-grain CG. Such gates are manually added by the designer typically at the top level of large blocks or subsystems and are often software-controlled. Therefore, this technique requires the full understanding of the design at a system scope. The enable signals used in architectural CG are usually controlled by software or system wide conditions. This method does not violate the timing condition, and has the highest efficiency among the low power techniques. The fine-grain CG, referred to automatic CG, is usually done by the implementation tool [12].

There are usually two ways to do optimization using CG. The first type offers the full ability for gating un-gated cells. Considering the Verilog code below,

```
Always @(posedge clk) begin
        If (enable)
           Q <= D;
End
```

The output port of the register only switches if the enable signal is true. When the enable signal is false the output is fed back to the input of the flop. Also, the clk signal toggles on every clock cycle. This means that the register captures values even when the input is not valid. This type of design introduces a lot of toggles when the logic is not in use. In order to solve this problem, a clock gate is inserted on the clk pin as figure 2.4 shows. This insertion avoids unwanted toggles on the clk pin. The power consumption generated by the redundant toggles can be avoided as well.

The second type offers partially CG for imperfectly gated cell. The Verilog code example is shown as follows,

```
Always @(posedge clk) begin
     If (En_a)
        Q_1 <= D;
```

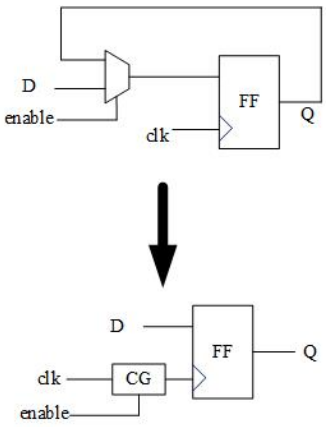


Figure 2.4: CG opportunity for ungated cells

```
        If  (En_b)
            Q <= Q_1;
End
```

Figure 2.5 illustrates the CG of the register in this example. The first flop has redundancies when En_a is true and En_b is false, the second flop has redundancies when En_b is true and En_a is false. With intimate knowledge of the design, the ability of the enable signals to gate these two registers can be strengthened by the combination of these two signals shown in the lower part of figure 2.5.

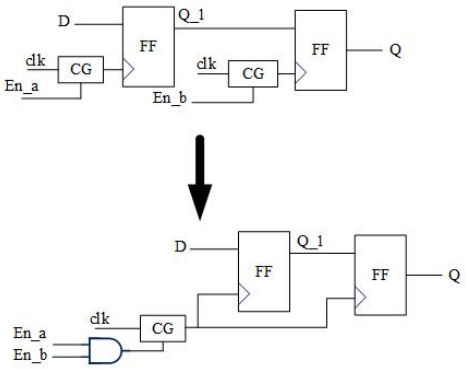


Figure 2.5: CG opportunity for imperfectly gated cells

2.5.2 Activity-driven clock gating

To discover the opportunities of CG in the design, the prerequisites of activity-driven CG are introduced. Activity-driven clock gating is done by using the

Switching Activity (SA) annotated files. The activities of pins and ports can be obtained from those files, such as Fast Signal Data Base (FSDB), Value Change Dump (VCD) and Switching Activity Interchange format (SAIF) files. The opportunities of CG are usually recognised by the implementation tool. At present, most of the tools are good at revealing Combinational Clock Gating (CCG) opportunities, while the Sequential Clock Gating (SCG) conditions are hard to discover. PowerPro can be used to compensate for this flaw by introducing the Observability Based Clock Gating (OBCG) and the Stability Based Clock Gating (SBCG) methods.

2.5.3 Combinational clock gating

The fine-grain clock gating handled by automatic power optimization tools is usually divided into CCG and SCG. Most of the power aware synthesis tools can operate CCG on designs, which discover the data held by registers and shut off the clock ports of those registers when there are no toggles on the data ports of those registers.

2.5.4 Sequential clock gating

SCG is a means to find new CG enable conditions through multiple clock cycles. SCG normally offers higher power saving potential than CCG. An example is given to better describe this approach in figure 2.6. There are two data paths in the design. A multiplexer is set to choose data from one path and transfer it through a FF to the output port. The data from each input path may be updated and transferred in many clock cycles while only one of them is in use. Then there occurs toggles on the unused path that cost redundant dynamic power. Under these circumstances, the select signal of the multiplexer can be used as the CG condition to gate the unused path, as illustrated in the lower part of figure 2.6. This is a simple example, normally the CG enable condition is very intricate and difficult to recognise through multiple clock cycles. Therefore, SCG is more complicated than CCG to implement and mainly operated manually by the designer.

2.5.5 Observability-based clock gating

The sequential clock gating dispose of two types of gating opportunities. The first method is to find the redundant writes that are unobservable downstream, which is called observability-based clock gating. This unobservable opportunity is presented in the example in the last section. Unobservable opportunity means that the writes to the input of a data path can not be observed at the output of the design [20].

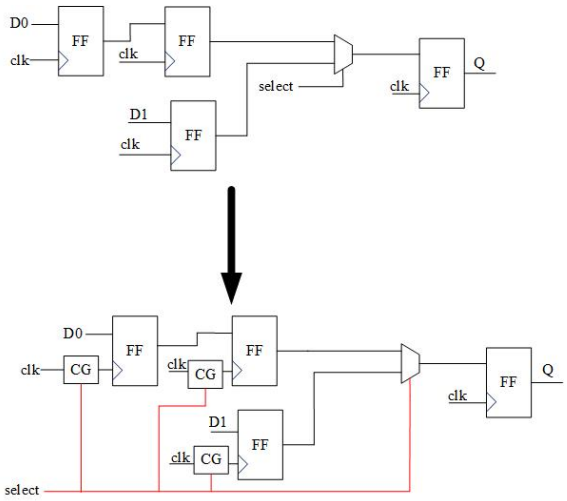


Figure 2.6: Sequential clock gating example

2.5.6 Stability-based clock gating

The second method is to discover and eliminate the inefficiency caused by the same data written consecutively to the input of a data path. This method is called stability-based clock gating [21]. SBCG can be used to solve the power vs energy issue described in section 2.1. The CG methods mentioned above are later used in the power optimization flow in this thesis.

2.6 Important power metrics

In this section, the definitions of important power metrics are provided. Those metrics are used as measurement criteria to evaluate the power saving potential of a design. Some metrics can also be used as conditions to customize the optimization procedure in order to achieve goals, such as high power efficiency and low cost, in a design.

2.6.1 Average switching activity

SA is a metric to directly measure dynamic power. SA can be defined as the toggle count per clock period. The switching activities of pins/ports within a module can be averaged to represent the general behavior of a module. This metric provides a good way to select and optimize logics at the module level.

By running ActivityExplorer[22], we can visualize this Average Switching Activity (ASA) in the hierarchical map. The activity profile can also be visualized in a runtime window(in figure 2.7). This metric helps to limit the potential region of CGs in a design, and to select the time interval in the full runtime window for efficient CG.

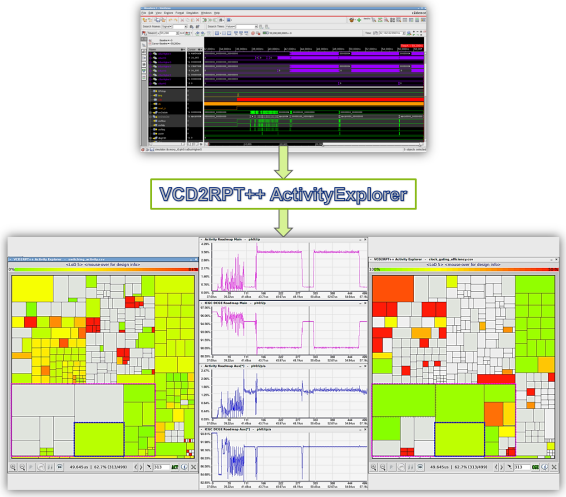


Figure 2.7: Switching activity visualization [3]

2.6.2 Static clock gating efficiency

Static Clock Gating Efficiency (SCGE) is an evaluative metric that is usually used together with other metrics to estimate the sufficiency of clock gating in a design. SCGE is defined as the ratio of the gated register count to the total register count in a design scope. SCGE is normally considered to be a structual metric for dynamic power. SCGE directlly illustrate the ratio of gated register in a design. The fact that a module has high SCGE does not necessarily means that it has low dynamic power consumption. For example, there are some logics that are always active in the design. Gating those logics may not reduce dynamic power, while the inserted gating cells would introduce more power consumption. But if SCGE is quite low, the design is usually insufficiently gated. SCGE can also be used as a complementary criterion to evaluate the quality of the optimization process together with other metrics to complete a credible assessment.

2.6.3 Dynamic clock gating efficiency

Dynamic Clock Gating Efficiency (DCGE) is one of the key metrics to identify inefficiently gated logics. The definition of DCGE is the percentage of time when the clock port is gated. DCGE is a precise indicator of gating condition in a design. This metric shows how effectively the redundant clock has been suppressed by inseted clock gates. However, a low DCGE does not necessarily means bad gating condition. The same example in SCGE subsection can explain this. There may be some blocks that are always active during the operation. Hence the clock that is fed into those blocks cannot be suppressed even with good CG condition. DCGE can be formalized by the equation (2.2). In this equation, Gated Clock Toggles (GCT) denotes the number of gated clock toggles, Total Clock Toggles (TCT) denotes the number of total toggles on the clock pin.

$$DCGE = \frac{GCT}{TCT} \tag{2.2}$$

In the example in figure 2.8, DCGE of the FF is 2/6 according to equation (2.2). The average DCGE in a module is the mean value of total DCGE of all registers in the module. Average DCGE in a design can be calculated in the same way. Average DCGE is primarily used to evaluate the power saving quality of a design.

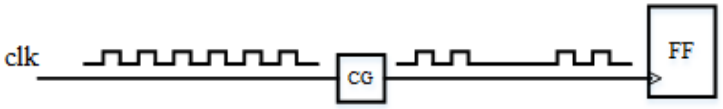


Figure 2.8: A gated FF

2.6.4 Ram read/write ports utilization

Memory Read Port Utilization (MRPU) and Memory Write Port Utilizations (MWPUs) are determinate metrics in memory gating [23]. The generic function to compute MRPU/MWPU is given as equation (2.3).

$$MRPU(MWPU) = \frac{RUC(WUC) * 100}{CC * MC} \tag{2.3}$$

In this equation, Read Utilization Counts (RUCs)/Write Utilization Count (WUC) stands for read/write utilization count that means the enabled time of memories in a clock cycle. Clock Cycle (CC) denotes clock cycles. Memory Counts (MCs) denotes the same type memory count in a design.

The algorithm of read/write utilization count is determined by different memory types. There are usually four types of standard memory used in ASIC designs,

which are 1rw ram, 1r1w ram, 2rw ram, 2r2w ram. In 1rw ram write operation and read operation share one port. 1r1w ram means the memory has one read port and one write port. In 2rw ram the memory has two partitions for read and write operations separately. There are two ports shared by read and write operations. 2r2w ram means that the memory has two partitions, and there are two read ports and two write ports. The utilization count of read operations for the memory with multiple ports can be defined as the mean of the total utilization count on read ports. The same rule applies for write utilization count. For the memory with shared read/write port, the read utilization count is usually accumulated when the shared enable signal is low. The write utilization count is accumulated when the shared port is high. The calculation for this type of memory normally requires the assistance from the Chip Select (CS) signal. For the memory with chip select pin (as the example in figure 2.9), the port utilization count can be calculated based on the Read/Write signal (Read/Write enable signal) and the CS signal. The pseudo-code below describes the process to calculate read port utilization. CS is active high and Read is active low. The memory performs reading(RUC increments by 1) when both signals are active. In the example presented in figure 2.9, both of the read utilization count and write utilization count equal to 1.

```
If (posedge clk) begin
    If (Read='0' and CS='1')  begin
        RCU++;
    end
end
```

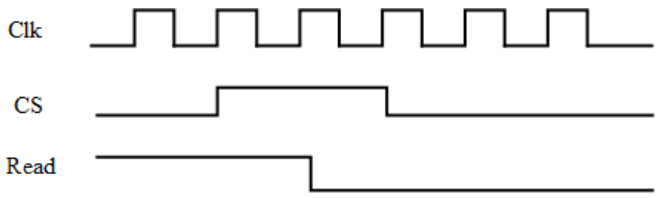


Figure 2.9: Memory signals with CS pin

For the memory without CS pin, the calculation needs to use the combination of read/write enable port, and data input port or Address ports. There is an example in figure 2.10. The read utilization count can be recorded as indicated in the pseudo codes. Read_old denotes the Read enable signal in the last CC, Address_old also denotes the Address signals in the last CC. The valid read operation occurs when address ports toggle and Read is low. Hence, read utilization count can increments by 1. In the example in figure 2.10, read utilization count number is 2 and write utilization count number is 1.

```
If ( posedge clk) begin
    If ( Read='0' ) begin
        If (Read!=Read_old or Address!=Address_old) begin
            RUC++;
        end
```

```
        end
    end
```

Memory port utilization can either be reported on a memory instance or on the same type of memory. To get the average memory read/write port utilization on an appointed type of memory, the mean of the total read/write port utilization of the memories instantiated by this appointed module should be calculated. This metric can be used as a criterion to choose inefficient of memories for memory gating. The memory with high MRPU/MWPU can be defined as high dissipative memory in the scenario.

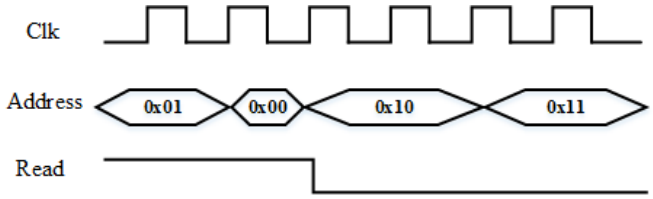


Figure 2.10: Memory signals without CS pin

2.7 Analytical tools

The automatic power optimization flow proposed in this thesis primarily involves a pre-optimization analysis stage, an optimization stage, and a post-optimization analysis stage. The tools used in this flow are mainly the analytical tools. The analytical tools are used to report power metrics of design and to track the power variations through the whole flow. The optimization tool is used to provide solutions for CG and automatically modify the design at RTL level.

2.7.1 ActivityExplorer

Ericsson provides an in-house tool chain for sanity check and power metric analysis of design [24]. This tool chain has three main usages that are inserted in different stages in the flow. These tools are VCD2RPT++ for SA analysis, VCD2TB for simulation replay, and ActivityExplorer for SA and Clock Gating Efficiency (CGE) visualization. From the early stage in the flow, this tool provides the average switching activity report and the average clock gating efficiency report. These reports can be used to identify inefficient modules in specific testcases. This toolchain also reports the memory utilization as a reference metric to make decisions on memory gating. Additionally, this tool provides a replay of RTL simulation for netlist design. This helps to analyze pre- and post- optimization

designs at netlist level, and to dump out a precise power metrics report for clock optimization quality in the end of the flow.

Figure 2.11 depicts the utility skeleton of a toolchain. The toolchain can handle analysis at RTL level as well as at gate level. A memory list including all the memories in the scope of the design hierarchy can be bound with an in-house memory activity monitor to observe the behaviour of these memories and track their activities during simulation. A reference signal toggle report can be generated in simulation for sanity check. The memory utilization report generated by the memory activity monitor can also be used to check sanity as the second assurance. This in-house solution can dump memory ports utilization reports for a memory instance or a memory type in a design. The testcases are simulated in the UVM based verification environment with respect to different usage-specified testcases. A VCD or FSDB file can be dumped by running such a simulation, which will later be fed into other tools as the activity annotation file in the testcase scenario. The VCD files dumped from RTL simulation are used as the source for VCD2RPT++ for RTL level evaluation, by specifying the sub-hierarchy. The activity report of the sub-modules in the target design can be acquired. Also, ActivityExplorer provides an activity tree map to visualize the activity profile of each sub-module. A screenshot sample of this explorer is shown in figure 2.12.

The left part of figure 2.12 shows the activity tree map at a hierarchical level in a design. This explorer also provides the profile of activity vs time for selected instances (right part of figure 2.12). During RTL analysis, this tool helps to visually scale down the region with suspiciously low CG efficiency and to identify time windows with high activity.

VCD2TB is a tool that is used to transform VCD files, that are dumped from RTL simulation, into power replay testbenches that are used for gate level simulation. This tool helps to replay the RTL simulation in the Netlist level for gate level analysis. The memory activity monitor can also be used at the gate level. The memory activity report and the reference signals dumped in gate level can be compared with those dumped in RTL level to validate the correctness of gate level simulation.

During netlist analysis, the VCD file dumped from netlist simulation is used as the source for VCD2RPT++ to generate an average switching activity report and an average clock gating efficiency report. The explorer not only visualize the activity profile in the design hierarchy, but also display the CGE profile of the design hierarchy in GUI. By carefully analyzing the SAs and CGE at simultaneous time points, the inefficient gated logics can be pointed out. Then, the profiles of SA and CGE can be provided, which makes it possible to identify the proper time interval for design optimization.

The Ericsson in-house toolset described above provides an effective and efficient environment for the detection of power saving potential in the early stage that

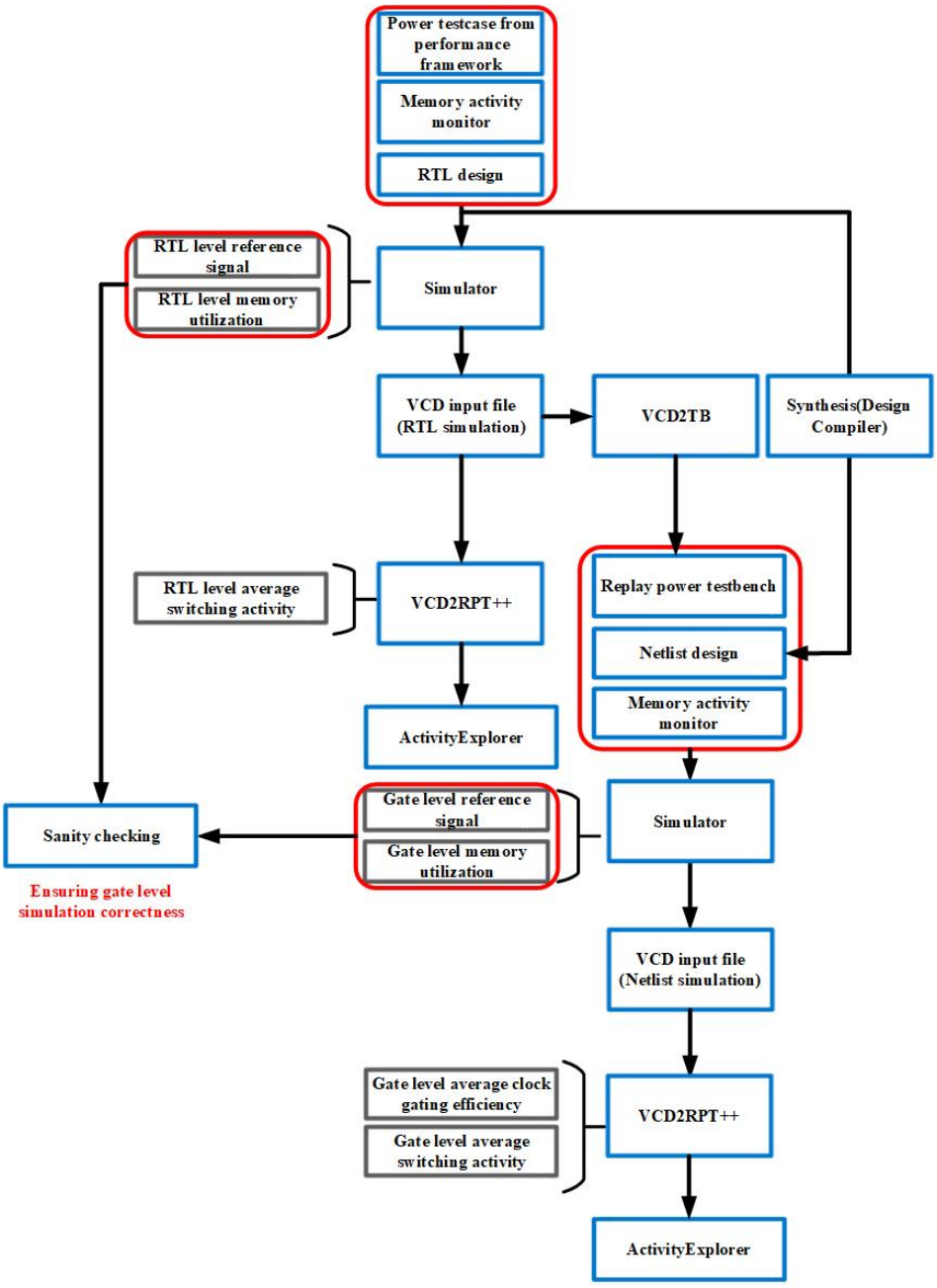


Figure 2.11: The procedure skeleton of Activity analysis flow

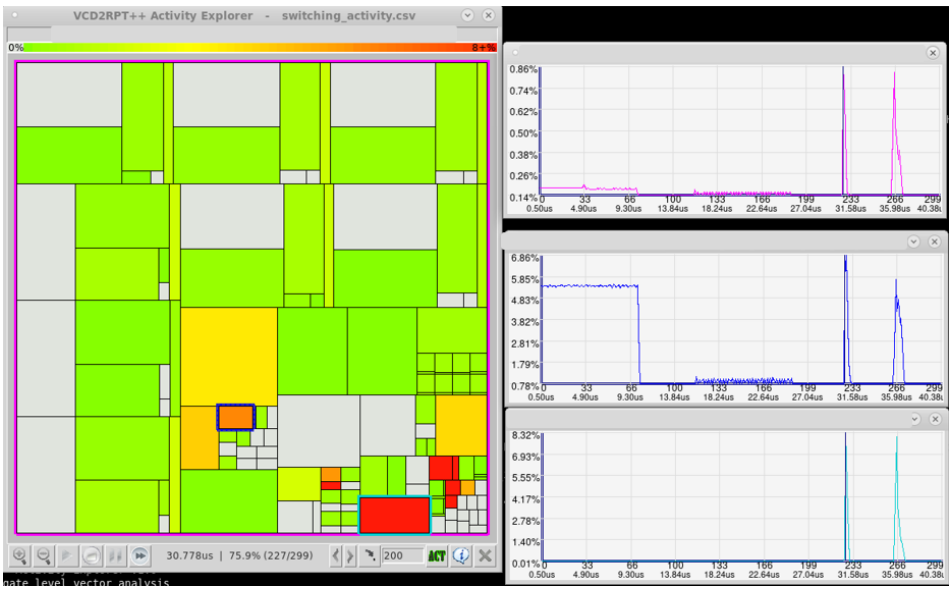


Figure 2.12: A case analyzed in the ActivityExplorer

reduces the efforts and runtime for later optimization in the flow. Beyond that, this tool can dump multiple power metrics for the evaluation of the quality of power optimization.

2.8 PrimeTimePX

PrimeTimePX can provide precise reports for power metrics, such as DCGE and hierarchical SA, and more importantly can provide an estimation of the power numbers for pre-layout and post-layout design [14]. Therefore, PrimeTimePX is utilized as the signoff tool in the IP design optimization flow.

The scope of this thesis is in the pre-CTS and pre-layout netlist for the optimized blocks. The analysis run by PrimeTimePX usually involves five steps (outlines in Figure 2.13). The first step involves building the background setup, for example specifying the analysis mode that can either be set averaged or time-based, and defining the technology libraries that provide the standard cells and memories used in a design. For the analysis of pre-CTS netlist, the clock tree buffer should be specified in this step. The second step is to read the netlist file and link the design with the modules from the technology libraries. After this step, SPEFs file should be read to annotate the parasitic resistance and capacitance. Synopsys Design Constraints (SDC) file is read to define the constraints on Input Outout (IO) path, pins and ports, and the timing constraint such as virtual clock, clock

transition time and clock delay. The third step is to load the SA file, for example a VCD file, to annotate the switching activities on nets and pins. This step annotates the specific testcase scenario for the power analysis on the design. These four steps can be combined as a preparatory procedure in the PrimeTimePX power analysis flow. The final step is to compute and report the power metric numbers and the power estimation numbers.

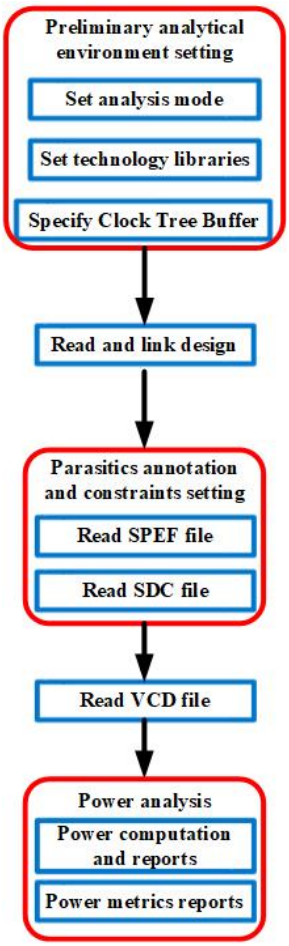


Figure 2.13: Power analysis flow using PrimeTimePX

PrimeTimePX can generate net based and cell based power reports, it can also summarize the power numbers by report power on seven power groups (memory power, io pad power, register power, combinational power, sequential power, black box power, and estimated clock power). The net switching power, leakage power and cell internal power introduced in the previous sections can also be reported by this tool.

In this thesis, PrimeTimePX is applied to the pre-layout and pre-CTS netlists, and is used in the end of the IP optimization flow as a guarantee of power saving performance.

2.9 Optimization tool

PowerPro is an efficient EDA power analysis and optimization platform, which integrates multiple power analysis and optimization schemes. PowerPro can either be set in the early IP design stage assisting designers to manually optimize the performance in early design level, or be set in the late design stage to automatically discover the CG opportunities. A procedure chart of PowerPro inserted IP front end design flow is illustrated in Figure 2.14. In the early phase of front end design flow for an IP, the unfinished functions have a big potential in power improvement.

PowerPro can be used as a guidance for the gating direction, by loading in the initial design. A vector-less analysis and rough estimation can be performed [25]. The metrics obtained from this early estimation gives an initial impression of the power saving quality. By running PowerPro in guided mode, observability based information and stability based information are listed revealing the ungated logics. The tool identifies two types of redundant writes. Based on the identification, the tool ranks the registers according to the potential savings on power by gating them. The two types of registers are stable register with low CGE and high-active register with a high percentage of unobservable writes. Based on the hints given by PowerPro, the designer can effectively carry out optimizations on inefficient logics.

In the case that a RTL design is released, the function of the design should not be modified. PowerPro supports running automatic power optimization mode for a ready-to-synthesis design. The automatic power reduction methodologies utilized by PowerPro has been discussed. It generally performs six steps of optimizations on design, which are shown in Figure 2.15. The designer can use the analytical power metrics to make an effective plan including the measures to take on the specific design. For example, the memory ports utilization rate helps to find the inefficient memories to be optimized. The memory gating usually spends a long period of runtime compared to other optimization means. It is rather inefficient to perform memory gating on all the memories in the design. Memory gating was not in our scope because the memories in the blocks were already efficiently gated. The different options for customizing an optimization flow are shown in Figure 2.15. A description of these measures is given below [26].

- Observability based clock gating: Identify the unobservable writes in a data path and find the new enables to gate the related registers on this path.

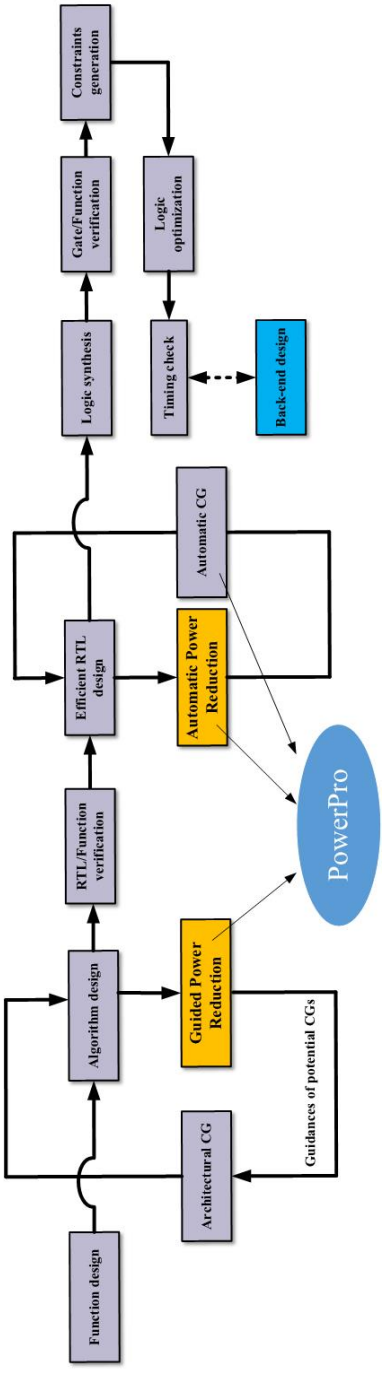


Figure 2.14: PowerPro inserted Front-end IP design flow

- Symbolic stability based clock gating: Identify the stable or unchanged writes for a period at the input of the register. The enable conditions can be generated at the head of the pipelined data path to gate the related registers.
- Constant stability based clock gating: Discover the constant or unchanged writes in the flops and generate new enables to gate these flops.
- Observability based memory gating: Identify the read out data from memory, which are not used on the downstream path. The enable signal should be generated to disable the memory during this redundant read period.
- Symbolic stability based memory gating: Identify the stable read out data from the same address without intervening writes, and gate off this repetitively readings of memory.
- Constant stability based memory gating: Discover the constant read out data from the same address in a memory, and gate the enable port of this memory.

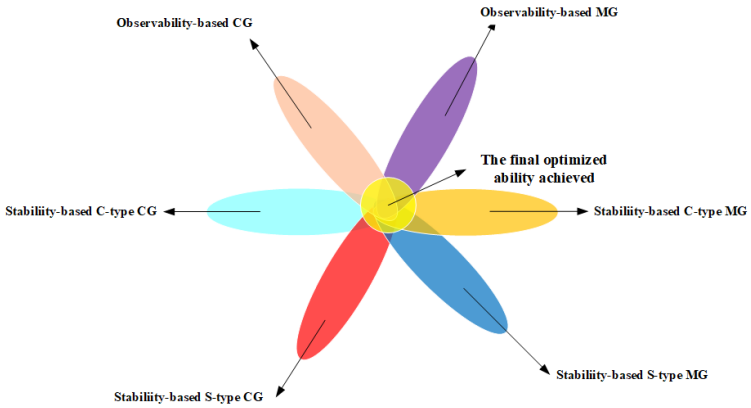


Figure 2.15: Power optimization measures provided by PowerPro

For a better estimation of power saving opportunities, PowerPro requires several resource files to setup the flow, such as initial RTL files, technology files, a SDC file, a SA annotation file, and a SPEFs file. Only the clock related commands in the sdc file are needed for optimization and analysis purposes. The other commands, that could lead to unmatching power estimation to PrimTimePX analysis results, such as load setting commands, should be commented. The setup of PowerPro normally involves four steps. The technology library list needs to be set and read in. The operation mode of PowerPro should be specified to guided or automatic mode. The gray box is defined if there are any modules that should not be modified. Black box should be specified if there are unsynthesizable modules or if there are missing definitions of modules existing in the RTL files. Beyond that, extra settings can be specified for specific purposes. The global variables are customized to meet the requirements. For example, the minimum bitwidth

for clockgate insertion can be set in PowerPro to assure efficient CG. The specific variable setting will be described in the next chapter. PowerPro imitate the behavior of the synthesis process to estimate accurate power numbers. The RTL files should be read into separate design libraries and be elaborated, and a design database should be built from the linked design. Powerpro provides a synthesis alike operation called prototype. The SPEFs file, the FSDB file, and the SDC file are read in before this step to build physical characterization. The prototype mainly performs three sub-operations [11]:

1. Consistency checks: checks the consistency of read-in libraries and constraints, and ensure that they are valid and accepted by the tool.
2. Normalization: transforms the design in accordance with the database norm.
3. Generic optimizations: performs logic optimizations in order to remove redundancies. This is similar to the logic optimization in synthesis.

After prototyping, the design is ready to be optimized. If stability based optimization is to be performed, a reset signal needs to be defined before all optimization operations. This provides the reset signal to the newly inserted gates in stability based optimization. As mentioned above, PowerPro provides six types of optimizations to improve the power quality in the design. If memory gating is to be performed, memory ports definitions should be specified before linking the design. Then PowerPro will include memory as part of power estimation. The memory gating can be done based on the power analysis on memories.

In the final step, PowerPro generates the power and metrics reports. Also, optimized RTLs, and a Sequential Logic Equivalence Checker (SLEC)-Pro Tool Command Language (TCL) script that checks the functionality equivalence of the optimized design, are written and dumped out automatically. As same as PrimeTimePX, PowerPro also provides power estimation in five power categories in accordance to the classification of power groups, i.e. IO pad, memory, register, black box, combinational, sequential and clock network power group. The power report presents leakage power, internal power and switching power. These numbers can be compared with the report from PrimeTimePX. The accuracy of power estimation by PowerPro will be discussed in the fourth chapter. All the metrics with power numbers can be reported before and after optimization to make an elaborate assessment of the optimization quality. Figure 2.16 presents a detailed IP optimization flow by PowerPro.

Furthermore, the files that are optimized by PowerPro should be verified using SLEC-Pro before getting into gate level analysis. SLEC-Pro is a functional verification tool at RTL level. A SLEC-Pro verification TCL script that binds a list of checking files for different optimization types is generated every time an optimization has been run by PowerPro. This TCL file can be fed into SLEC-Pro to verify if the optimized files have the equivalent logics to the original files. If the

results confirm this, then the modified RTL design can be deemed as functionally reliable. In this thesis, all the optimizations are conducted by PowerPro 10.4_5.

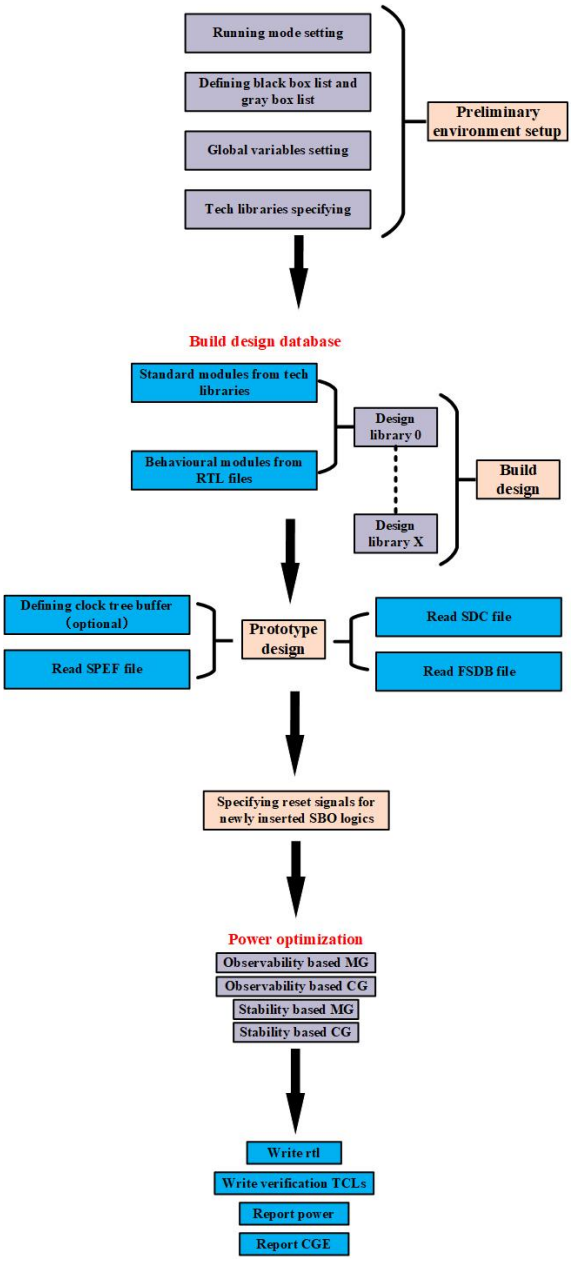


Figure 2.16: General PowerPro optimization flow

Chapter 3

Design of automatic power analysis and optimization flow

The idea of the automatic power optimization flow is to achieve a novel flow that features automatic power optimization along with integrated formal verification guarantees for bug-free RTL. The background knowledge of this flow and the fundamental tools have been described and commented in previous sections. This flow focuses on efficient automatic optimization in a way that is aligned with the estimation on power saving potential at the pre-optimization stage and that is aligned with the analysis on power saving quality at the post-optimization stage. In this section, the flow will be described in the order of a IP design flow.

3.1 Power saving potential estimation flow

Power saving potential estimation, as the initial step in flow, has a salient effect on power optimization quality. This step is normally usage-specific, but the methodology behind it can be generalized to fit all IPs or sub-modules in an ASIC design. In this step, Ericsson in-house tools are used to select effective testcase or a combination of testcases, and to narrow down the effective time windows for optimization within the determined operation scope. The details of testcase selection and time window identification are presented in the subsections.

3.1.1 Selection of testcases

In order to mimic the physical operation point of a design, usage-specific testcases are generated and simulated in an UVM based verification environment. Those

testcases not only indicate different power consumption, but also activate different parts of the target design. Therefore, different testcases can illustrate different potential for power saving. As mentioned in a previous work [9], block characterization is a very critical step before making decisions on the right testcase. The characterization classifies performance at reset mode, idle mode, low activity mode, typical mode, high activity mode, high DC/DC thermal mode and error mode. The characteristics of those modes have been recorded and organised in [9]. It can be summarized as follows,

1. A testcase at reset mode involves very low utilization. This mode contributes to a very low power consumption.
2. Idle mode also involves low utilization. Idle mode usually consumes somewhate slightly more power than reset mode. Idle mode and reset mode can both be characterized as very low activity mode.
3. Low activity mode involves scenarios such as initialization and configuration.
4. Typical mode represents the normal functions operated by the design, which shows a period of averaged and stable activities through the operation time window. The power consumption in this mode is in the medium range.
5. High activity mode is defined as the operation mode that consists of high traffic through the operation period. This mode normally consumes power above the medium level.
6. DC/DC mode is about sudden but realistic changes in activity from very low to very high point and vice versa. Thermal is about large time constant high activity scenarios. These are typically not interesting scenarios for power optimization.
7. Error mode is more complicate and needs specific analysis under different error situations. This mode could either lead to the worst power consumption scenario or to a scenario with no negative effect on power dissipation.

ASA is normally used to estimate the power performance of a testcase. The purpose of defining the characteristic of each mode is to discover the full potential of power saving in the corresponding scenario. For example, for operations in idle mode, most of the blocks are inactive. That leaves a big room for stability CG if unwanted activities occur in this scenario. When the block is functioning in typical mode, some unobservable redundancies can be discovered and gated. Especially for high traffic mode, the avoidable traffic in blocks should be minimized to save power.

Different functions of a design activate both usage-specific blocks and blocks for common use. Hence, different optimization solutions should be devised and committed for different cases in the typical mode. This can be illustrated by figure 3.1. The average utilization map for sub-blocks in an IP block sample are divided

into two different usage scenarios. From this figure, gray denotes inactive, green denotes low traffic, yellow means medium traffic, orange means slightly elevated traffic, red denotes high traffic. Figure 3.1 a) depicts the situation in a typical mode scenario. The computation unit and First-In, First-Outs (FIFOs) are occupied with high activities. More observability CG opportunities can be found in these two blocks, while more stability CG opportunities can be found in blocks like register controller and common memory. The timer block is probably lacking of CG opportunities because of its small size. In a reset mode situation depicted in figure 3.1 b), the computation is in an inactive state while common memory block is in high use because of initializations. This scenario might require a totally different testcase for optimization comparing to the previous scenario. One thing should be noticed; that register control, FIFOs and common memory blocks are common blocks that are used by both scenarios. Therefore, the optimization for one usage scenario can also reduce the power consumption in another usage scenario.

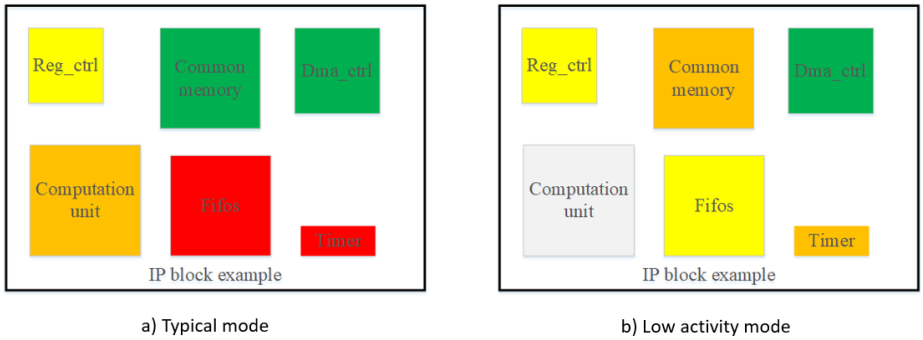


Figure 3.1: Utilization map of some operation examples

The optimization of a design should cover as many sub-blocks as possible, and it should also be generic and efficient for most usage scenarios. That is because the power optimization is usually a one-time decision before going to the back end process. To achieve the optimal solution, the testcases of a block should be fully understood. The testcase analysis should be expanded on the single testcase vs mix-testcases optimization scheme. This scheme requires some preliminary knowledge about the usage conditions for a block in the real world, i.e. the most common usage conditions should be prioritized. In reality, an IP block is usually instantiated many times in a design for different usages. Based on the actual situations, the usage-specific testcase should be selected and analyzed. The generally used testcases for an IP provided by the design team are listed in Table 3.1.

Those testcases are mostly designed in the verification environment, but they can also be utilized in the power optimization prone environment. Therefore, these testcases should first be charaterized with performance mode. The performance points of the the samples have been presented above. For example, an initialization

Table 3.1: General test for IPs in an UVM based simulation environment

Testcase	**comment**
Random test	This test drives constrained random accesses to random blocks such as registers and caches in design
Direct access test	Direct accesses to registers
Large test	This test drives constrained large size accesses to all the valid directions in a block.
Small test	This test drives constrained small size accesses to all the valid directions in a block.
Initialization test	This test does full initialization
Halt test	This test randomly inserts halts in a process
Reset test	This test runs reset for a block
High load test	This test drives high traffic load to a block
Illegal test	This test runs error signals to a block
Config test	This test builds configurations for a block

test can be related to a low power scenario, while a random test can be related to a medium power scenario. Based on the actual usage scenario, a proper testcase can be chosen to run the optimization. This approach can propagate down to sub-blocks for target block optimization, which is supported by the testcase of direct accesses to the target block.

If no special testcases are specified for optimization, the general analysis should be conducted on all the testcases. In addition, the combinations of testcases should also be considered to perform mix-testcase optimization. The weight factor of each testcase should be decided in the mix-testcase optimization considering the usage rate of each testcase in the real world. For example, if a block is at idle state for around 70% of the time in real life, the weight factor for the idle testcase should be set to 0.7 while the weight testcase for function testcase should be set to 0.3. However a weight factor is not a friendly option for reproducing the testcase on the netlist. That is needed for verifying the optimization result at netlist level. Instead tailoring a test to include multiple sub-tests in sequence can allow the tool to identify multiple opportunities and also be able to run at netlist level. A recommended testcase for optimization should have "valleys and hills" that covers all the possible performace points. Hence, a synthetic testcase should combine "reset", "idle", "low activity", "medium activity" and "high activity" with reasonable duration(ladder type). The conceptual figure of a synthetic testcase is shown in figure 3.2. The whole procedure flow of testcase selection is illustrated in figure 3.3. This testcase is recommended as a default option. If no specific testcases are provided and if one does not know how the design is going to be used, the synthetic testcase should be adopted as a baseline.

With all the available testcases, power estimation can be conducted and compared. The optimal testcase or a combination of testcases can be chosen to move on to the next stage.

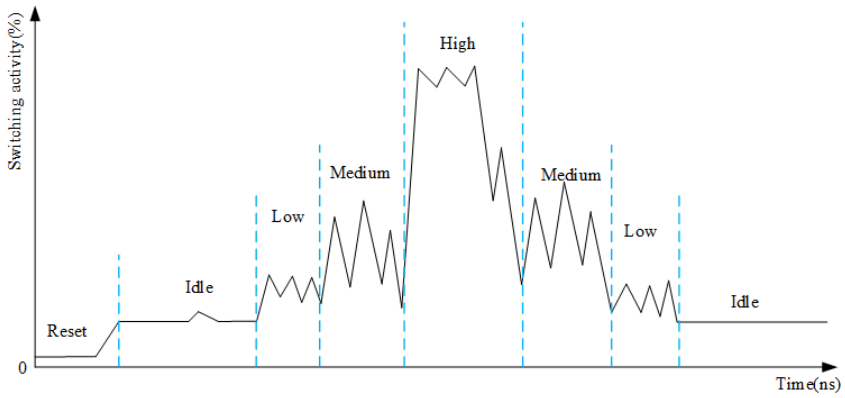


Figure 3.2: Synthetic testcase for optimization

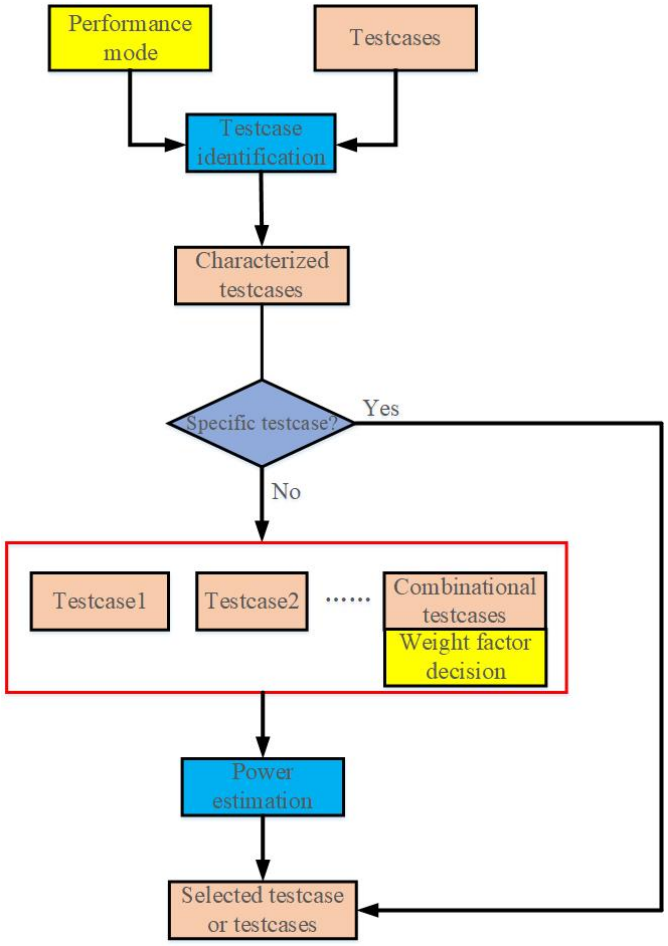


Figure 3.3: Testcase selection flow

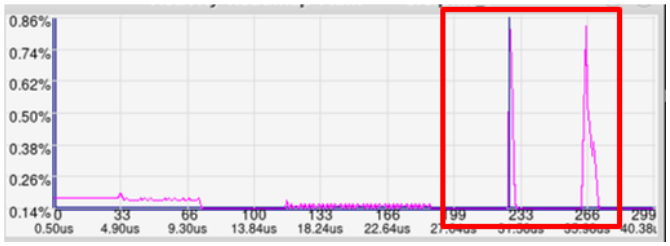


Figure 3.4: Activity profile for an IP with a highlighted time window that corresponds to optimization

3.1.2 Power estimation at RTL level

As proposed above, the available testcases are filtered by performance mode characterization and the candidate testcases are determined. The next step is to find the optimal testcase based on multiple metrics, namely DCGE, SCGE), dynamic power reduction rate and area change. Power estimation at RTL level can be done by PowerPro. Hence, the power estimation step is overlapping with the power optimization step. This section only focuses on the early estimation of power reduction by PowerPro. The optimization procedures will be explained in the next section.

Power estimation starts by locating the timing interval for power optimization. This step is performed using the toolset presented in section 2.7.1. The optimization time window should contain the main activities in the whole operation. An example is shown in figure 3.4. The highlighted time interval contains two spikes in the activity profile of an IP. These two spikes illustrate the power consumption of the analyzed IP based on the chosen testcase. This data can be fed in to PowerPro as the source of observability optimization. The idle interval between these two spikes helps to discover opportunities for stability optimization. A testcase constrained by a time window that includes a mix of idle states and activities is usually a good testcase for power optimization.

After the time interval is determined for each testcase, PowerPro environment should be set up to implement standalone optimization (single testcase) and weighted optimization (combination of testcases). The flow presented in figure 3.5 illustrates the measures and criteria for selecting the optimal testcase. When the optimization is finished, reports are generated for each candidate testcase on technology-independent metrics (area variation, DCGE and SCGE) and on dynamic power estimation by PowerPro. A necessary condition for area increase should be checked to make sure that no heavy effort is added on the back end side. If the optimized design has more than 2% area increase(emprirical data), the optimization should rerun with new area constraints. If all the conditions are fulfilled, the comparison of DCGE variation and SCGE) variation between eligible testcases should be made. This helps to select the optimal testcase with most DCGE increase and

most SCGE) increase. Then, the power saving potential of the optimal testcase can be derived from these analyses at RTL level.

This step is an initial selection of the optimal testcase, which should be verified on the gate level. In this thesis, multiple optional optimization testcases are chosen for gate level analysis to demonstrate this method. The sign-off report from PrimeTimePX can be used to make the final decision.

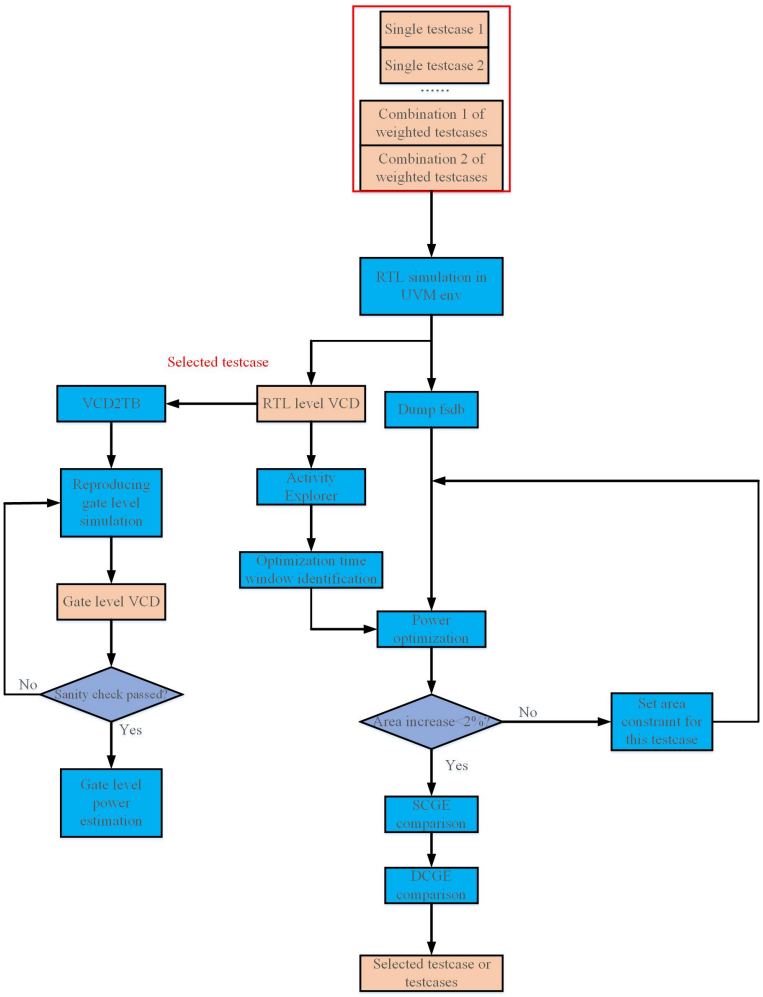


Figure 3.5: Power estimation flow for candidate testcases

3.2 Power optimization flow

In the previous Ericsson work[9], designers have to manually analyze and optimize the blocks to improve the combinational and sequential clock gating. However, PowerPro provides an automatic flow to identify the gating opportunities. Based on gating opportunity identification, PowerPro generates new enable conditions or finds the existed conditions to implement code modification on the original RTL files.

The methodology flow in figure 3.6 describes the concept of power optimization. The preliminary setting of PowerPro is explained in section 2.8. The global variables are normally set up to constrain the optimization. For example, the minimum bitwidth for clockgate insertion can be set to filter out ineffective CG moves. The running mode can be set to iterative mode to discover and clean up gating opportunities that could cause compilation issues. After setting up environment for PowerPro, the FSDB file of an usage-specific testcase is read in the tool. Multiple FSDB files can also be read in to run the mix-testcase optimization. In this flow, annotation rate is used to guarantee the reliability of FSDB file. The annotation rate in a run should be higher than 80%. If the annotation rate is lower than 80%, the design has too low assertion to optimize the design. There could be several reasons for this, such as that the signal path name from the FSDB file does not match the hierarchy of prototyped design in PowerPro, or that the FSDB file does not include any activities for multi-dimensional array. This step requires manually debugging to identify the problem and to determine the solution to the problem.

When PowerPro runs automatic CG opportunity identification and optimization, the committed moves are normally higher than 98%(empirical data). PowerPro decommits the moves that cannot be patched and ineffective moves. If more than 2% of moves are decommitted, debugging is needed in this step to anchor the problem for each decommitted move. Based on the guidances provided by PowerPro, manual patching or other means are required to solve the problems to achieve higher commitment rate. The threshold of 2% area increase(empirical data) should be maintained.

3.3 Post optimization functional verification

Post optimization functional verification integrates equivalence checking and quality analysis from both RTL level and gate level. This procedure is a sign-off step to validate the correctness of the optimized design and the superiority of selected testcase. A direct check provided by PowerPro compares the original RTL model to the optimized RTL model using sequential analysis technology. This verification does not depend on any testbench. It can be seen as a static functionality check. On top of this, provides a sanity check approach based on the simulations

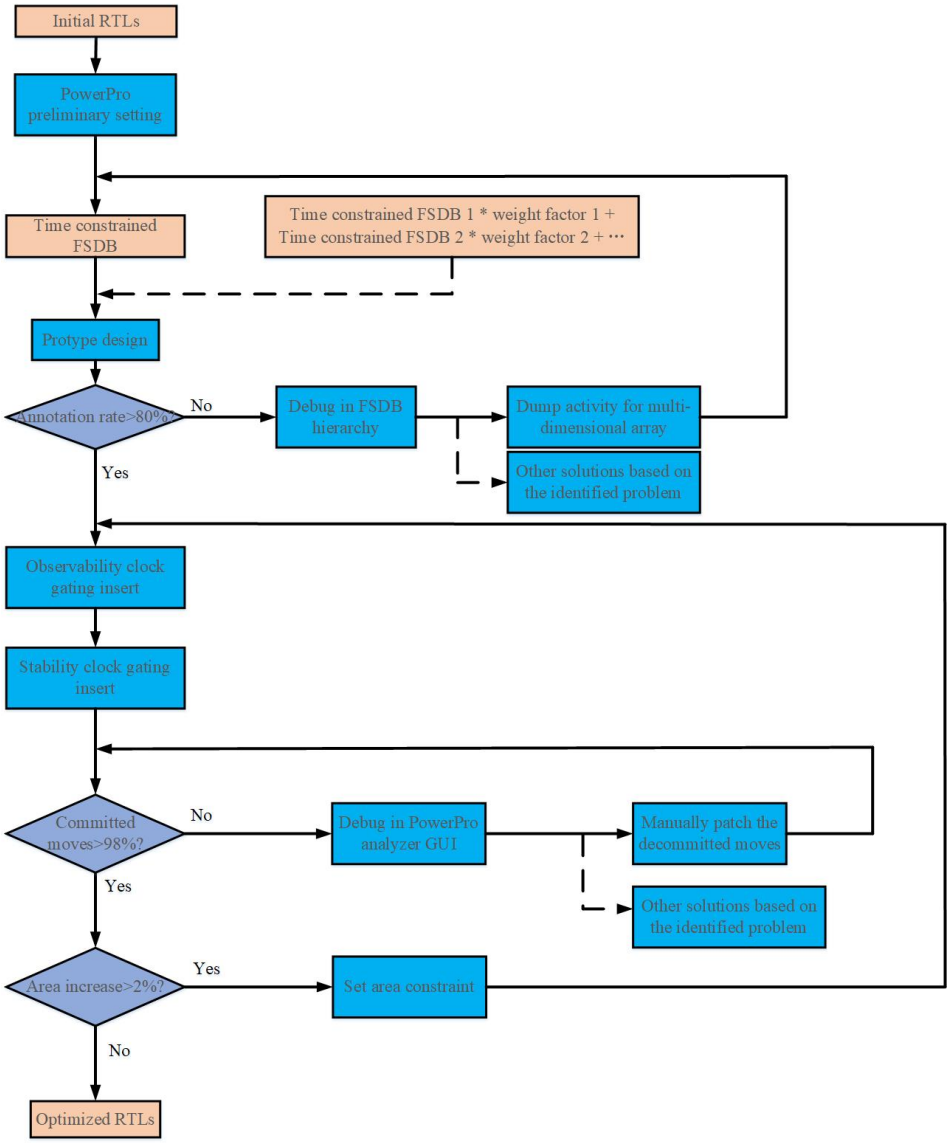


Figure 3.6: Concept diagram of power optimization

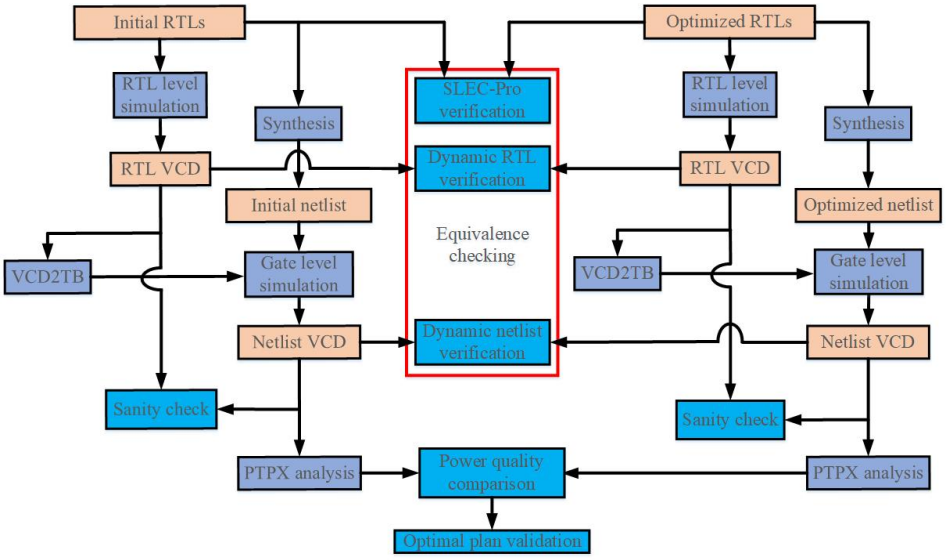


Figure 3.7: Post optimization functional verification flow

in the functional verification environment (UVM environment). The in-house tool VCD2TB [24] introduced in the previous chapters has linked the RTL verification environment with the netlist DUT simulation. This enables the power analysis on the gate level.

The flow in figure 3.7 illustrates the procedures of quality check, starting with static functionality check(using SLEC-Pro). If all the gated logics are proven, the SLEC-Pro checking passes. Otherwise, SLEC-Pro will provide remedy plans to conduct an error-clean optimization. Dynamic verification, which is similar to the sanity check, is implemented by comparing the metrics including reference signal and memory activity from the initial RTL design simulation and those from the optimized RTL design simulation. The primary goal of this check is to ensure netlist sim correstness but it also indirectly shows that initial and optimized rtl have same outputs (reference signals and memory activities) with a basic level of confidence. Dynamic verification should also be done after netlist simulation to validate the pre-layout netlist designs generated in the synthesis process.

The early power estimation is based on the gate level prototype with physical characterization that mimics the behavior of the pre-layout netlist design. Hence, the accuracy of power estimation made by PowerPro is not as high as the accuracy of power estimation made at netlist level. In the synthesis process, the SCGE) report and the summary report, which provides timing slack and active area size, should be compared between the initial netlist design and the optimized netlist design. Timing slack and area increase should meet the conditions before power quality comparison. According to the description of PrimeTimePX in section 2.7.2, PrimeTimePX analysis is utilized as a sign-off process to validate the

improvement of power quality of the optimized design. The comparison is made on DCGE, internal power, switching power, dynamic power and total power. This process gives a comprehensive comparison on the power numbers and power metrics. This step is also used to validate the reliability of the optimal testcase chosen at RTL analysis level. The accuracy of the power estimation made by PowerPro is also compared to the power estimation made by PrimeTimePX in the case study chapter.

3.4 Automatic power analysis and optimization flow

In this section, an automatic power analysis and optimization flow for the whole process is presented in order to integrate all the steps proposed above. This flow starts with the RTL design stage, and continues in with the early analysis stage. An UVM based simulation environment is used to launch the preliminary analysis with the assistance of Ericsson in-house tools. The performance modes of the testcases provided in the verification framework are identified. Therefore, the early power saving potential can be estimated for each testcase using Activity Explorer and PowerPro. The next stage is the power optimization stage that is overlapping with the power estimation stage. Based on the comparison between the power estimation of the testcases, an proper optimization testcase is selected. Based on this testcase, PowerPro modifies the original RTL design to generate an optimized design. A static verification method is provided by PowerPro to check the equivalence of the optimized design functionality with the functionality of the original design. A dynamic verification can be done in the UVM based simulation environment. If the all the RTL level verifications are passed, the optimized RTL should be synthesized in order to carry out analysis at the netlist level. a sanity check between RTL level simulation and gate level simulation should be conducted to validate the pre-layout netlist. The dynamic verification at gate level gives additional assurance of the functionality in the optimized design. The final stage is to validate the improvement on power quality of the optimization. In this stage, PrimeTimePX is used as a sign-off tool to give a precise power analysis at gate level.

Figure 3.8 depicts the summary flow distributed into five stages, namely Hardware Description Language (HDL) design stage, RTL analysis stage, optimization stage, synthesis stage, and gate level stage. The tool chain presented in the figure connects those five stages, the blue arrow illustrates the path of the initial design while the red arrow denotes the path of the optimized design.

In the next chapter, case studies on EMCA IP blocks are conducted and discussed. The results are analysed in order to consolidate the efficiency of this flow.

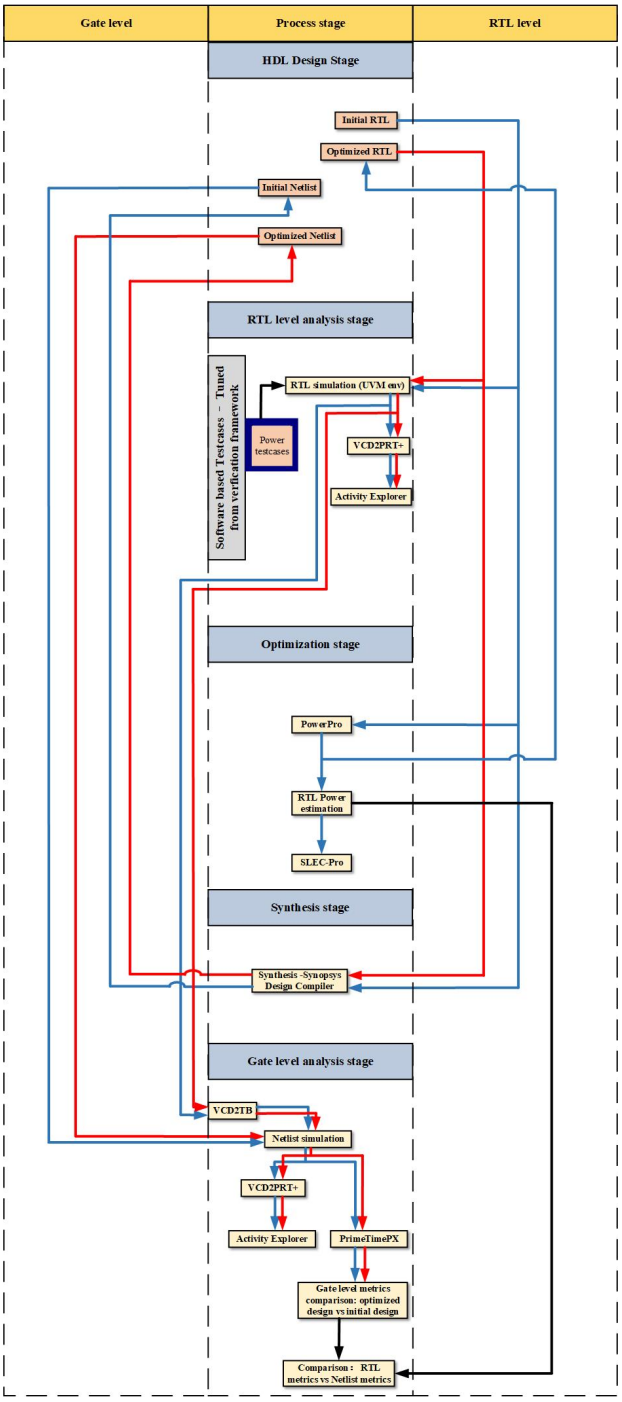


Figure 3.8: Automatic power analysis and optimization flow

Chapter 4

Results and discussion

In this chapter, experiments are designed to evaluate the automatic power analysis and optimization flow, and to collect empirical data on the expected power savings. Four EMCA IP blocks are chosen for the case studies. All of them do not employ hierarchical clock gating so they rely on a very efficient local clock gating. A brief introduction of these IP blocks is provided. The experimental results are collected and analyzed based on each step of the flow. Discussions will be extended from power saving potential estimation to the validation of power quality improvement. This chapter is aiming to validate the results expected from the flow, and to demonstrate the generality of this flow. This means that the flow should be applicable to any IP or any sub-blocks of an IP block. The four IPs are optimized based on the testcases chosen from verification environment. We want to find out how much CG potential can be fulfilled by PowerPro on those blocks based on the optimal testcases selected by the flow. Also, we manually tailored a testcase with ladder type for one of the IPs in order to demonstrate the effectiveness of the optimization using synthetic testcase(as metioned in section 3.1.1). It is designed to prove that this ladder testcase can be used as a default option for optimization if no specific testcase is provided. The block sizes of those blocks are listed in table 4.1. In the table, the number of instances are provided for memory count instead of the actual area. Gate count contains both the FF gates and the combinational gates. Power potential of blocks with different sizes are estimated.

4.1 Optimization of block1

Block1 is a fast and flexible (programmable) job scheduler in hardware that is broadly used in Ericsson ASIC Baseband Modules. Even if just a 1% decrease in the dynamic power consumption of block1 could be achieved, the total energy reduction in all of the Ericsson products that have block1 would be very substan-

Table 4.1: Block sizes w.r.t experimented blocks

Block name	**Block 1a (Netlist)**	**Block 1b (Netlist)**	**Block 1c (Netlist)**	**Block 1d (Netlist)**	**Block2 (Netlist)**
Number of FFs	314k	16k	3k	73k	49k
Number of memories	40	59	11	30	6
Gate count	5054k	298k	59k	1914k	859k
Block name	**Block 3a (Netlist)**	**Block 3b (Netlist)**	**Block 3c (Netlist)**	**Block 4a (RTL)**	**Block 4b (RTL)**
Number of FFs	66k	3k	16k	17k	15k
Number of memories	0	0	8	18	18
Gate count	1549k	33k	277k	101k	97k

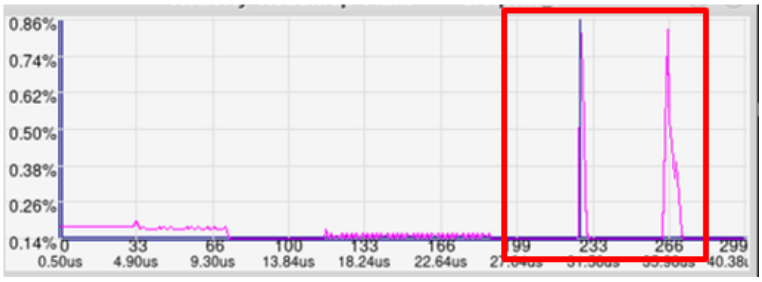
tial. Block1 is a large hierarchical block that has many sub-blocks. According to the user manual of PowerPro, the recommended granularity for optimization by PowerPro is between 100k FFs and 300k FFs. Block1 has around 499k FFs. Hence, we decided to take a botton up approach to optimize the sub-blocks separately in block1. Block1 has many functions, such as memory allocation, Resource scheduling, Queue handling and Timer Queue Pool. Those functions are achieved by the main sub-modules inside of block1. For example, block 1a is an activity control processor in the block1 design, that has the largest area(314k FFs) in block1. Block 1b is a event signal processor in block1, that has 16k FFs. Block 1c is a timer queue pool within the block1 design, that has 3k FFs. Block 1d is a resource pool scheduler in block1, that has 73k FFs. Those four sub-modules are chosen for the following tests.

The experiments are planned as follows. Because block1 has different usages in different products, it is hard to define the usage rate for all the testcases. Theoretically, an optimal testcase for optimization should have random accesses to different sub-modules in a design, and have idling states between these accesses. This kind of testcase covers the general active mode of each sub-block that triggers OBCG, and covers the inactive mode of each sub-block that triggers SBCG. Therefore, the random tests that usually involve general situations in a design are developed to optimize block1. At the beginning, two testcases, that are tuned from the verification environment, are selected to simulate block1. We assume that each testcase has 50% usage rate in reality. In the later discussions, we call those two testcases as testcase1 and testcase2. Visualization is propagated down to the sub-module level (block 1a, block 1b, block 1c and block 1d). Then, the activity profile of each module can be captured. ASA is calculated to identify the performance point of each testcase. From the activity profile window, optimization and analysis time interval is chosen. After the pre-optimization analysis, three candidate optimizations are designed on the basis of those testcases with regard to their performance modes. For block1, three optimizations have been designed with two testcases, given below,

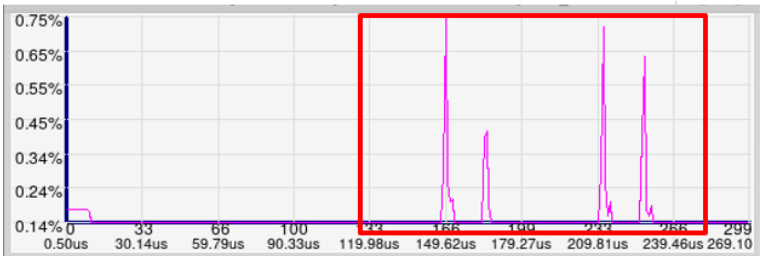
1. Standalone optimization using testcase1
2. Standalone optimization using testcase2
3. Mix-testcase optimization using both testcases: testcase1(weight factor1: 0.5) and testcase2(weight factor2: 0.5).

4.1.1 Optimization on block 1a

The activity profiles of block 1a are shown in figure 4.1. According to the method provided in section 3.1.2, the activity time windows are chosen for these two testcases separately; illustrated by the red square in figure 4.1 a) and in figure 4.1 b). The ASA is calculated for both testcases. The ASA in figure 4.1 a) is around 0.17%, and the ASA in figure 4.1 b) is around 0.16%. Therefore, the performance modes of both testcases are defined as low activity.



(a) Activity profile by running testcase1



(b) Activity profile by running testcase2

Figure 4.1: Activity profile of block 1a running two different testcases

Figure 4.2 illustrates DCGE comparison and SCGE) comparison between the three optimizations based on PowerPro reports. From figure 4.2, it is noticeable that testcase1 optimization and testcase2 optimization have similar CGE improvement (both DCGE and SCGE), while testcase2 optimization performed more observability optimization than testcase1 optimization. This reveals that, although, testcase1 optimization and testcase2 optimization have the same percentage of increase on CGE, the constitution of the gating conditions in the two optimizations are different. This could lead to better power quality in one of those optimizations in a specific usage scenario. It is reasonable to conclude, depending on the CGE comparison, that mix-testcase optimization is the optimal optimization with the highest DCGE and the highest SCGE).

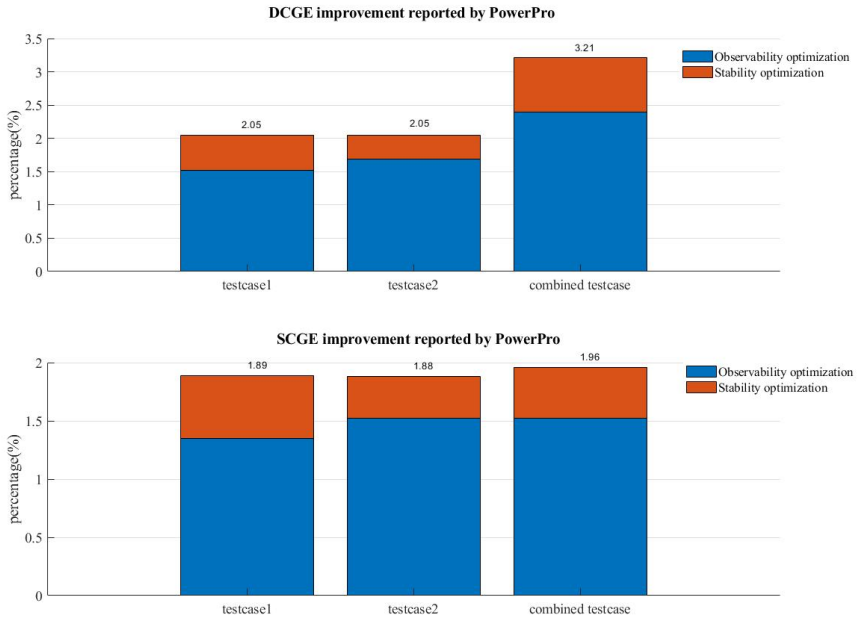


Figure 4.2: RTL CGE comparison between three optimizations for block 1a

Table 4.2: Power metric comparison for block 1a at gate level

Metric comparison		Testcase1 optimization	Testcase2 optimization	Mix-testcase optimization
SCGE increased		1.68%	1.68%	1.66%
Timing analysis		passed	passed	passed
Active area variation		0.06% lower	0.03% higher	0.14% higher
Analysis on testcase1	Internal power decreased	15.7%	15.5%	16.7%
	Switching power decreased	18.03%	17.5%	19%
	Dynamic power decreased	16.5%	16.1%	17.5%
	DCGE increased	1.7%	1.7%	1.9%
	Total power decreased	6.7%	6.3%	6.8%
Analysis on testcase2	Internal power decreased	16.7%	16.7%	17.9%
	Switching power decreased	19%	18.4%	18.4%
	Dynamic power decreased	17.5%	17.3%	18%
	DCGE increased	1.7%	1.7%	1.8%
	Total power decreased	6.8%	6.5%	7%

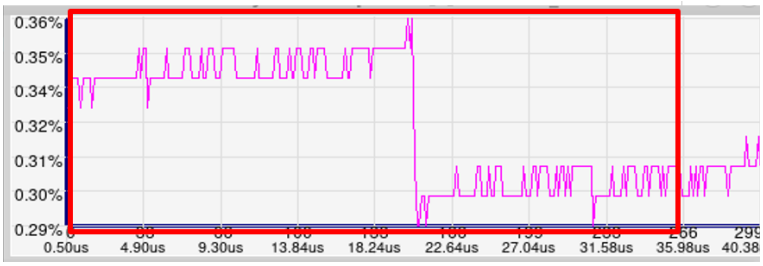
Table 4.2 collects the data for power metrics comparison between the three optimized designs. The analysis time windows for the two testcases are the same as the optimization time windows highlighted in figure 4.1. Comparing the analysis results running testcase1 with those three optimized netlists, mix-testcase optimization achieves the best performance on DCGE increase. Although testcase1 optimization has similar SCGE) and DCGE improvements as in testcase2 optimization, testcase1 optimization achieves slightly more dynamic power reduction than testcase2 optimization. This has already been assumed based on the power estimation reports at RTL level. The analysis results running testcase2 with the three optimized designs are shown in the lower part of 4.2. The results match the estimation at RTL level, which validates that mix-testcase optimization is the most effective optimization for block 1a. A 17.5% reduction in dynamic power is expected for mix-testcase optimization in testcase1 scenario. A 18% reduction in dynamic power is expected for mix-testcase optimization in testcase2 scenario. Testcase1 optimization and testcase2 optimization have the equivalent quality of power savings in both scenarios. Also, it is surprising to see that the standalone testcase optimization, not only, achieves high quality on power savings in the corresponding usage scenario, but also reduces a large amount of power consumption in other usage scenarios.

4.1.2 Optimization of block 1b

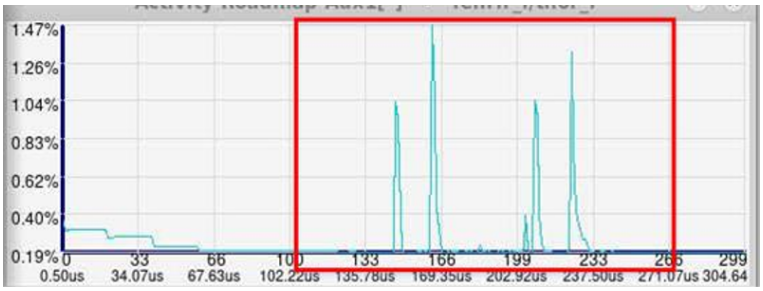
The activity profiles of block 1b are shown in Figure 4-3. Optimization windows with regard to each testcase are highlighted in the figure. The ASA calculated in figure 4.3 a) is 0.32%. The ASA in figure 4.3 b) is 0.27%. Therefore, the performance points of both testcases are defined as low activity. The power estimation at RTL level is implemented on block 1b. The CGE results are shown in figure 4.4. Based on the quality analysis, the order of the three optimizations regarding the potential of power improvement are as follows: mix-testcase optimization> testcase1 optimization > testcase2 optimization.

Table 4.3: Power metric comparison for block 1b at gate level

Metric comparison		Testcase1 optimization	Testcase2 optimization	Mix-testcase optimization
SCGE increased		7.48%	7.7%	7.56%
Timing analysis		passed	passed	passed
Active area variation		0.029% higher	0.023% higher	0.008% higher
Analysis on testcase1	Internal power decreased	1.9%	1.6%	1.9%
	Switching power decreased	25.9%	24.1%	25.9%
	Dynamic power decreased	2.8%	2.4%	2.8%
	DCGE increased	9.1%	7.7%	8.9%
	Total power decreased	1.2%	1.1%	1.2%
Analysis on testcase2	Internal power decreased	13.9%	14.4%	14.2%
	Switching power decreased	25%	25.87%	25.6%
	Dynamic power decreased	16.6%	17.2%	16.9%
	DCGE increased	6.9%	7.3%	7.1%
	Total power decreased	1.61%	1.68%	1.68%



(a) Activity profile by running testcase1



(b) Activity profile by running testcase2

Figure 4.3: Activity profile of block 1b running two different testcases

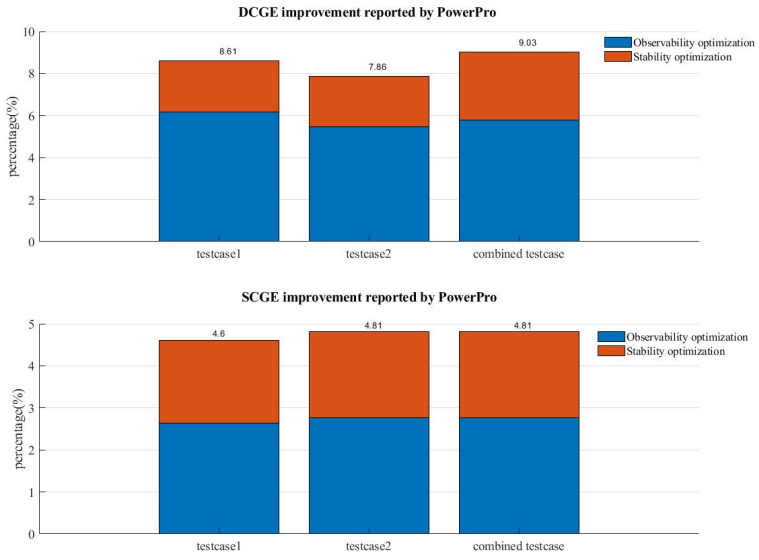


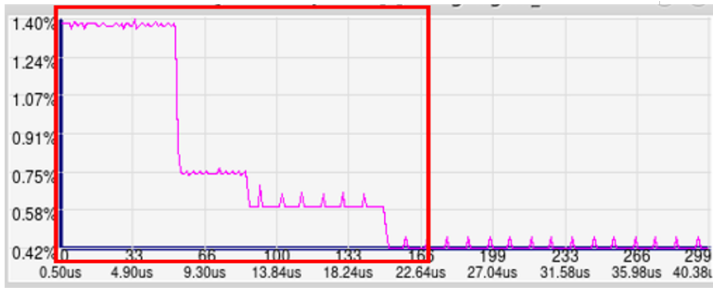
Figure 4.4: RTL CGE comparison between three optimizations for block 1b

The results of the power analysis at gate level are listed in table 4.3. The analyses are conducted on the two testcases metioned above. The time windows chosen for the analyses are the same as the optimization windows. The analysis running simulation of testcase1 shows that mix-testcase optimization and testcase1 optimization have the best quality of power savings. The analysis running simulation of testcase2 shows that testcase2 optimization achieves the highest improvement on DCGE. Mix-testcase optimization has slightly lower performance on power reduction than testcase2 optimization, which can be suspected to be due to noises generated during the processes. Overall, mix-testcase optimization is the optimal optimization for block 1b. Beyond that, the difference between the power reduction rate in the two testcases' analyses is caused by the initiation of memories in the selected analysis window for testcase1. The selected analysis window for testcase 2 does not include the memory initialization phase. This is illustrated in the memory utilization comparison where simulations are run separately using two testcases. The total memory utilization rate of block 1b running testcase1 is 22.65%, and the total memory utilization rate of block 1b running testcase2 is 414.08%. This leads to a large difference in the internal power between these two testcases. The memory gating efficiencies in all IPs that are used in the experiments are up to 99%. Therefore, memory gating is not performed in the experiments.

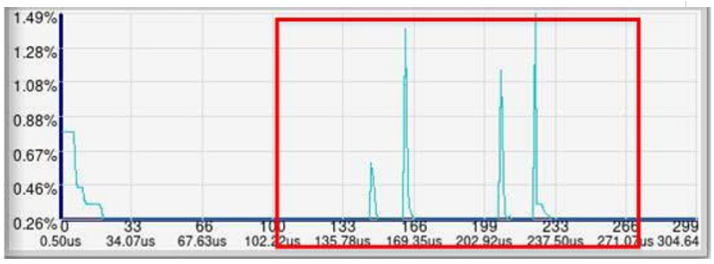
4.1.3 Optimization of block 1c

This section presents the analysis and optimization on block 1c. The activity profiles for block 1c and the defined optimization time windows are displayed in figure 4.5. The ASA calculated in figure 4.5 a) is 0.43, while the ASA in figure 4.5 b) is 0.33. Hence, testcase1 is defined as a typical mode for block 1c, and testcase2 is defined as a low activity mode for block 1c. The results of power estimation at RTL level are illustrated in figure 4.6. It shows that testcase2 optimization is the optimal optimization. This reveals that, for an IP block, the optimization of the combined testcases is not always better than the single testcase optimization. The reason for this is that PowerPro commits all the possible CG moves for the optimization of the combined testcases, but only commits effective CG moves for single testcase optimization.

The quantitative data collected for block 1c are listed in Table 4.4. From the data, the conclusion can be drawn that testcase2 optimization has the highest increase in SCGE). In the analysis running testcase1 simulation as well as running testcase2 simulation, testcase2 optimization achieves the highest increase in DCGE. The dynamic power reduction yeilded by three optimizations are equivalent to each other. Therefore, testcase2 optimization is the optimal testcase. One statement should be made here: the differences between dynamic power between the three optimizations under the two analyses are less than 0.03uW, which is negligible. CGEs should be the primary metrics to define the quality of power savings.



(a) Activity profile by running testcase1



(b) Activity profile by running testcase2

Figure 4.5: Activity profile of block 1c running two different test-cases

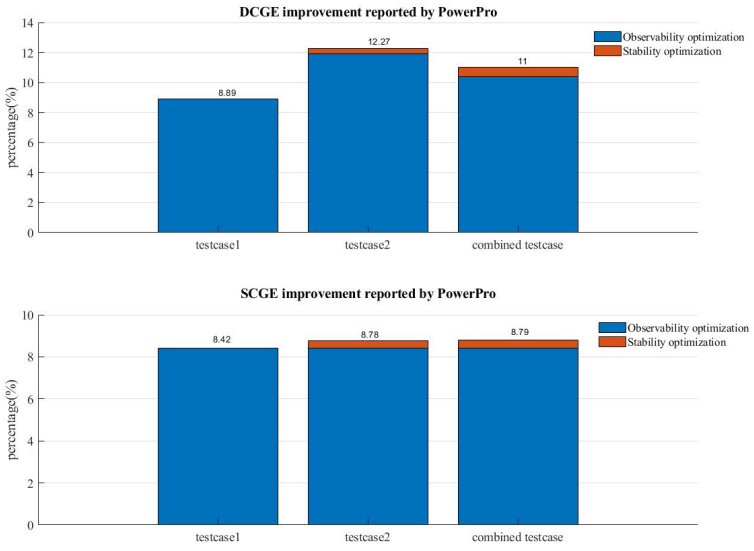


Figure 4.6: RTL CGE comparison between three optimizations for block 1c

Table 4.4: Power metric comparison for block 1c at gate level

Metric comparison		Testcase1 optimization	Testcase2 optimization	Mix-testcase optimization
SCGE increased		10.69%	11.02%	10.88%
Timing analysis		passed	passed	passed
Active area variation		0%	0.03% higher	0.01% lower
Analysis on testcase1	Internal power decreased	1.1%	1.1%	1.1%
	Switching power decreased	20.3%	18.7%	21.1%
	Dynamic power decreased	1.77%	1.72%	1.79%
	DCGE increased	7.9%	8.3%	8%
	Total power decreased	1%	0.09%	1%
Analysis on testcase2	Internal power decreased	19.0%	18.3%	18.7%
	Switching power decreased	24.8%	23.1%	25.6%
	Dynamic power decreased	20.8%	19.8%	20.9%
	DCGE increased	10.6%	11%	10.8%
	Total power decreased	3.1%	3.1%	3.1%

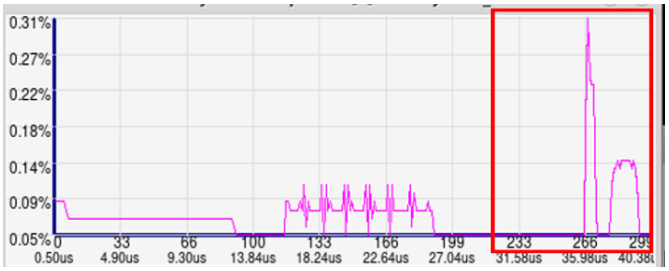
4.1.4 Optimization of block 1d

The last sub-block that gets optimized in block1 is block 1d. The activity profile and the corresponding time window for each testcase are displayed in figure 4.7. The ASA calculated in figure 4.7 a) is 0.09, and the ASA in figure 4.7 b) is 0.08. Hence, Testcase1 is defined as a low activity mode as well as testcase2. The CGE comparison between three optimizations at RTL level is shown in figure 4.8. According to the method developed in the flow, mix-testcase optimization should has the best performance. However, we have discovered the drawback in mix-testcase optimization that ineffective moves might be committed by PowerPro. Therefore, analysis at gate level is needed to make a solid decision.

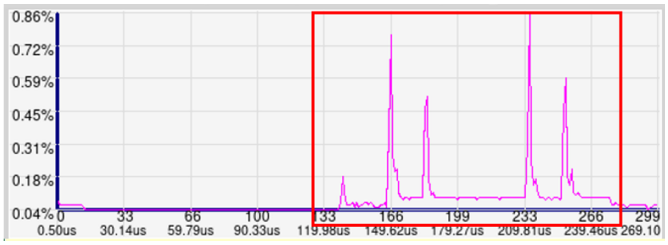
Table 4.5: Power metric comparison for block 1d at gate level

Metric comparison		Testcase1 optimization	Testcase2 optimization	Mix-testcase optimization
SCGE increased		2.67%	3.05%	3.24%
Timing analysis		passed	passed	passed
Active area variation		0.05% lower	0.03% lower	1% lower
Analysis on testcase1	Internal power decreased	9.2%	12%	10.8%
	Switching power decreased	10.5%	14.3%	12.8%
	Dynamic power decreased	9.6%	12.8%	11.5%
	DCGE increased	2.6%	3%	3.1%
	Total power decreased	4.4%	5.3%	5.3%
Analysis on testcase2	Internal power decreased	8.9%	11.7%	10.5%
	Switching power decreased	10.2%	13.9%	11.7%
	Dynamic power decreased	9.4%	12.4%	10.9%
	DCGE increased	2.6%	3%	3.1%
	Total power decreased	4.3%	5.3%	5.2%

Table 4.5 lists the data recorded in netlist analysis. The results show that, for both the analysis running testcase1 and the analysis running testcase2, testcase2



(a) Activity profile by running configuration test



(b) Activity profile by running tasking test

Figure 4.7: Activity profile of block 1d running two different test-cases

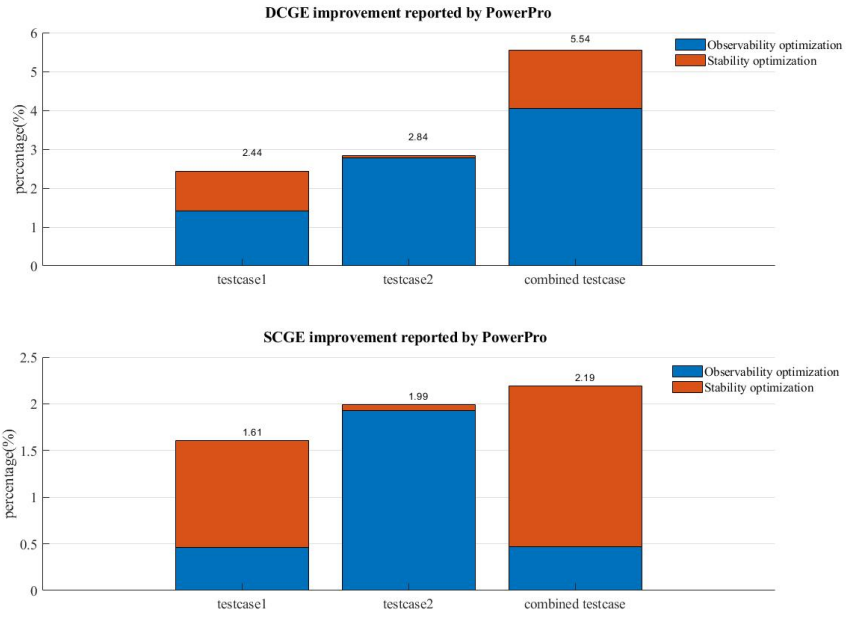


Figure 4.8: RTL CGE comparison between three optimizations for block 1d

optimization has the best performance on dynamic power reduction. But mix-testcase optimization has the highest DCGE increase among three optimizations. For the sake of achieving the highest performance on power savings, testcase2 should be chosen to optimize block 1d.

4.2 Optimization of block2

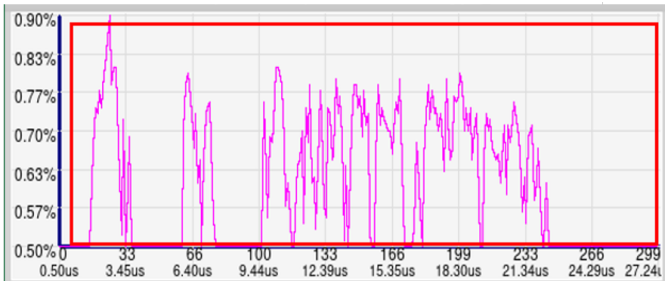
The second case study is conducted on block2. Block2 is a direct memory access controller in EMCA that has around 49k FFs. For this module, we manually tailored a synthetic testcase as mentioned in section 3.1. Beyond that, another two usage-specific testcases are provided for analysis on block2. The test is designed following the same procedures as the experiments presented above.

The activity of these two testcases are profiled in figure 4.9. As can be seen in the figure, the ASA for testcase1 is 0.6, and the ASA for testcase2 is 1.3. We can derive that testcase1 is operating at typical mode while testcase2 is operating at high activity mode. The full time window is chosen for both testcases to include the main activities distributed through the whole operation. We assumed that block2 is running configuration 35% of the time and is performing tasks 65% of the time. The activity profile of the synthetic testcase is shown in figure 4.10. This testcase combined reset mode, idle mode, low activity mode, medium activity mode and high activity mode. The full time window is chosen for this synthetic testcase. Four optimizations are designed as below,

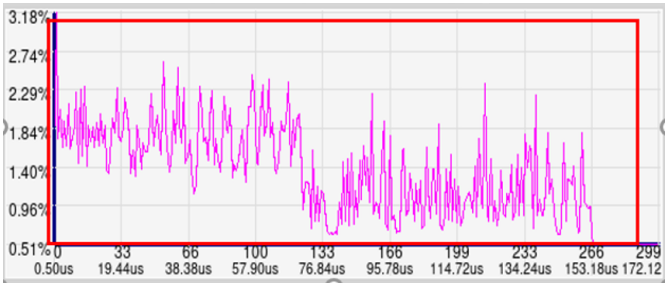
1. Standalone optimization using configuration test
2. Standalone optimization using tasking test
3. Mix-testcase optimization using both testcases: tasking test(weight factor1: 0.65) and configuration test(weight factor2: 0.35)
4. Synthetic testcase optimization: Ladder type with: "reset", "idle", "low", "medium", "high", "medium", "low", "idle".

The comparison of CGEs between the four different optimizations for block2 at RTL estimation stage is shown in figure 4.11. Based on the result of the comparison, mix-testcase optimization should be chosen as the optimal testcase.

The power estimation at netlist level (in Table 4.6) gives more precise data on the power quality improvements with regard to the three optimized designs. From the table, it is clear that mix-testcase optimization is not the optimal optimization. The reason for this has been explained before; that redundant CG moves has been committed by the mix-testcase optimization. The configuration test optimization and the tasking test optimization show similar performances in power



(a) Activity profile by running configuration test



(b) Activity profile by running tasking test

Figure 4.9: Activity profile of block2 running two different testcases

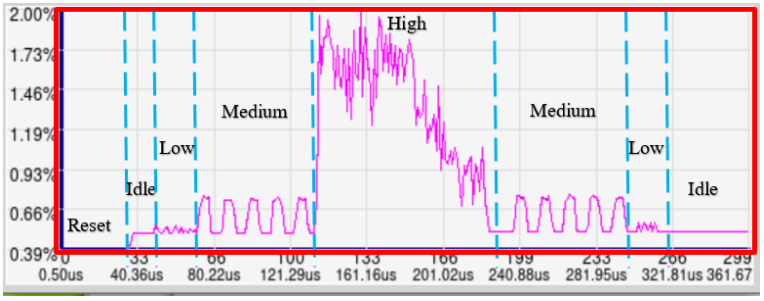


Figure 4.10: Activity profile of block2 running synthetic testcase

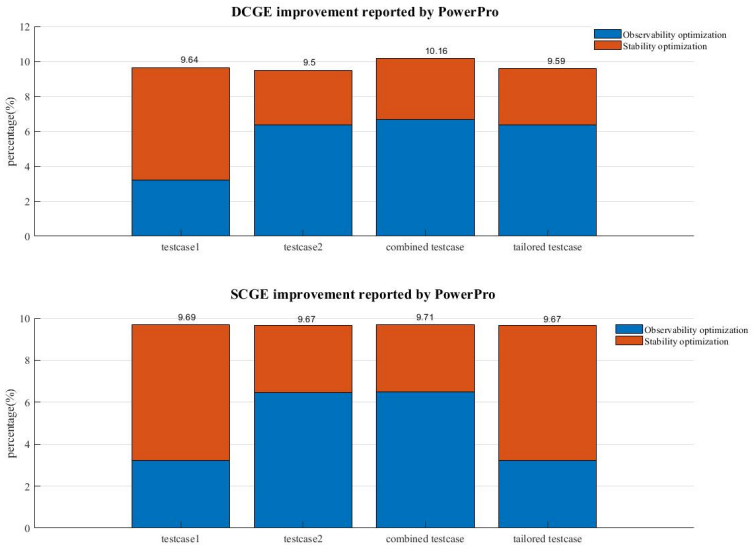


Figure 4.11: RTL CGE comparison between four optimizations for block2

Table 4.6: Power metric comparison for block2 at gate level

Metric comparison		Configuration test optimization	Tasking test optimization	Mix-testcase optimization
SCGE increased		8.36%	8.36%	10.88%
Timing analysis		passed	passed	passed
Active area variation		0.5% higher	0.4% higher	0.01% lower
Analysis on configuration test	Internal power decreased	38.7%	38.7%	36.9%
	Switching power decreased	41.4%	41.4%	39.7%
	Dynamic power decreased	39.6%	39.6%	37.9%
	DCGE increased	7.9%	7.9%	7.8%
	Total power decreased	33.3%	33.3%	31.9%
Analysis on tasking test	Internal power decreased	33.9%	33.9%	31.5%
	Switching power decreased	35.7%	35.7%	32.9%
	Dynamic power decreased	34.5%	34.5%	32%
	DCGE increased	7.8%	7.8%	7.5%
	Total power decreased	29%	29.4%	27.3%

Table 4.7: Netlist power metrics for block2 using synthetic testcase

Power metrics	Synthetic testcase optimization for Block2
SCGE increased	8.35%
Timing analysis	passed
Area variation	0.46% higher
Internal power decreased	34%
Switching power decreased	37.4%
Dynamic power decreased	35.1%
DCGE increased	7.3%
Total power decreased	29.9%

reduction (around 39% of dynamic power reduction in configuration test analysis, and around 34% of dynamic power reduction in tasking test analysis). It is recommended to always trust standalone optimization. But the mix-testcase optimization can be used as a contrast and, in some cases, as an optional optimization. In summary; although dynamic power estimation is dependent on the usage scenario and the technical libraries provided, a 30% of reduction in total power is still a very promising result for a bug-free design that is is ready to release.

Power metrics using the synthetic testcase are listed in table 4.7. 35% dynamic power savings and 30% total power savings show a good enough improvement for block2. This illustrates that although the synthetic testcase does not represent the actual operations, it still has very good performance on optimization. This testcase should be created as a default option if no specific testcase is provided.

4.3 Power saving potential of other EMCA IP blocks

After the validation of this flow, we employed the flow on two more blocks to identify the power saving potential in those blocks. Block3 is a block that consists of associated memories and controllers. To simplify the synthesis process, the sub-blocks of block3 are optimized separately. The improvement on the power metrics is illustrated by the netlist analysis results shown in table 4.8. Block3 is a shared level 3 cache controller for program and data. It is a hierarchical design that has 175k FFs. Block 3a that has 67k FFs achieved 12.7% reduction on dynamic power. Alough Block 3b is a really small design that only has 3k FFs, the dynamic power of it decreased by around 60%. Block 3c, that involves 16k FFs, also has very distinct improvement on power savings with 20.8% reduction on dynamic power. Those results show a very promising future that could lead to a large amount of energy saving in all the products that involve block3.

Block4 is a design that sorts data from an external interface into antenna specific buffers in the memory and notifies softwares when it is available for further processing. We only analyzed the potential of power savings in this block at RTL

level due to the lack of synthesis resources. However, the metric numbers reported by PowerPro give us a rough prediction. The power metric estimation of block4 is presented in table 4.9. PowerPro estimated 14.7% decrease of dynamic power in block 4a(with 17k FFs), and 5.2% reduction on dynamic power in block 4b(with 15k FFs). Based on the comparison in the previous experimental results between PrimeTimePX and PowerPro, there are reasons to guess a larger reduction of power in the precise analysis at netlist level. To sum up, this estimation reflects the prospect of the attainable quality of clock gating in block4.

Table 4.8: Power metric for block3 at gate level

Metric comparison	Block3		
	Block 3a	Block 3b	Block 3c
SCGE increased	2.4%	20.38%	4.87%
Timing analysis	passed	passed	passed
Area variation	0.3% higher	1.6% higher	0.1% higher
Internal power decreased	12.2%	60.2%	20.8%
Switching power decreased	13.7%	57.8%	20.8%
Dynamic power decreased	12.7%	59.3%	20.8%
DCGE increased	2.1%	23.2%	5.4%
Total power decreased	11.1%	55.8%	12.3%

Table 4.9: Power metric for block4 at RTL stage

Metric comparison	Block4	
	Block 4a	Block 4b
SCGE increased	0.23%	-0.69%
DCGE increased	3.05%	1.61%
Dynamic power decreased	14.7%	5.2%
Total power decreased	4.8%	1.9%

4.4 Summary

The end to end runtime comparison between all the experimented blocks is shown in table 4.10. In general, the optimization runtime in this flow is much lower than the synthesis runtime for an ASIC design, and also much lower than the time consumed by the low-power RTL coding in the previous flow. This low-power RTL coding could take few days to optimize a normally sized ASIC design. Only block 1d took more time to optimize than to synthesize. This could be explained by the large multi-dimensional arrays in block 1d that consumed a very long time in OBCG. This automatic power analysis and optimization flow requires very low efforts to optimize an ASIC design, which stands out in terms of competitiveness from other industrial low-power methods.

The max dynamic power reductions of all the analyzed blocks achieved by this flow are summarized in table 4.11. These data are visualized in figure 4.12. On

Table 4.10: End to end runtime comparison between all the experimented blocks

Block name	Average optimization runtime	Average synthesis runtime
Block 1a	3h 42min	12h 10min
Block 1b	20min	1h 25min
Block 1c	4min	25min
Block 1d	14h 42min	10h 30min
Block2	1h 20min	2h
Block 3a	1h 05min	5h
Block 3b	1min	17min
Block 3c	1h	44min
Block 4a	43min	-
Block 4b	53min	-

average, this flow has shown a good ability of design optimization on the aspect of power savings. As mentioned in the flow description, a recommended testcase should represent the behavior of real opearations. The mix-testcase optimization usually covers those conditions, but it is unfriendly to the netlist analysis. Also, the performance of the mix-testcase optimization is fluctuating. Therefore, it is recommended to provide a synthetic testcase that combines valleys and hills i.e. multiple analysis points so that it is reproducible and comparable at netlist level. Although this flow has shown a very good performance so far, it should not be used in the start point of an IP design process. The automatic modified design always uses complicated gating conditions passing through many CCs and transferring multiple sub-blocks. Hence, it is very difficult to manually make code modifications that could introduce complex functional errors to the design after automatic optimization. And the generated RTLs cannot be optimized again by PowerPro. Hence, it is suggested to run for pipecleaning purposes during the first RTL release. Considering the automatic optimization is a one time process, it should be run after the final RTL is ready and verified to acquire the optimized RTL for the final release. The human developed RTL is always the start point for a bug fix or migration to a new project. The auto-generated RTL code is used only after all features/fixes are in place for a release to the integration and back end team. It should never be used as a baseline for a new project or bug fixes.

To summarize, this novel flow has performed very well on analyzing and optimizing generic IP blocks. The promising results have been obtained from blocks of different sizes and functions. The case studies using this flow has combined mature technologies to realize the goals of the thesis. The runtime of this flow is significantly lower than the previously designed flows. Hence, the developed flow has proven to be ready for usage at the industrial level and to be plugged into the IP design flow in Ericsson.

Table 4.11: Max observed dynamic power reduction that corresponds to the optimal testcase

Block name	**Block 1a (Netlist)**	**Block 1b (Netlist)**	**Block 1c (Netlist)**	**Block 1d (Netlist)**	**Block2 (Netlist)**
Max dynamic power reduction	18%	17.2%	21.4%	12.8%	39.6%
Block name	**Block 3a (Netlist)**	**Block 3b (Netlist)**	**Block 3c (Netlist)**	**Block 4a (RTL)**	**Block 4b (RTL)**
Max dynamic power reduction	12.7%	59.3%	20.8%	14.7%	5.2%

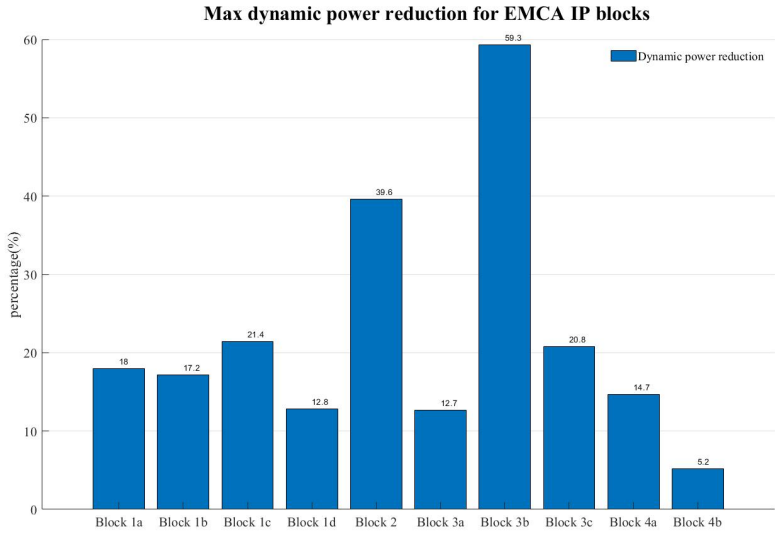


Figure 4.12: Max observed dynamic power reduction for all the experimented blocks

Chapter 5

Conclusions and future work

The main scope of this thesis has been to design an automatic flow for optimizing IP blocks without having full understanding of the design along with integrated formal verification guarantees for bug-free RTL.

Normally an ASIC design has more than ten levels of hierarchy and is linked by hundreds of mix-language HDL files. It is very time-consuming to anchor an enable condition through multiple modules. Also, discovering the clock gating opportunities manually through multiple clock cycles is very inefficient. To be able to handle this in the old solutions, a complete understanding of the design is required [27]. Also, the designer must be very proficient in uncovering gating opportunities. But still, a manually modified design can miss so many intricate CG opportunities. Also, in a large design, many of the manually gated logics might be partially gated. Manual insertion of clock gates may introduce functional errors, i.e. functional bugs and timing mismatch.

To fully achieve the potential on power reduction, this new flow introduces a new potent approach based on the PowerPro EDA tool: PowerPro. It has proven to uncover and to gate the inefficient logics, and to strengthen the partially-gated logics in a design. The designed flow has been proven to be well-suited to the existing environment for power analysis. Yet, compared to the related flows, the greatest improvement of this flow is the automatic optimization integrated with early power analysis. Instead of focusing on the optimization for specific IP blocks, this thesis proposes a method that generally serves as a boost to low-energy on EMCA IP blocks for 5G products.

This flow has connected Ericsson in-house tools with effective commercial tools together. Different IP blocks are researched by the flow. The experiments have shown very promising results on block1 and block2. As mentioned above, the optimizations of the sub-blocks of block1 have achieved a significant improvement in DCGE and SCGE. For block2, there is a large potential for power reduction. The

dynamic power reduction estimated for block2 is up to 30% within the analytical scenario. The concerns that a specific testcase might optimize only the design in the corresponding scenario have been overturned. It is proven by the experiments that a design optimized with a specific testcase not only shows improvement of power reduction in the same testcase analysis, but that it also consumes less dynamic power when it is run for another testcase.

The exploration of the feasibility of the flow has brought insights into power optimization for EMCA IP blocks within the industrial scope. One possible limitation is that the characterized testcases still are not fully understood. It can be derived from the power estimation results at RTL level that the composition ratios of two gating types are dependent on the testcase provided. The relationship between different testcases and the effects of the constitution of the gating logics need to be researched further in the future. If such a study on the relationship between power testcase and optimization quality can be done, the optimal power testcase can be designed to realize the best optimization on IPs in general. Based on the results we got from this project, the next step is to pilot this flow in a real production environment. And we need to resolve version control and IP process issues that arise with automated RTL generation. Also, we need to analyze and optimize more blocks across different IP programs in the future.

References

[1] 'a technical look at 5g energy consumption and performance' [online]. available https://www.ericsson.com/en/blog/2019/9/energy-consumption-5g-nr [accessed 11-may-2019].

[2] 'energy efficiency in gpu applications, part 1' [online]. available https://community.arm.com/developer/tools-software/graphics/b/blog/posts/energy-efficiency-in-gpu-applications-part-1 [accessed 11-may-2019].

[3] Ioannis Savvidis. Power analysis & optimization. In *Ericsson Internal document*.

[4] 'reports on the increasing energy consumption of wireless systems and digital ecosystem' [online]. available https://ehtrust.org/science/reports-on-power-consumption-and-increasing-energy-use-of-wireless-systems-and-digital-ecosystem/ [accessed 11-may-2019].

[5] Zhang T. Li Z., Wang X. *5G+ Future Cornerstone of a Digital Society. In 5G+*. Springer, 2021.

[6] Michael Keating, David Flynn, Rob Aitken, Alan Gibbons, and Kaijian Shi. *Low Power Methodology Manual For System-on-Chip Design*. Springer Publishing Company, Incorporated, 2007.

[7] B. M. Rao and V RAVISHARADHYAH. Clock power optimizations in vlsi design at advanced technology nodes. *International Journal of Computer Applications*, 169:29–34, 2017.

[8] 'a review paper on cmos, soi and finfet technology' [online]. available https://www.design-reuse.com/articles/41330/cmos-soi-finfet-technology-review-paper.html [accessed 11-may-2019].

[9] Neerajnayan Balachandran. Low power memory controller subsystem ip exploration using rtl power flow : An end-to-end power analysis and reduction

methodology. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2020.

[10] 'how 5g is influencing silicon design' [online]. available https://semiengineering.com/how-5g-is-influencing-silicon-design/ [accessed 11-may-2019].

[11] Mentor Internal document. Powerpro user manual.

[12] Rakesh Chadha and J. Bhasker. *An ASIC Low Power Primer: Analysis, Techniques and Specification*. Springer Publishing Company, Incorporated, 2012.

[13] 'designing for peak power in mobile electronic devices' [online]. available http://sensiot.be/designing-for-peak-power-in-mobile-electronic-devices/ [accessed 11-may-2019].

[14] Synopsys Internal document. Primetimepx user guide.

[15] K. Nose and T. Sakurai. Analysis and future trend of short-circuit power. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(9):1023–1030, 2000.

[16] 'what is a finfet?' [online]. available https://eepower.com/technical-articles/what-is-a-finfet/ [accessed 11-may-2019].

[17] M. Jurczak, N. Collaert, A. Veloso, T. Hoffmann, and S. Biesemans. Review of finfet technology. In *2009 IEEE International SOI Conference*, pages 1–4, 2009.

[18] Chapter 3 - low power design techniques, design methodology, and tools. In Findlay Shearer, editor, *Power Management in Mobile Devices*, pages 77–115. Newnes, Burlington, 2008.

[19] Ioannis Savvidis. Power savings in mpsoc. Master's thesis, KTH, School of Information and Communication Technology (ICT).

[20] Nainala Vyagrheswarudu, Subrangshu Das, and Abhishek Ranjan. Poweradviser: An rtl power platform for interactive sequential optimizations. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 550–553, 2012.

[21] Jianfeng Liu, Mi-Suk Hong, Kyungtae Do, Jung Yun Choi, Jaehong Park, Mohit Kumar, Manish Kumar, Nikhil Tripathi, and Abhishek Ranjan. Clock domain crossing aware sequential clock gating. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, 2015.

[22] Ericsson AB Ioannis Savvidis. The quest for easy power-aware sw development: A novel approach to dsp code profiling for energy/performance tradeoffs. In *DAC*, 2018.

[23] Ioannis Savvidis. Emca activity monitor. In *Ericsson Internal document*.

[24] Ericsson AB Ioannis Savvidis. Dump, convert and replay: A targeted methodology to mitigating gate-level power simulations effort. In *DAC*, 2019.

[25] 'the powerpro guided flow: A manual use mode for applications that require user intervention' [online]. available https://eda.sw.siemens.com/en-us [accessed 11-may-2019].

[26] 'alchip minimizes dynamic power for high-performance computing asics' [online]. available https://resources.sw.siemens.com/en-us/white-paper-alchip-minimizes-dynamic-power-for-high-performance-computing-asics [accessed 11-may-2019].

[27] Kanika Sahni, Kiran Rawat, Sujata Pandey, and Ziauddin Ahmad. Power optimization of communication system using clock gating technique. In *2015 Fifth International Conference on Advanced Computing Communication Technologies*, pages 375–378, 2015.