Metrics-Driven Front-End Power Optimization for the EMCA DSP

XINYU XUE

Metrics-Driven Front-End Power Optimization for the EMCA DSP

XINYU XUE

Master's Programme, Embedded Systems, 120 credits
Date: March 24, 2025

Supervisors: Ioannis Savvidis, Ritika Ratnu
Examiner: Ahmed Hemani
School of Electrical Engineering and Computer Science
Host company: Ericsson AB
Swedish title: Metrikstyrd front-end effektoptimering för EMCA DSP

Abstract

The increasing energy demands of mobile networks pose significant challenges to sustainability and cost efficiency. In the Ericsson Many-Core Architecture (EMCA), the in-house Digital Signal Processor (DSP) Intellectual Property (IP) block, which is central to high-performance radio signal processing, employs a wide range of power-saving features. However, any further small power optimization in the DSP could make a big difference at Application-Specific Integrated Circuit (ASIC) System on Chip (SoC) level due to the vast number of instances integrated. Prior differential energy analysis has shown that when the DSP employs a Run-fast-then-stop scheme, which is to complete the workload with maximum performance and then enters a power-saving mode for the remaining time slack to reduce power, it is more energy efficient as compared to a Just-in-time scheme, which is trying to adjust the peak performance to elongate the runtime with a lower average power consumption goal. This observation indicates that there are indeed remaining power optimization opportunities for registers, memories, combinational logic, and the lower levels of the clock tree.

This thesis applied a hybrid power analysis and optimization flow to the DSP. The flow combines automated power optimization using PowerPro with integrated formal verification for bug-free Register Transfer Level (RTL) codes and a manual optimization framework targeting potential high-value manual changes. Automated optimizations achieved a 4.49% reduction in dynamic power and a 0.84% improvement in Dynamic Clock Gating Efficiency (DCGE) for the DSP block, with potential increases up to 12.73% power savings and 2.38% DCGE improvement when manual optimizations are applied.

The study demonstrates this methodology to another block (arbiter and router) in EMCA. This block showed a 6.52% reduction in dynamic power and a 1.28% DCGE improvement with PowerPro, with manual optimizations potential of total power savings of 27.7% and DCGE improvements of 5.5%.

This hybrid flow enables IP blocks to be treated as black boxes, simplifying the optimization process for engineers outside the design team. Additionally, the methodology demonstrated minimal impact on area and timing, making it practical for real-world application. This approach sets a foundation for energy-efficient ASIC design, addressing critical sustainability challenges in next-generation mobile networks.

Keywords

Power analysis, Power optimization, Clock gating, Dynamic power, Application specific integrated circuit (ASIC)

Sammanfattning

Mobilnätens ökande energibehov innebär betydande utmaningar för hållbarhet och kostnadseffektivitet. I Ericssons många kärna-arkitektur (EMCA), använder det interna digitala signalprocessorn (DSP) immateriella rättigheter (IP), som är centralt för högpresterande radiosignalbehandling, ett brett utbud av energibesparande funktioner. Men varje ytterligare liten effektoptimering i DSP kan göra stor skillnad på den applikationsspecifika integrerade kretsen (ASIC) System-on-Chip (SoC) nivå på grund av det stora antalet integrerade instanser. Tidigare differentiell energianalys har visat att när DSP använder ett kör-snabbt-sedan-stopp-schema, vilket är att slutföra arbetsbelastningen med maximal prestanda och sedan går in i ett energisparläge under den återstående tiden för att minska strömmen, är det mer energieffektivt jämfört med ett Just-in-time-schema, som försöker justera toppprestanda för att förlänga körtiden med ett lägre genomsnittlig energiförbrukningsmål. Denna observation indikerar att det verkligen finns kvarstående effektoptimeringsmöjligheter för register, minnen, kombinationslogik och de lägre nivåerna av klockträdet.

Denna avhandling tillämpade en hybrid effektanalys och optimeringsflöde på DSP. Flödet kombinerar automatiserad effektoptimering med PowerPro med integrerad formell verifiering för felfria registeröverföringsnivåkoder (RTL) och ett manuellt optimeringsramverk som riktar in sig på potentiella högvärdiga manuella ändringar. Automatiserade optimeringar uppnådde en minskning med 4,49% i dynamisk effekt och en 0,84% förbättring av DCGE för DSP-blocket, med potentiella ökningar upp till 12,73% energibesparingar och 2,38% DCGE-förbättring när manuella optimeringar tillämpas.

Studien visar denna metod för ett annat block (arbiter och router) i EMCA. Detta block visade en 6,52% minskning av dynamisk effekt och en 1,28% DCGE-förbättring med PowerPro, med manuell optimeringspotential för total energibesparing på 27,7% och DCGE-förbättringar på 5,5%.

Detta hybridflöde gör att IP-block kan behandlas som svarta lådor, vilket förenklar optimeringsprocessen för ingenjörer utanför designteamet. Dessutom visade metoden minimal påverkan på yta och timing, vilket gjorde den praktisk för tillämpning i verkligheten. Detta tillvägagångssätt lägger grunden för energieffektiv ASIC-design, som tar itu med kritiska hållbarhetsutmaningar i nästa generations mobilnät.

Nyckelord

Effektanalys, Effektoptimering, Clock gating, Dynamisk effekt, Applikations-specifika integrerade kretsar (ASIC)

Acknowledgments

Upon completing this work, I would like to express my gratitude to all those who supported me throughout this journey.

First and foremost, I would like to express my sincerest gratitude to Ioannis Savvidis, my supervisor at Ericsson. From my very first day onboard to the final presentation, he provided invaluable guidance at every step of this study. He always gave me patient guidance, whether it was addressing high-level research strategies or delving into specific metrics and details. His profound expertise and previous work in this field helped me overcome numerous challenges and even inspired my future career.

I would like to thank Anders Engqvist, my line manager at Ericsson. He provided me with the resources and permissions I needed to do this work and introduced me to his team so that I could better carry out this work.

I also owe a special debt of gratitude to the design team, Jakob Brundin and Marcus Nordh. Their extensive experience and knowledge of the DSP IP have provided a lot of inspiration and help for this work. Marcus also helped me with many detailed technical issues in the early stages of this work.

My thanks also go to several other colleagues at Ericsson who supported me during various stages of this study: Zilin Zhang, Anjana Ramesh Menon, Eric Roller, Henrick Steffansson, Joao Altermann, Jithin Sasidharan, and Govind Sreekumar.

I would like to express my appreciation to the team at Siemens EDA for their guidance and support with PowerPro: Richard Langridge, Himanshu Banga, and Anders Fredriksson. Additionally, I am grateful to Robin Theander at Synopsys for his valuable instructions regarding Design Compiler.

I would like to express my gratitude to my supervisor at KTH, Ritika Ratnu, for her invaluable guidance and support throughout my thesis work. I would also like to extend my heartfelt thanks to my examiner, professor Ahmed Hemani, make this work available for publication in the future.

Finally, my deepest gratitude goes to my parents. I would not have made it this far without their unwavering support and love.

Completing this thesis marks a significant milestone in my academic career and brings my master’s degree journey to a perfect end. No matter where the future takes me, I will always remember where I started and remain deeply grateful to all those who helped me along the way.

Stockholm, March 2025
Xinyu Xue

Contents

1 Introduction **1**
1.1 Background 1
1.2 Problem 3
1.3 Purpose 3
1.4 Goals 3
1.5 Delimitations 4
1.6 Structure of the thesis 4

2 Background **5**
2.1 Power dissipation in ASICs 5
2.1.1 Static power 5
2.1.2 Static power optimization 6
2.1.3 Dynamic power 7
2.1.3.1 Internal power 7
2.1.3.2 Switching power 8
2.1.4 Dynamic power optimization 8
2.2 Clock gating 9
2.2.1 Hierarchical clock gating 10
2.2.2 Local clock gating 10
2.2.2.1 Combinational clock gating 11
2.2.2.2 Sequential clock gating 11
2.2.3 Clock gating threshold 14
2.3 Differential energy analysis 15
2.3.1 Theory 15
2.3.2 Analysis on the DSP block 16
2.4 Dynamic power metrics 18
2.4.1 Switching activity 18
2.4.2 Static clock gating efficiency (SCGE) 19
2.4.3 Dynamic clock gating efficiency (DCGE) 20

2.4.4 Register output activity density for flip-flops (ROADF) 21
2.4.5 Register output activity density for enables (ROADE) . 22

3 Methods **23**
3.1 Power analysis and optimization tools 23
3.1.1 ActivityExplorer 23
3.1.2 VCD2TB 24
3.1.3 Spyglass Power 25
3.1.4 PowerPro and SLEC-Pro 25
3.1.5 PrimeTimePX 26
3.2 Building power and power metrics curves 26
3.3 Power optimization flow 30
3.4 Replay gate-level simulation 33
3.5 Post-optimization verification 33
3.5.1 Equivalence check 33
3.5.2 Sanity check 35
3.6 Post-optimization analysis 36
3.6.1 RTL analysis 38
3.6.1.1 Code comparison before and after PowerPro optimization 39
3.6.1.2 Analysis of optimization effects at the hierarchical instance level 40
3.6.1.3 Analysis of optimization effects at the register level 41
3.6.2 Gate-level analysis 42
3.6.3 Estimation of potential optimization opportunities and effects of future manual changes 43

4 Results and Analysis **45**
4.1 Variation in characteristics of the IP blocks after power optimization 45
4.2 Optimization of Block 1 46
4.2.1 Baseline power metrics curves 46
4.2.2 Power optimization settings 48
4.2.3 RTL power analysis 50
4.2.4 Gate-level power analysis 55
4.2.5 Estimation of manual optimization 58
4.2.6 Clock gating threshold analysis 59
4.3 Optimization of Block 2 61

4.3.1 Baseline power metrics curves and optimization settings 61
4.3.2 RTL power analysis 62
4.3.3 Gate-level power analysis 63
4.3.4 Estimation of manual optimization 66

5 Conclusions and Future work 69
5.1 Conclusions 69
5.2 Future work 70

References 71

List of Figures

1.1 Breaking the energy curve [1] 2

2.1 Leakage current in a Complementary Metal Oxide Semiconductor (CMOS) inverter 6
2.2 Short-circuit current in a CMOS inverter 7
2.3 Capacitive charging and discharging in a CMOS inverter . . . 8
2.4 An example of combinational clock gating 12
2.5 An example of sequential clock gating 13
2.6 An example of recirculating register 14
2.7 Concept of differential energy analysis [20] 16
2.8 Activity graphs of Run-Fast-Then-Stop (RFTS) and Just-In-Time (JIT) scheme 17

3.1 The Graphical User Interface (GUI) of ActivityExplorer . . . 24
3.2 An example of the power curve [19] 27
3.3 Steps to build the power metrics curves 29
3.4 Power optimization flow using PowerPro 31
3.5 Post-optimization verification flow 34
3.6 Post-optimization analysis flow 37
3.7 Example of code before PowerPro optimization 39
3.8 Example of code after PowerPro optimization 40
3.9 Register matrix and “the bad corner” 43

4.1 Dynamic power curves of the baseline design of Block 1 . . . 47
4.2 DCGE and activity curves of the baseline design of Block 1 . . 49
4.3 Baseline and optimized total dynamic power curves for Block 1 at RTL level (Cropped View) 51
4.4 Total Dynamic power savings for Block 1 at RTL level 52
4.5 Baseline and optimized DCGE curves for Block 1 at RTL level 53
4.6 DCGE improvements for Block 1 at RTL level 54

4.7 Baseline and optimized total dynamic power curves for Block 1 at gate level (Cropped View) 56
4.8 Baseline and optimized DCGE curves for Block 1 at gate level 57
4.9 Impact of clock gating threshold on total dynamic power . . . 59
4.10 Impact of clock gating threshold on DCGE 60
4.11 Dynamic power curves of the baseline design of Block 2 . . . 62
4.12 DCGE and activity curves of the baseline design of Block 2 . . 63
4.13 Baseline and optimized total dynamic power curves for Block 2 at RTL level 64
4.14 Baseline and optimized DCGE curves for Block 2 at RTL level 64
4.15 Baseline and optimized total dynamic power curves for Block 2 at gate level 65
4.16 Baseline and optimized DCGE curves for Block 2 at gate level 65

List of Tables

2.1 Reproduced results of differential energy analysis on the DSP . 18

4.1 Variation in characteristics of the IP blocks 46
4.2 Fast Signal Database (FSDB) weight factors input PowerPro . 50
4.3 Total dynamic power savings for Block 1 at RTL level 52
4.4 DCGE improvements for Block 1 at RTL level 55
4.5 Total dynamic power savings for Block 1 at RTL vs. Gate Level 57
4.6 DCGE Improvements for Block 1 at RTL vs. Gate Level . . . 58
4.7 Optimization results for Block 2 at RTL vs. Gate Level 66

List of acronyms and abbreviations

ASIC	Application-Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
CSV	Comma-Separated Values
DCGE	Dynamic Clock Gating Efficiency
DSP	Digital Signal Processor
DUT	Design Under Test
EDA	Electronic Design Automation
EMCA	Ericsson Many-Core Architecture
FF	Flip-Flop
FIR	Finite Impulse Response
FSDB	Fast Signal Database
GLS	Gate Level Simulation
GUI	Graphical User Interface
ICGC	Integrated Clock Gating Cells
IP	Intellectual Property
JIT	Just-In-Time
MAC	Multiply and Accumulate
NMOS	N-type Metal Oxide Semiconductor
PMOS	P-type Metal Oxide Semiconductor
RFTS	Run-Fast-Then-Stop
ROADE	Register Output Activity Density for Enables
ROADF	Register Output Activity Density for Fip-flops
RTL	Register Transfer Level
SCGE	Static Clock Gating Efficiency

SoC	System on Chip
TCL	Tool Command Language
VCD	Value Change Dump

List of Symbols Used

The following symbols will be later used within the body of the thesis.

Δt_i	Duration of the i-th time interval, see equation (2.2)
$\text{DCGE}_{\text{manual}}$	Estimated DCGE improvement from manual optimization, see equation (3.2)
$\text{DCGE}_{\text{PowerPro}}$	DCGE improvement achieved by PowerPro, ... page 44
B_{PowerPro}	Total bit-width of all registers optimized by PowerPro, ... page 44
B_{top10}	Total bit-width of the top 10 registers and register arrays, ... page 44
E	Total energy consumed over a given period of time, see equation (2.2)
N	Total number of discrete time intervals, ... page 15
$P(t)$	Instantaneous power as a function of time, see equation (2.2)
P_i	Power consumption during the i-th time interval, see equation (2.2)
P_{manual}	Estimated total dynamic power savings from manual optimization, see equation (3.2)
P_{PowerPro}	Total dynamic power savings achieved by PowerPro, ... page 44

Chapter 1

Introduction

This chapter describes the specific problem that this thesis addresses, the context of the problem, and the goals of this thesis project, and it outlines the structure of the thesis.

1.1 Background

The rapid expansion of mobile network traffic presents significant challenges in terms of energy consumption and sustainability. While mobile networks are essential for enabling digital transformation, their operational energy demands contribute substantially to the industry's carbon footprint. The introduction of 5G networks presents both opportunities and challenges. If deployed in a manner similar to previous generations, like 3G and 4G, to meet increasing traffic demands, energy consumption could increase dramatically, with some service providers predicting doubling energy use to meet increasing traffic demands, which is unsustainable in terms of cost and environmental impact [1].

To address these issues, Ericsson has developed a holistic approach to "breaking the energy curve", which is shown in Figure 1.1, in order to reduce overall network energy consumption while meeting the growing demand for data.

When this overarching approach is specific to digital System on Chip (SoC) Application-Specific Integrated Circuits (ASICs) at the Intellectual Property (IP) level, low-power front-end design becomes a crucial focus. This includes two primary strategies. The first is hierarchical clock and data gating, which disables digital functions when they are not being used. The second

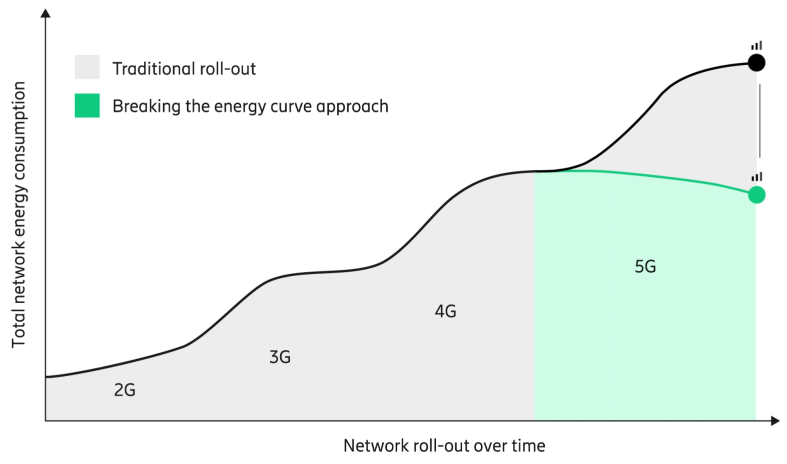


Figure 1.1: Breaking the energy curve [1]

is local clock gating within individual IP blocks and low-power features are consistently active and enabled within IP.

Ericsson Many-Core Architecture (EMCA) is an important architecture for radio signal processing with extreme data rates [2]. The Digital Signal Processor (DSP) IP block lies at the heart of the EMCA. Therefore, it should be considered an essential block for applying low-power design.

The DSP IP is highly optimized and hierarchical clock gating has been applied. However, prior differential energy analysis has shown that when DSP employs a Run-fast-then-stop scheme, which is to complete the workload with maximum performance and then enters a power saving mode for the remaining time slack to reduce power, it is more energy efficient than a Just-in-time scheme, which is trying to adjust the peak performance to prolong the run time with a lower average power consumption goal. This observation indicates that there are indeed remaining power optimization opportunities for local clock gating in DSP IP (more detailed information on differential energy analysis will be described in Section 2.3).

Thus, it is essential to perform further power optimization for DSP. Due to the large number of integrated DSP instances, any small power optimization in DSP could make a massive difference at the ASIC SoC level. This thesis project will focus on identifying and delivering optimization opportunities using a hybrid flow, including automatic power optimization with integrated formal verification guarantees for bug-free Register Transfer Level (RTL) and a manual optimization flow for more sophisticated changes.

1.2 Problem

The research questions of this thesis are:

1. How can a hybrid flow be used to characterize and optimize the DSP block, based on front-end energy performance metrics?

2. What strategies ensure optimal DSP power optimization for the best optimization effect? How much power can be saved?

1.3 Purpose

This thesis aims to perform a hybrid power analysis and optimization flow for the highly optimized DSP IP block within EMCA. This research aims to improve the energy efficiency of the DSP block by identifying and addressing residual power inefficiencies, contributing to overall power savings within EMCA. By leveraging both automated and manual optimization techniques, the study provides a systematic approach to power optimization based on front-end energy performance metrics.

This study offers practical insights into advanced power saving techniques, enabling more efficient local clock setting at the IP level. The optimization flow can be completed by anyone outside the design team, who treats design as a black box. This ensures quick and easy power optimization and is a scalable framework to optimize power at high performance ASICs.

Furthermore, the ethical considerations of this study align with global sustainability goals in terms of social and environmental benefits. Energy-efficient design reduces operational costs and mitigates the environmental footprint of mobile networks, supporting long-term ecological balance.

1.4 Goals

The primary goal of this thesis is to design and validate a hybrid power analysis and optimization flow for the DSP block within EMCA, in order to improve its power efficiency. This goal is divided into the following sub-goals:

1. **Power analysis and profiling:** Perform a comprehensive power analysis for the DSP block by extracting energy performance metrics, allowing detailed power profiling.

2. **Power optimization:** Apply a hybrid optimization flow to the DSP block, integrating automated methods based on the Electronic Design Automation (EDA) tools with manual analysis for improved power savings. The effectiveness of this optimization will be validated, ensuring the correctness of the design and quantifying the power reduction achieved.

3. **Application on additional EMCA IP blocks:** Extend the hybrid optimization flow to another critical EMCA block to further improve overall power efficiency. This step also serves to validate the scalability and effectiveness of the flow across diverse IP blocks.

1.5 Delimitations

This thesis focuses on power analysis and optimization for two specific IP blocks within the EMCA: 1) the DSP and 2) the Arbiter and Router. The optimization process follows a hybrid approach consisting of two phases.

In the first automated phase, the EDA tools were utilized to perform automatic optimizations, generating optimized RTL code. These optimized designs were verified by several EDA tools, and their effectiveness was validated.

The second phase involves the identification of registers with high potential for further manual optimization. Specific metrics were utilized to extract a list of such registers based on their contribution to power savings. However, the scope of this thesis does not include the implementation of manual optimization. Although the study provides a detailed list of the registers for manual optimization, no modifications to the RTL code were performed beyond the automated phase. Further design refinement through manual optimization will be delegated to the design team for future work.

1.6 Structure of the thesis

Chapter 2 presents relevant background information about power and power optimization in ASIC. Chapter 3 describes the tools and processes used for power analysis and optimization. It presents a hybrid flow combining automated and manual techniques. Chapter 4 presents the results collected in two case studies, in which two IP blocks are analyzed and optimized using the flow mentioned in Chapter 3. Chapter 5 concludes the thesis and discusses future work.

Chapter 2

Background

This chapter provides basic background information about power dissipation in ASIC designs. In addition, it describes key power optimization techniques such as hierarchical and local clock gating and relevant metrics such as Dynamic Clock Gating Efficiency (DCGE) that are used to evaluate their effectiveness. The chapter also discusses related work, including differential energy analysis, a method for identifying power inefficiencies by comparing energy consumption under varied job execution schemes. These concepts establish the foundation for understanding and addressing power optimization challenges in digital ASIC design.

2.1 Power dissipation in ASICs

Power consumption has become a critical concern in the design of ASIC due to the increasing demand for high-performance and low-power electronic devices. Understanding the different types of power consumption is essential for developing strategies to minimize energy usage and improve thermal performance.

Power dissipation in an ASIC can be broadly classified into two main categories: static power and dynamic power. Each category arises from different physical phenomena within the circuit components and affects the overall efficiency of ASIC [3].

2.1.1 Static power

Static power is the power consumed by a circuit when it is in steady state and does not switch. This power dissipation is primarily due to leakage currents that flow through transistors even when they are turned off, so static power is also called leakage power. The main sources of leakage currents include the following:

- **Sub-threshold leakage:** The current that flows from the drain to the source of a transistor when the transistor operates in the weak inversion region where the gate voltage is lower than the threshold voltage.
- **Gate leakage:** The current that flows directly from the gate of the transistor through the oxide to the substrate due to gate oxide tunneling and hot carrier injection.

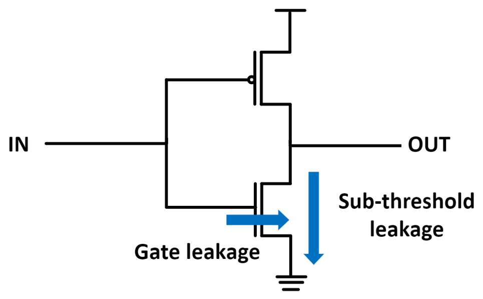


Figure 2.1: Leakage current in a Complementary Metal Oxide Semiconductor (CMOS) inverter

These two types of leakage current in CMOS are shown in figure 2.1.

2.1.2 Static power optimization

As transistor sizes shrink in advanced manufacturing processes, leakage currents become more significant due to reduced threshold voltages and thinner gate oxides [4]. To reduce leakage current and static power consumption, several techniques can be employed. An effective method is to use high-threshold voltage (high-V_{th}) transistors in non-critical paths to minimize sub-threshold leakage currents [5]. Power gating is another strategy, in which sleep transistors disconnect the power supply from idle circuits, effectively reducing leakage during standby modes [6]. Furthermore, applying reverse body bias can increase the threshold voltage during idle periods, thereby decreasing leakage currents [7].

2.1.3 Dynamic power

Dynamic power is the power consumed when transistors in the circuit are switching states, from logic high to logic low or vice versa, which means the power consumed due to applying a time-variant signal to the circuit.

Dynamic power consists of two main components:

1. **Internal (Short-Circuit) Power**
2. **Switching Power**

These components are associated with different activities during the operation of the circuit [8].

2.1.3.1 Internal power

Internal power, also known as short-circuit power, is dissipated due to short-circuit currents that flow directly from the supply voltage to the ground during the switching of transistors. Figure 2.2 shows an example of the short-circuit current in the CMOS inverter. Such short-circuit currents occur because, during a transition from logic high to logic low or vice versa, there is often a finite period in which one transistor is turning on while the other has not fully turned off, creating a direct current path [9].

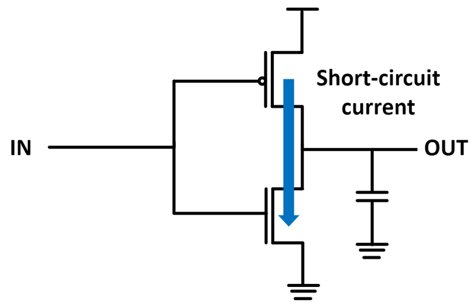


Figure 2.2: Short-circuit current in a CMOS inverter

Although internal power is typically smaller than the switching power, it can become significant at high clock frequencies or in circuits with very steep input transitions. As integration density and clock speeds increase, even a modest increase in short-circuit events per switching cycle can contribute noticeably to the total dynamic power budget. Furthermore, changing device structures and lower supply voltages in advanced technology nodes require careful management of internal power components [10].

2.1.3.2 Switching power

Switching power is the dominant component of dynamic power in digital circuits. Switching power arises from the charging and discharging of the load capacitances during signal transitions [8, 10].

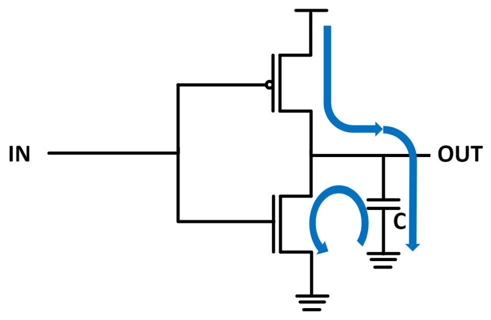


Figure 2.3: Capacitive charging and discharging in a CMOS inverter

Switching power arises from the process of charging and discharging capacitances at the nodes of a circuit during signal transitions. Figure 2.3 shows the charging and discharging of the capacitance in the CMOS inverter. When a digital signal transitions from a low logic state to a high logic state, the capacitance at the output node is charged by the current flowing through the pull-up network implemented by a P-type Metal Oxide Semiconductor (PMOS) transistor. This charging process draws energy from the power supply to store it as electrical energy within the capacitor. In contrast, when the signal transitions from a high logic state to a low logic state, the stored energy in the capacitor is discharged through the pulldown network implemented by an N-type Metal Oxide Semiconductor (NMOS) transistor and dissipated as heat in the circuit. This repetitive cycle of charging and discharging occurs at every node in the circuit that experiences signal transitions, contributing to the overall dynamic power dissipation [11].

2.1.4 Dynamic power optimization

For internal power, the optimization techniques mainly include optimizing transistor sizes, adjusting transistor threshold voltages, and controlling input slew rates by carefully inserting buffers or resizing the driver [12], etc.

For switching power, its value can be calculated using the following equation:

$$P_{\text{switching}} = \alpha \times C_L \times V_{\text{DD}}^2 \times f \tag{2.1}$$

where:

- α is the activity factor (probability of a node switching per clock cycle),
- C_L is the load capacitance,
- V_{DD} is the supply voltage,
- f is the clock frequency [10].

Reducing any of these parameters can decrease the switching power, which is crucial for low-power design considerations.

One strategy to reduce switching power is to reduce the supply voltage (V_{DD}), which has the most significant impact due to the quadratic relationship between voltage and power consumption. Methods such as dynamic voltage scaling (DVS), in which processors adjust their voltage dynamically based on performance needs, are widely used [13]. However, voltage reduction must be carefully balanced against its impact on circuit speed.

Frequency reduction is another effective technique, as it decreases the number of transitions per second. For example, clock gating reduces the effective clock frequency to zero for idle registers or blocks, thereby eliminating unnecessary toggling.

2.2 Clock gating

Clock gating is one of the most effective techniques to reduce dynamic power in ASICs. Clock gating reduces unnecessary switching activity by selectively disabling the clock signal to specific parts when they are not active, thereby reducing the switching power, which is the dominant part of the total dynamic power.

Clock gating can be categorized into hierarchical clock gating and local clock gating, each serving distinct purposes based on the granularity of the power savings. This section delves into the principles, implementation strategies, and benefits of clock gating techniques and explores the importance of parameters such as the clock gating threshold.

2.2.1 Hierarchical clock gating

Hierarchical clock gating, also known as system clock gating, is a coarse-grained power optimization technique that reduces switching activity by gating the clock for the IP blocks. This mechanism involved integrating the hier clock gating cell to gate the root clock at the top level of an IP block. The enable condition of the clock gating can typically be triggered either by software commands or by detecting idle conditions at the IP level, such as inactive interfaces or the absence of pending tasks, then the cell will stop the clock for the entire block and effectively disable all the switching activity in the block [14].

Hierarchical clock gating is highly effective for power savings because it globally disables the clock of the entire module. However, it is only applicable in scenarios where the module is in an entirely sleep state. For example, it is not suitable for scenarios such as stall conditions, where the module temporarily halts operations but may still require partial activity or data retention to maintain the system state and resume functionality seamlessly. Furthermore, hierarchical clock gating is also unsuitable for high-activity scenarios. For such operating conditions, alternative clock gating techniques, such as local clock gating, are required to complement the power-saving strategies and ensure efficient clock signal management.

2.2.2 Local clock gating

Local clock gating is another type of clock gating that reduces clock activity in the IP block.

Unlike hierarchical clock gating, which shuts off all clocks at the top level of an IP block when it enters a complete sleep state, local clock gating selectively suspends the clock for specific registers within the block while the block continues to produce outputs. As a fine-grained clock gating technique, it remains effective even when hierarchical clock gating cannot be applied, enabling the gating of internal register clocks and yielding power savings for specific register banks and logic connected to them. Local clock gating can be used in conjunction with hierarchical clock gating to achieve further power savings.

Local clock gating is typically implemented through gating conditions specified in RTL code, which is identified by synthesis tools and synthesized into Integrated Clock Gating Cells (ICGC). Therefore, correct and efficient inserting enable conditions for local clock gating in the RTL code is critical and requires significant effort. Optimizing local clock gating is also one of the key focus areas to improve the overall efficiency of the module.

Local clock gating can be categorized into two types: combinational clock gating and sequential clock gating.

2.2.2.1 Combinational clock gating

Combinational clock gating is a technique that detects when a register holds its data without changes and disables the clock signal to the register during such periods. This approach reduces the dynamic power consumed by both the register and the clock network driving it [15].

Opportunities for the insertion of combinational clock gating can be identified by analyzing conditional assignments in the RTL code. For example, clock gating logic can replace constructs like if (cond) out $\leq$ in. Power-aware logic synthesis tools are capable of recognizing such coding patterns and automatically substituting them with clock gating logic to optimize power consumption effectively.

The following Verilog code snippet is an example of inferable combinational clock gating:

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

In the provided code, the register Q loads new data only when the EN signal is activated; otherwise, it retains its previous value. If the power-aware logic synthesis tools can be used, they will identify the clock gating condition and substitute the logic to an integrated clock gating cell at the clock input of the register. The ICGC will suppress the clock toggle unless the enable signal EN is activated [16]. The example circuit is shown in figure 2.4.

2.2.2.2 Sequential clock gating

Sequential clock gating is another type of local clock gating that can be inferred from the RTL code.

Sequential analysis-based clock gating involves deriving new enable conditions to control the clock signal, effectively gating it when new data are unnecessary in downstream logic or when the data remain stable or invalid. This method is referred to as the sequential clock gating transformation [15].

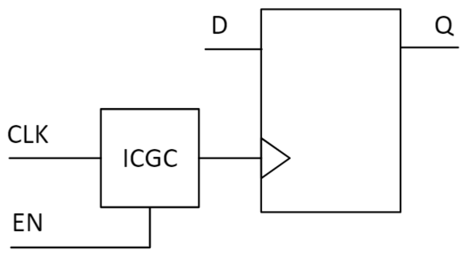


Figure 2.4: An example of combinational clock gating

An example of this transformation can be seen in the following RTL code snippet:

```
always @(posedge clk) begin
        Q0 <= D0;
        Q1 <= D1;
        EN <= SEL;
end
assign OUT = EN ? Q0 : Q1;
```

In this code, no clock gating conditions can be detected directly, and the registers Q_0 and Q_1 latch new data values every cycle. As a result, after the sequential clock gating transformation of the RTL code, when the code is processed by low-power RTL synthesis tools, these registers would not initially have clock gating implementation.

However, closer inspection reveals that if the EN signal is high, the data latched into Q_1 during the previous cycle is not used. In such cases, the previous value of Q_1 can be retained instead of being updated. Through sequential reasoning, $\sim(SEL)$ can be identified as the new enable condition for Q_1. Similarly, for Q_0, the sequential analysis identifies SEL as the new enable condition. These new conditions correspond to the updated RTL code snippet:

```
always @(posedge clk) begin
      if (SEL)
           Q0 <= D0;
      if (~SEL)
           Q1 <= D1;
```

```
        EN <= SEL;
end
assign OUT = EN ? Q0 : Q1;
```

When processed by low-power synthesis tools, the updated code ensures the insertion of clock gating logic for both Q_0 and Q_1. The example circuit is shown in figure 2.5.

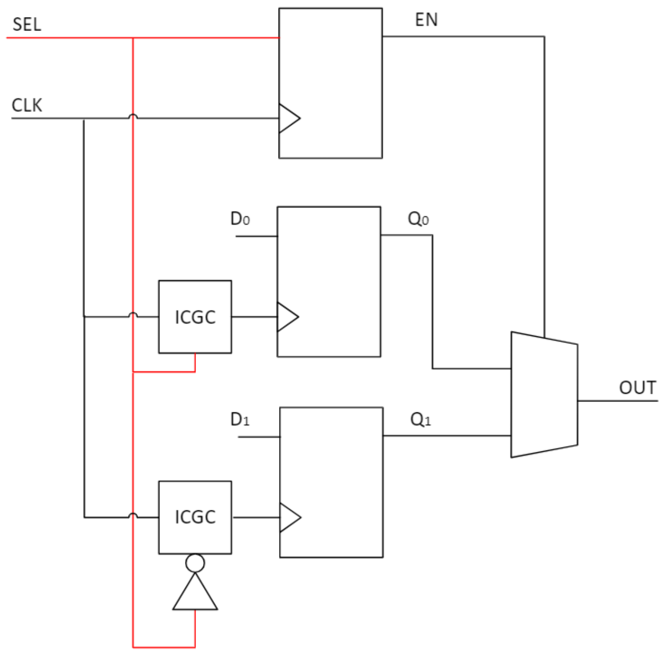


Figure 2.5: An example of sequential clock gating

System-level and sequential clock gating has a broader impact on power savings than simple combinational clock gating because they typically address global clocking inefficiencies. Sequential clock gating, in particular, is highly effective in reducing peak power consumption. Such transformations were implemented manually, requiring designers to identify and modify enable conditions.

However, this process is inherently challenging and prone to errors. Identifying opportunities for clock gating requires detailed analysis and implementing gating logic without introducing functional errors can be complex. Moreover, verifying the functional correctness of these changes is difficult, as most functional test benches may fail to provide adequate coverage for such optimizations.

2.2.3 Clock gating threshold

The clock gating threshold refers to the minimum number of bits required for a register to qualify for clock gating during synthesis. This threshold is a critical parameter used in various steps involving EDA tools, including synthesis, RTL power analysis, and RTL clock gating optimization.

For registers in the RTL code with inferable clock gating enable conditions if their bit-width is greater than or equal to the threshold, the tool generates a clock gating cell to gate the clock input of the register, which is a similar implementation as 2.4.

However, for registers with a bit-width smaller than the threshold, the tool synthesizes a feedback multiplexer at the data input instead, which is a similar implementation to 2.6.

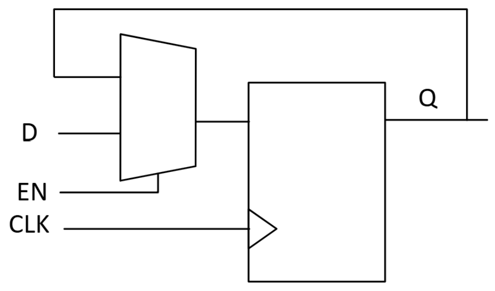


Figure 2.6: An example of recirculating register

This approach, known as a “recirculating register” implementation, uses the enable signal to either select a new data value or retain the previous value by recirculating it [17]. Importantly, no clock gating cell is added to the clock input in this type of implementation, resulting in clock activity remaining unaffected and failing to reduce unnecessary toggling. Moreover, the redundant feedback multiplexer introduced to implement the logic in the RTL code not only increases resource usage but also leads to higher power consumption. Therefore, it is important to check the synthesis reports and avoid the occurrences of the recirculating registers.

2.3 Differential energy analysis

Differential energy analysis, introduced by ANSYS and Qualcomm, is a method used to analyze and identify power optimization opportunities within a design [18]. Differential energy analysis can be a macroscopical analysis, which means identifying the power bugs at early stages without understanding the inner details of the design, just putting into the tests and profiling the blocks. In the context of this study, this approach was applied to the DSP IP block, revealing a certain inefficiency in local clock gating.

2.3.1 Theory

Differential energy analysis focuses on energy, which is a fundamental metric that represents the total power consumed over time. Mathematically, energy (E) is calculated as the integral of power (P) over time (t). The equation is given as follows:

$$E = \int P(t)\,dt \approx \sum_{i=1}^{N} P_i \cdot \Delta t_i \tag{2.2}$$

Where:

- E is the total energy consumed,
- $P(t)$ is the instantaneous power as a function of time,
- P_i is the power consumption during the i-th time interval,
- Δt_i is the duration of the i-th time interval,
- N is the total number of intervals.

The core idea of differential energy analysis is to compare the energy consumed by the design during a test case with the energy usage of a slowed version of the same test case [19]. Slowdown is introduced by adding stalls, starvation, or latencies to the test case without changing the original workload. In theory, the energy consumed by the typical and delayed test cases should remain equal for a consistent workload. This is because power consumption should decrease proportionally as the test duration increases due to added latencies.

Figure 2.7 illustrates the concept of differential energy analysis. In each plot in the figure, the blue region corresponds to the power versus time

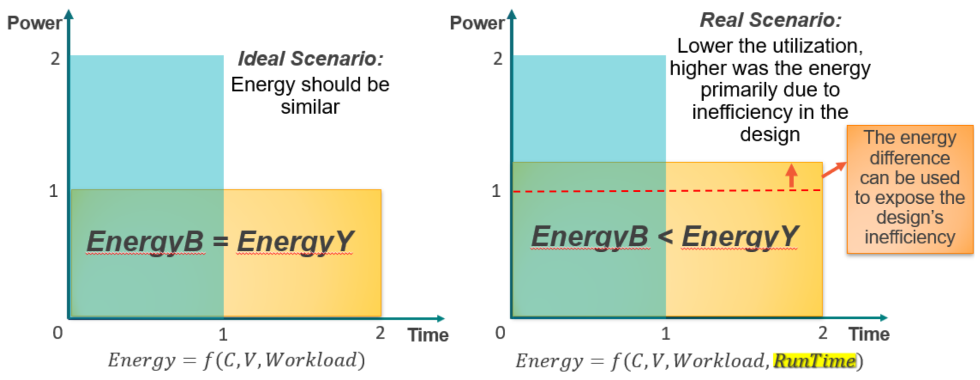

Power
2
1
0
1
2
Time
Ideal Scenario:
Energy should be similar
EnergyB = EnergyY
Energy = f(C,V,Workload)
Power
2
1
0
1
2
Time
Real Scenario:
Lower the utilization, higher was the energy primarily due to inefficiency in the design
The energy difference can be used to expose the design's inefficiency
EnergyB < EnergyY
Energy = f(C,V,Workload,RunTime)

Figure 2.7: Concept of differential energy analysis [20]

profile of a typical job, with the area under the curve representing its energy consumption. The yellow region illustrates the same job with added latencies, resulting in a longer completion duration. Since both scenarios perform the same workload, their energy consumption, corresponding to the area under the respective curves, should remain identical.

The plot on the left represents an ideal design in terms of power because the area of the blue region is equal to the area of the yellow region, which indicates that both scenarios have the same energy consumption. In contrast, the plot on the right illustrates a more realistic situation where the energy consumption of the delayed job exceeds that of the typical job. Consequently, the energy consumed becomes not only a function of the workload but also dependent on the runtime of the job, which is undesirable and exposes the inefficiency of the design.

2.3.2 Analysis on the DSP block

This study reproduced the previous work [21], which is the differential analysis of the DSP block. Two sets of assembly codes provided by the work [22] were used to implement the Finite Impulse Response (FIR) algorithm using the DSP. The core of the FIR algorithm is a set of multiply and accumulate operations, and the Multiply and Accumulate (MAC) units within the DSP are the primary resources for completing this algorithm. The assembly codes provide two working schemes for the FIR algorithm by adjusting the number of MAC units used for computing simultaneously. The schemes are as follows:

1. **Run-Fast-Then-Stop (RFTS) scheme:** 16 MAC units are used, which

allows the DSP to complete the workload with nominal performance and then go to certain sleep states for the remaining slack. The energy consumption of this scheme corresponds to the blue area in figure 2.7.

2. **Just-In-Time (JIT) scheme:** 4 MAC units are used, which adjusts peak performance to elongate runtime with lower power consumption. The energy consumption of this scheme corresponds to the yellow area in figure 2.7.

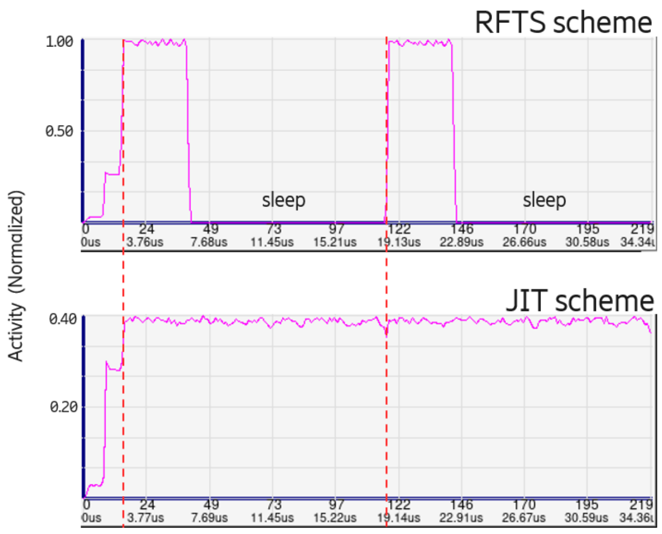


Figure 2.8: Activity graphs of RFTS and JIT scheme

The activity graphs of when DSP applies two job execution schemes are shown in figure 2.8. Two identical FIR job runs were performed under two different execution schemes. The upper section of the figure illustrates the DSP activity graph for the RFTS scheme. With the RFTS scheme, the DSP operates at higher performance levels before going into a sleep state, repeating a high activity-to-sleep cycle twice. The duration of sleep is adjusted so that the total time for one active and sleep cycle matches the time required for the execution of a single job under the JIT scheme. In contrast, with the JIT scheme, the DSP operates at a lower performance level continuously until completing a single run, repeating a moderate-activity cycle twice.

With the RFTS scheme, sleep periods activate hierarchical clock gating, effectively suppressing clock toggles within DSP during these intervals. In contrast, in the JIT scheme, the DSP remains continuously active, causing the hierarchical clock gating to be inapplicable. Only local clock gating is active in this case. Therefore, the differential energy analysis of these two job execution schemes mainly evaluates whether local clock gating in DSP can achieve a similar effect of clock toggle suppression and corresponding power reduction as hierarchical clock gating.

Table 2.1: Reproduced results of differential energy analysis on the DSP

Metric	JIT scheme	RFTS scheme	Ratio
Unitless Energy	538813	277968	1.94
Total Power Normalized	100%	52.09%	1.92
Dynamic Power Normalized	100%	51.59%	1.94

Table 2.1 presents the results of the reproduction of the differential energy analysis performed on the DSP. The results are comparable to the previous results in [21]. The data in the table indicate that the unitless energy consumed by DSP with the JIT scheme is approximately 1.9 times that of the RFTS scheme. This corresponds to the area of the yellow section in Figure 2.7 being approximately 1.9 times larger than that of the blue section. Ideally, this ratio should be close to 1. The observed discrepancy arises because, with the JIT scheme, local clock gating of the DSP cannot achieve effects comparable to hierarchical clock gating. As a result, not all redundant clock toggles are suppressed with the JIT scheme. This finding highlights the potential for further optimization of the local clock gating in the DSP.

2.4 Dynamic power metrics

Beyond analyzing dynamic power alone, a more detailed investigation of power inefficiencies, including potential power bugs, was carried out at the register and block levels. This involved utilizing various power metrics. This section introduces the definitions of several key dynamic power metrics that are fundamental in analyzing and improving the power efficiency of digital designs.

2.4.1 Switching activity

Switching activity refers to the frequency of signal toggles relative to the clock cycle, representing the dynamic behavior of signals in a design. It is a crucial characterization metric used in power analysis, particularly at the RTL level, where it provides a reasonably accurate estimate of signal behavior during simulation.

Activity is typically measured as the number of toggles on a signal per clock cycle and can be expressed as a percentage. For example, A signal that toggles on every clock edge (both rising and falling) has an activity of 200%, such as a continuously running clock. A signal that changes its value once per clock cycle is defined as having an activity of 100%.

Captured activity data can be categorized into the following:

- **Total Average Activity:** Reflects the overall toggle behavior across the design, combining all signals, including clocks, combinational logic, and registers.
- **Average Combinational Activity:** Focuses on the toggle rate of combinational logic, providing insight into the activity levels of purely logic-driven paths.
- **Average Register Activity:** Measures the toggle rate of data registers that are directly influenced by clock and data transitions.

Activity is crucial for estimating dynamic power consumption, as toggles directly correlate with dynamic power dissipation. The collected activity data, visualized as average activity over time using tools such as ActivityExplorer and Spyglass Power, allows designers to assess the activity levels of a block at different times. This helps infer power consumption patterns and select time windows that represent varying power levels for detailed power analysis.

In addition, average activity can be used as a preliminary indicator of potential power problems for specific operating points. For example, at idle or stall operating points where no tasks are being executed, or data updates occur, the activity value should ideally be close to zero. If it is found to be significantly higher, it suggests the possibility of redundant activity caused by insufficient clock gating. This could lead to unnecessary dynamic power consumption, highlighting the need for further investigation and potential optimization.

2.4.2 Static clock gating efficiency (SCGE)

Static Clock Gating Efficiency (SCGE), also known as clock gating ratio, is a metric that quantifies the proportion of clocked elements within a design that are gated with clock gating logic. It is typically expressed as the percentage of registers with clock gating enabled out of the total number of registers in the design. Mathematically, it can be represented as:

$$SCGE = \frac{\textit{Number of clock gated registers}}{\textit{Number of total registers}} \tag{2.3}$$

SCGE can be observed in the reports generated from RTL power analysis and the synthesis report for the pre-layout netlist. A low SCGE in these reports often indicates that the design lacks sufficient clock gating and may have significant power inefficiencies. Blocks with minimal or no clock gating should be improved to reduce power consumption.

However, a high SCGE does not necessarily guarantee that the design employs effective clock gating. In some cases, clock gating cells may fail to suppress clock activity effectively when the associated registers have no data activity. As a result, a high SCGE might not reflect underlying inefficiencies in such clock gating implementation. Therefore, evaluating the efficiency of clock gating within a block requires the consideration of additional metrics along with SCGE to provide a comprehensive assessment.

2.4.3 Dynamic clock gating efficiency (DCGE)

DCGE is a metric that evaluates the effectiveness of clock gating in suppressing unnecessary clock toggling in real time during circuit operation. Unlike static clock gating efficiency, which measures the proportion of gated registers in a design, DCGE is an activity-dependent metric. It focuses on the actual runtime behavior of the clock network and registers under specific workloads or test scenarios.

DCGE is typically expressed as the ratio of the number of clock toggles suppressed by clock gating to the total number of clock toggles that would occur without clock gating. Mathematically, it can be defined as follows:

$$\begin{aligned} DCGE &= \frac{\textit{Number of gated clock toggles}}{\textit{Number of total clock toggles}} \\ &= 1 - \frac{\textit{Number of active clock toggles after clock gating}}{\textit{Number of total clock toggles}} \end{aligned} \tag{2.4}$$

Since DCGE measures the efficiency based on the actual clock activity, it provides a more accurate representation of the effectiveness of clock gating compared to the structurally focused SCGE. This is particularly evident in idle or stall states, where no data updates should occur within the block. In such scenarios, DCGE serves as a reliable metric to evaluate the clock gating performance, with a healthy DCGE value close to 100%. This indicates that the majority of register clock toggles are successfully suppressed by clock gating. Otherwise, a low DCGE at the stall and idle operating points indicates power bugs in the design.

However, in high-activity scenarios, DCGE may not accurately reflect the true efficiency of clock gating. As more registers need to toggle, achieving a DCGE value close to 100% becomes unrealistic and unnecessary. In these cases, additional metrics should be used to evaluate the effectiveness of clock gating under high-activity conditions.

That said, even in high-activity operating points, if design optimizations are applied that do not alter the functionality of the block but result in a relative increase in DCGE, this improvement can still be considered an enhancement in clock gating. This is because more clock toggles are being suppressed without affecting the functionality of the design, demonstrating improved power efficiency.

2.4.4 Register output activity density for flip-flops (ROADF)

Register Output Activity Density for Fip-flops (ROADF) is a metric that quantifies the activity level of the flip-flop by measuring the ratio of its data output toggles to the total number of clock cycles. ROADF is mathematically defined as:

$$ROADF = \frac{Number\ of\ toggles\ on\ the\ data\ pin\ of\ the\ flip-flop}{Number\ of\ toggles\ on\ the\ clock\ pin\ of\ the\ flip-flop} \tag{2.5}$$

ROADF is an effective metric for evaluating the quality of clock gating for a specific register. According to its definition, a low ROADF value indicates that the number of toggles on the data pin is significantly lower than that on the clock pin. This suggests that many clock pin toggles are redundant. In contrast, a ROADF value approaching 100% implies that almost no clock pin toggles are unnecessary, reflecting efficient clock gating.

ROADF can be analyzed using EDA tools such as Spyglass Power, typically inspected alongside DCGE in the register view. When a register exhibits both low ROADF and low DCGE, it indicates inefficient clock gating, resulting in excessive clock toggles. This information serves as a valuable reference for designers, helping them identify specific registers that should be targeted for optimization to reduce redundant clock activity and improve power efficiency.

2.4.5 Register output activity density for enables (ROADE)

Register Output Activity Density for Enables (ROADE) is an extension of ROADF, evaluated using the same criterion, but is a metric for all registers that share the same clock enable signal. ROADE is mathematically defined as:

$$ROADE = \frac{\textit{Number of toggles on the data pin (s)}}{\textit{Number of toggles on the clock pin}} \tag{2.6}$$

When multiple registers share the same enable signal, ROADE aggregates the toggles of all data pins associated with these registers. However, if all registers toggle simultaneously, it is counted as a single toggle to prevent overestimating activity. The denominator accounts for clock pin toggles rather than the root clock, ensuring that the effect of clock gating enable conditions is properly reflected.

ROADE is a critical metric used to assess the effectiveness of clock gating, particularly in scenarios where the design operates under non-idle test cases. Unlike idle conditions, where a low DCGE often signifies inefficient clock gating at active operating points, a low DCGE may simply reflect the need for frequent data transitions that prevent clock suppression. In such cases, ROADE provides a more comprehensive measure of clock gating performance by taking into account the relationship between data transitions and clock activity in the design [23]. A high ROADE value indicates effective clock gating, where clock edges are predominantly used for meaningful data transitions, while a low ROADE suggests redundant clock toggles that do not correspond to data output activity. This metric is particularly useful for identifying areas where clock toggling might be optimized to reduce unnecessary power consumption.

ROADE is a valuable power profiling metric generated by tools such as SpyGlass Power. The ideal value of ROADE for healthy clock gating within a design would be above 90%. By representing the quality of the enabling conditions, ROADE helps designers evaluate and refine clock gating strategies to maximize power efficiency, making it an essential metric for power-aware design optimization.

Chapter 3

Methods

A comprehensive power analysis and optimization flow is applied to eliminate the residual power bugs in the design. This flow mainly contains two phases: Phase 1 is automatic power optimization by enhancing the clock gating in the design with the aid of EDA tools; Phase 2 is the extraction of specific registers whose clock gating could still be manually improved after Phase 1. The following subsections will describe the EDA tools and the main steps in this flow.

3.1 Power analysis and optimization tools

In this study, several EDA tools were used to perform power analysis and optimization. In addition to these tools, some commonly used ASIC tools were also used. Xcelium, provided by Cadence, and Verdi, provided by Synopsys, were used to compile, simulate the design, and inspect the simulation waveform. Synopsys Design Compiler was used to synthesize the RTL codes and produce pre-layout netlists.

3.1.1 ActivityExplorer

ActivityExplorer, also known as VCD2RPT++, is an Ericsson in-house tool for activity analysis of a block [24].

The tool takes in the RTL model or the netlist of the block, with the Value Change Dump (VCD) simulation files, to perform the visualized analysis of the switching activity of the block. As shown in figure 3.1, the Graphical User Interface (GUI) of ActivityExplorer provides the time-based activity analysis at different hierarchical instances in the block and produces the visualization activity waveforms. One can quickly inspect the waveform and identify the activity states of each sub-block in the whole simulation window; hence, it is an efficient tool for activity analysis and power bugs inspection at the early stage of power analysis.

In this study, ActivityExplorer was a guide for choosing the appropriate time window based on different activity levels for further detailed power analysis.

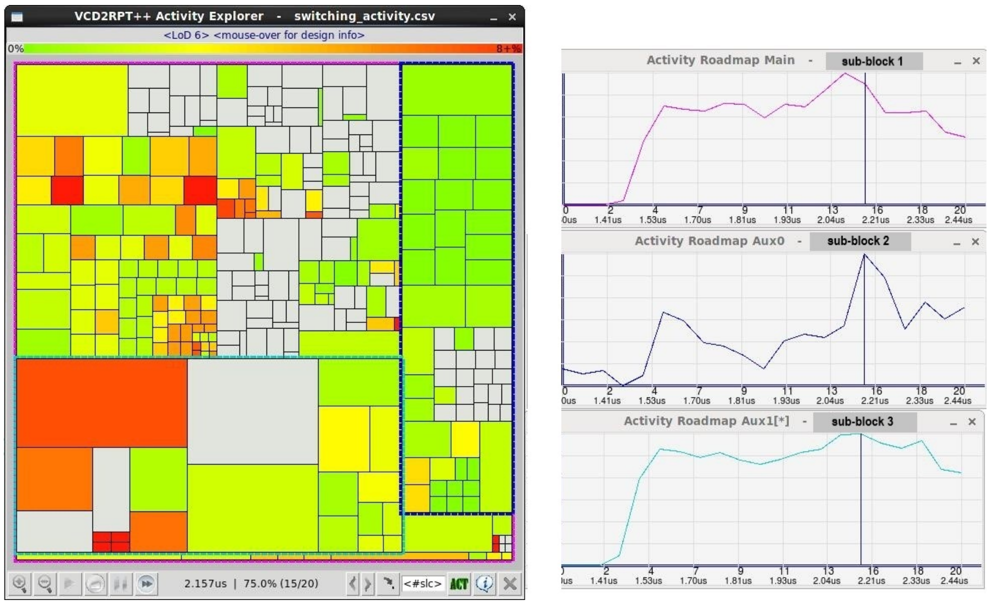

VCD2RPT++ Activity Explorer - switching_activity.csv
<LoD 6> <mouse-over for design info>
0%
8+%
2.157us | 75.0% (15/20)
<#slc>
ACT
Activity Roadmap Main - sub-block 1
Activity Roadmap Aux0 - sub-block 2
Activity Roadmap Aux1[*] - sub-block 3
0us 1.41us 1.53us 1.70us 1.81us 1.93us 2.04us 2.21us 2.33us 2.44us

Figure 3.1: The GUI of ActivityExplorer

3.1.2 VCD2TB

VCD2TB is an Ericsson in-house tool that helps bring up a power Gate Level Simulation (GLS) by converting the input stimuli VCD file into a SystemVerilog testbench [25].

A typical functional verification environment is built around RTL model simulations, so a netlist Design Under Test (DUT) cannot easily plug and play in such an environment. VCD2TB is a tool that saves this effort because it can decouple verifiers from power GLS to save resources, and no special GLS environment that brings up effort is needed.

In this study, VCD2TB was used to generate test benches for different test cases, which will be used for power GLS needed in gate-level power analysis.

3.1.3 Spyglass Power

Spyglass Power, a part of the Spyglass suite, is a power estimation tool provided by Synopsys.

Spyglass Power takes in the RTL codes of the block, the technology library, and the simulation waveform file (the Fast Signal Database (FSDB) format is being used in this study). It will output detailed power and metrics data at different hierarchical levels in the block. It can also produce the activity and metrics of every specific register, for example, DCGE and ROADF.

This tool provides a quick early-stage power analysis, as it only takes in the RTL codes rather than the real netlist. It uses the technology library to perform a pseudo-synthesis to the block, produce a pseudo-netlist, and then perform power estimation based on it. While SpyGlass Power offers valuable early-stage insights into the power consumption of the block at the RTL level, its power analysis can occasionally be less precise than that performed on a real-synthesized netlist. Thus, netlist power analysis is also involved in the process of this study to lead to more robust and accurate power estimation.

In this study, Spyglass Power was used to perform an RTL power analysis for the baseline design and all optimized designs to validate the optimization in the early stages. A detailed description of the usage of Spyglass Power is provided in Subsection 3.6.1.

3.1.4 PowerPro and SLEC-Pro

PowerPro is an EDA tool provided by Siemens. It provides a fully automated power optimization flow, focusing on improving the local clock gating efficiency of the block. SLEC-Pro is a companion tool with PowerPro and provides an equivalence check as a preliminary verification of the PowerPro-produced code.

Compared with fully manual power optimization, which requires iterative identification and optimization of power bugs that require a lot of human effort, PowerPro is able to realize fully automatic clock gating optimization and formal verification that produce bug-free RTL codes.

In this study, PowerPro was used to produce power-optimized RTL codes, where we were able to know in advance what DCGE boost is achievable based on the RTL changes introduced by the tool. SLEC-Pro was used to preliminary verify the optimized code. A more detailed description of the power optimization flow using these tools will be provided in subsection 3.3.

3.1.5 PrimeTimePX

PrimeTimePX is a Synopsys power analysis tool that performs an accurate power analysis at the gate level.

PrimeTimePX takes in the netlist of the block, technology library, constraints and parasitic files, and the gate-level simulation waveform file (the VCD format is being used in this study). It can provide precise reports for power and DCGE for all the hierarchical instances.

In this study, PrimeTimePX was used to perform a power analysis for the netlists of the baseline and optimized design after RTL power analysis, as the final validation of the optimization effects.

3.2 Building power and power metrics curves

The initial step of this study was to collect the power and power metrics at different operating points and build curves using the data collected at these points by interpolation.

The operating points mainly included the following working modes:

1. **The lowest power state:** Let the block work in sleep mode and turn on the hierarchical clock gating (if applicable to the block). This mode involves very low utilization and leads to the lowest power consumption. Power consumption and DCGE data collected in this state are the criteria to judge whether hierarchical clock gating is efficient. If the dynamic power is close to 0, and DCGE is close to 100%, it can be judged that hierarchical clock gating is effective.

2. **Low power state:** Let the block work in idle or stall mode. Turn off the hierarchical clock gating (if applicable to the block). This mode involves low utilization and leads to low power consumption. Power consumption and DCGE data collected in this state are the criteria to judge whether local clock gating is efficient. If the dynamic power is close to 0, and DCGE is close to 100%, it can be judged that local clock gating is effective. Otherwise, the local clock gating applied to the block should be improved.

3. **Intermediate power state:** Let the block perform normal functions. In this state, the power consumption of the block was at an intermediate level between the low-power states and higher-power states.

4. **High power state:** Let the block work in a high-traffic working mode. This state contained the highest activity working mode that the block could achieve, leading to the highest power consumption.

All the above types of operating points were selected from the test cases given by the design team. The types and quantities of operating points might vary depending on the module being analyzed. For example, if hierarchical clock gating is not applicable to the tested design, the lowest power state was not included. If the module was so small that the power consumption in high and low-activity states did not differ much, fewer operating points might be needed. On the contrary, if the module was so big that its peak power consumption was very large, more operating points in the intermediate power states might be needed to make the interpolation more accurate.

Figure 3.2 showed an example of a power curve. The power consumption data were collected at different operating points, arranged by the activity intensity, and plotted on the graph, then the curve was made using piecewise linear interpolation. Multiple curves were made with a similar style, using different types of power and power metrics data, such as combinational dynamic power, sequential dynamic power, DCGE, etc.

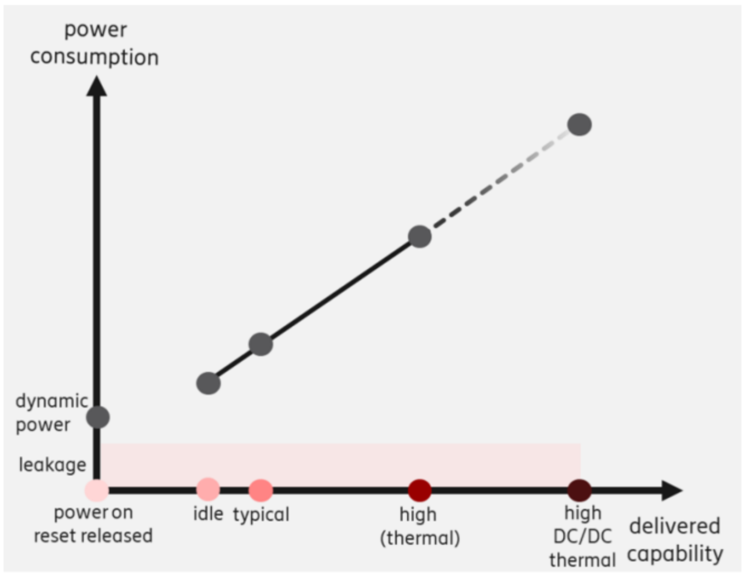


Figure 3.2: An example of the power curve [19]

The purposes of building power and power metrics curves were two-fold:

1. **Power characterization:** By building the curves, it could show the power consumption of the block under various working conditions so

that analysts and the design team could have a clear understanding of the power consumption of the block. In particular, it could be directly observed from the curves whether there are power consumption problems for some special operating points. For example, if the power consumption of the operating points of stall or idle (local clock gating was in effect) was more significant than the power consumption of the operating points of sleep (hierarchical clock gating was in effect), there was a potential optimization opportunity for the local clock gating.

2. **Inspection of the optimization effects:** After some optimization of the block and power analysis, the new curves were created and overlayed with the baseline curves in the same graph. The differences between the new and optimized curves and the baseline curves were the criteria for measuring the optimization. If the optimized curves for dynamic power were generally located below the baseline curves, optimization was considered to lead to a reduction in power consumption, which represents a positive optimization. Similarly, if the optimized curves for DCGE were generally located above the baseline curves, the optimization was regarded to have improved DCGE, which also represented a positive optimization.

The steps for performing power analysis, collecting power metrics data, and building the curves are shown in Figure 3.3.

For the initial step of power analysis, RTL simulation was performed using Xcelium and Verdi based on the test cases given by the design team. During the RTL simulation, VCD simulation waveform files were produced. For each test case, the initial RTL simulation was conducted in a full-window format, meaning the simulation time covered the entire duration of the test case. This approach was intended to allow, in subsequent analyses, the selection of a time segment within the complete test case that would represent a typical activity level for this test case.

The VCD files obtained in the previous step were used as input for the Activity Explorer. Based on the activity variation curves in the Activity Explorer GUI, which showed how activity changes over time, a shorter and more representative time window was selected for each test case to serve as the time window for power analysis.

The selected time windows for the test cases were re-simulated using the same tools to generate FSDB simulation waveform files. The resulting FSDB files will also be retained as input files for the subsequent power optimization process. The simulation waveform files, along with the RTL code of the

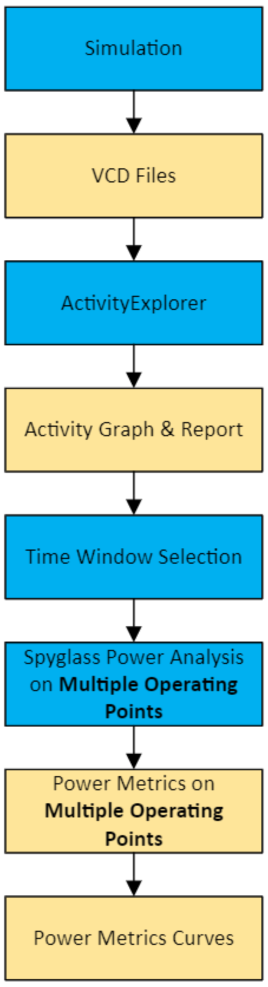

Simulation
VCD Files
ActivityExplorer
Activity Graph & Report
Time Window Selection
Spyglass Power Analysis on **Multiple Operating Points**
Power Metrics on **Multiple Operating Points**
Power Metrics Curves

Figure 3.3: Steps to build the power metrics curves

module and technology libraries, were then input into Spyglass Power for power analysis. From the final Comma-Separated Values (CSV) file output by Spyglass Power, several types of metrics were extracted: total dynamic power, combinational dynamic power, sequential dynamic power, memory dynamic power, clock dynamic power, and DCGE, etc.

Using the above process, each test case underwent simulation, activity analysis, time window selection, re-simulation, and power analysis to obtain power metrics. The power metrics data collected at multiple operating points across various test cases were then sorted in ascending order based on activity levels and plotted in the graph. Finally, piecewise interpolation was used to construct power metrics curves from these data points.

3.3 Power optimization flow

The EDA tool PowerPro was used as the power optimization tool in this study. PowerPro enables automated identification of clock-gating optimization opportunities in the RTL code and performs power optimization by modifying the original RTL code to insert clock-gating conditions [26].

The main steps in power optimization using PowerPro are shown in Figure 3.4. The main flow was performed using a Tool Command Language (TCL) script.

In the flow, the main inputs were the initial RTL codes, technology libraries, and one or a mix of the simulation waveforms in the FSDB format. The FSDB files were the ones that were used for the Spyglass Power analysis. PowerPro performed a structural-based optimization. It identified all possible functionally correct structural gating opportunities from the analysis of the RTL source code. The optimal set of conditions was selected by focusing on optimization or checking power saving by using the FSDB simulation waveforms. The weight factor was used when multiple FSDB files were available. The value specifies by which the switching activity data in the FSDB file must be scaled before annotating the switching activity data.

After the read-in of the input files, PowerPro performed the operation of prototyping the design. In prototyping, PowerPro mainly performed the following steps [27]:

1. **Consistency checks:** Ensured that the library, netlist, and constraints provided to PowerPro were consistent and were within the valid subset accepted by the tool.

2. **Normalization:** Adjusted the design to align with database invariants. Since the design was stored in the database at the operator level, certain modifications were required.

3. **Generic Optimizations:** Performed technology-independent optimizations to remove redundancy in the design.

The next step was to check the annotation rate of the FSDB files. The annotation rates reported for each FSDB file were indicators for judging whether the waveform file matched the design itself, and the value of the annotation rate for every FSDB file should be greater than 80%. Otherwise,

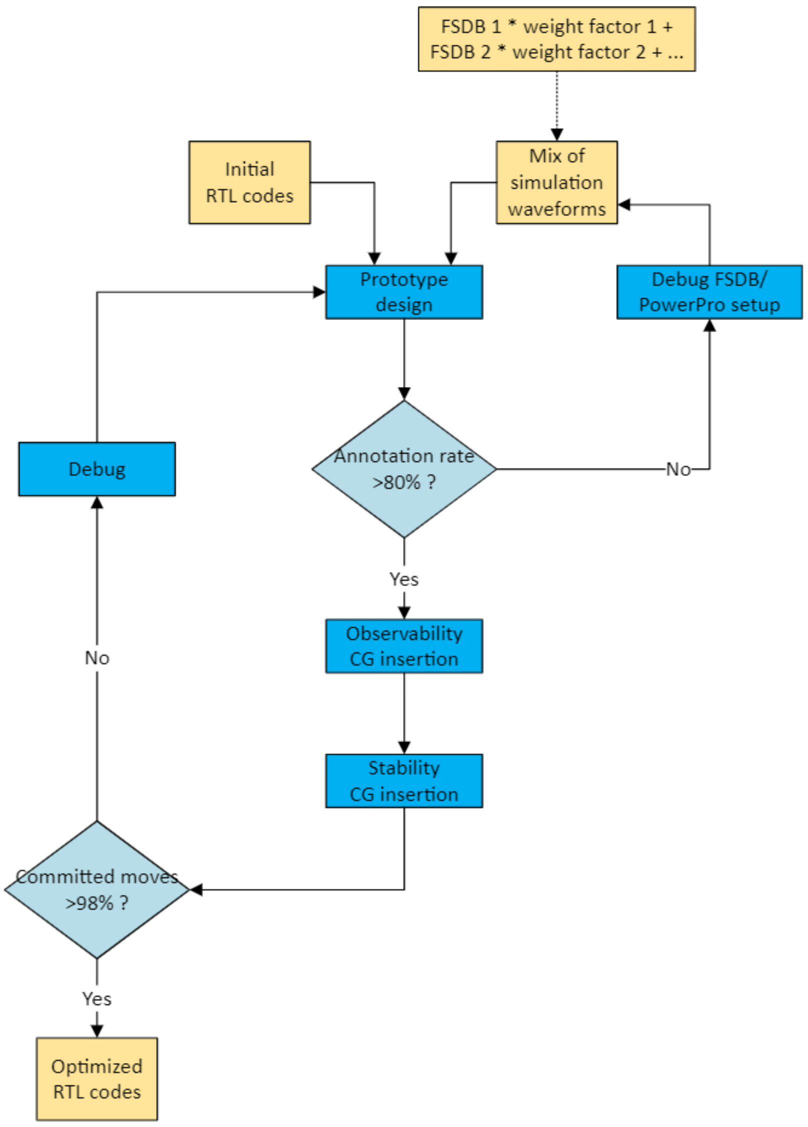


Figure 3.4: Power optimization flow using PowerPro

some debug steps were needed to ensure the correctness and effectiveness of optimization. The leading possible causes of a poor annotation rate in FSDB include the FSDB dumper that lacks read access to all regions in the design, resulting in incomplete activity information for all modules in FSDB, or a mismatch between the module path specified in the PowerPro setup and the actual path. To handle these issues requires manual debugging based on information from the report. Once the FSDB annotation rate reached 80% or higher, clock gating insertion could proceed.

The following step of clock-gating insertion was the core step of optimization. This study mainly used the following two types of clock gating insertion using PowerPro [28]:

1. **Observability-based clock gating:** Identified the unobservable writes in a data path and found the new enable condition to gate the related registers in this path.

2. **Stability-based clock gating:** This type of clock gating worked by identifying periods of inactivity in registers or flip-flops and gating the clock signal to avoid unnecessary switching during these stable or constant states. It included two subtypes:

    (a) Symbolic stability-based clock gating: Identified the stable or unchanged writes for a period at the input of a register. The enable conditions could be generated at the head of the pipelined data path to gate the related registers.

    (b) Constant stability-based clock gating: Identified the constant or unchanged writes in the flops and generated new enable condition to gate these flops.

After the automatic insertion of different types of clock gating, PowerPro automatically compiled the modified codes. During the compilation, if the tool found any moves in the patched codes to be ineffective or could not be compiled correctly, it recommitted that move and recompiled the codes until every move was clean. According to the empirical data [26], If less than 98% moves were committed, the settings of PowePro should be debugged.

Once all of the above steps were completed and cleaned, the optimized RTL codes were created. The pre-layout netlists for the baseline and optimized designs were created using the Synopsys Design Compiler. These netlists were used for further gate-level simulation and gate-level power analysis.

Although PowerPro compiled and provided an optimized power report after generating the optimized code, for more reliable results in this study, PowerPro was used only to generate the optimized code, not as a final verification of correctness and validation of the optimization effectiveness. Some third-party tools were used for verification and power analysis.

3.4 Replay gate-level simulation

The replay gate-level simulation served as an essential step in this study, preparing for subsequent sanity check and gate-level power analysis. RTL simulation was conducted using the original RTL code and all test cases, generating the input stimuli VCD for each test case. These VCD files were then fed into VCD2TB to create the SystemVerilog test bench for gate-level simulation corresponding to each test case.

Using these test benches generated from VCD2TB, the gate-level simulation could fully reuse the existing RTL simulation environment and independently compile and run with the Verilog netlist. For each test case, the gate-level simulation was performed in full window format, which means that the simulation time was the same as the initial RTL simulation and covered the entire duration of the test case.

During the gate-level simulation, power VCD files were generated for future gate-level power analysis with PrimeTimePX. The values of the gate-level reference signals were also written into text files in real-time during the simulation, comprising a selection of critical signals. These signals were chosen for their sensitivity to potential simulation issues such as improper initialization, corruption by unknown values (Xs), misaligned events, or other discrepancies. In such cases, these signals would likely exhibit incorrect values, making them effective indicators of simulation integrity. These reference signals were subsequently used for sanity checks to ensure the accuracy and reliability of the gate-level simulation, which were introduced in Section 3.5.2.

3.5 Post-optimization verification

To ensure that the optimizations brought about by PowerPro did not change the original functionality of the design, verification steps were required to guarantee that the improved RTL maintained functional integrity. Post-optimization verification was mainly divided into equivalence checks and sanity checks. The primary process in the verification flow is illustrated in 3.5.

3.5.1 Equivalence check

SLEC-Pro was a companion tool for PowerPro. In this study, it was used after the generation of the optimized codes, serving as a sequential

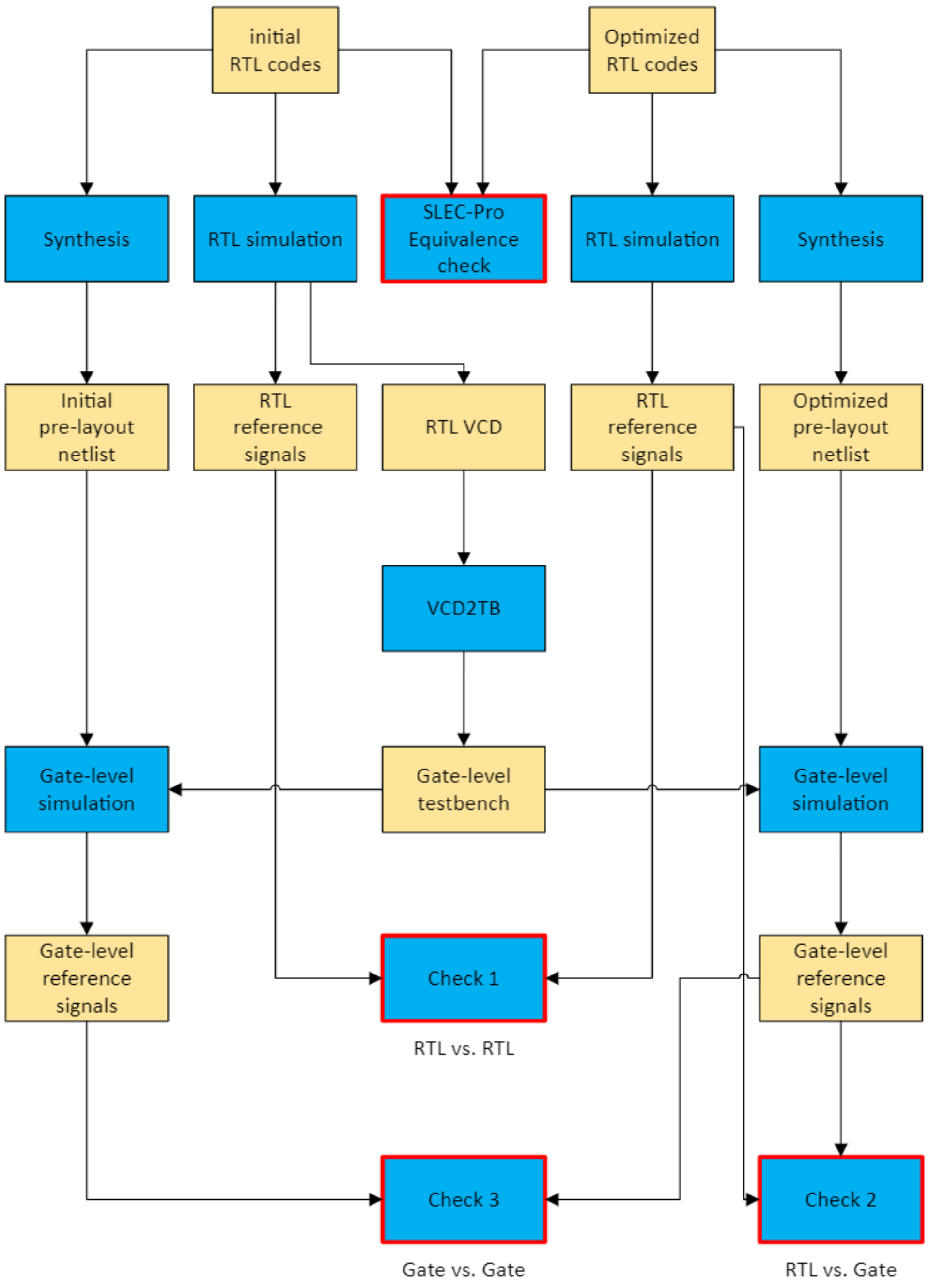


Figure 3.5: Post-optimization verification flow

logic equivalence check solution. The TCL scripts that set up SLEC-Pro were automatically generated by PowerPro during the optimization process.

By using the generated scripts, SLEC-Pro automatically performed formal verification to compare the original RTL with the low-power RTL generated by PowerPro. This approach significantly reduced the effort required to verify low-power modifications and enhanced the efficiency of the equivalence check process.

3.5.2 Sanity check

If the SLEC-Pro results were clean, further verification steps were performed. Sanity checks were performed using reference signals. Sanity checks verified the correctness and consistency of the design by comparing critical signal values under various conditions to detect any functional mismatches. For each test case, the original RTL code, the PowerPro-optimized RTL code, the original pre-layout netlist, and the optimized pre-layout netlist were simulated, and their reference signals were output to text files.

After the simulations, three comparisons of the reference signals were carried out by directly checking whether the text files containing the reference signals were identical:

1. **RTL vs. RTL:** The first comparison was between the reference signals obtained by simulating the original RTL code and those from the PowerPro-optimized RTL code. This step ensured that the functionality of the RTL code remained the same before and after optimization when running the test cases.

2. **RTL vs. Gate:** The second comparison was between the reference signals obtained from simulating the original RTL code and those from simulating the pre-layout netlist synthesized from the original RTL code. This step verified that the test benches generated by VCD2TB for gate-level simulation, as well as the original RTL simulation environment test cases, provided the same stimulus for the design. Furthermore, it confirmed that the synthesis flow produced a netlist functionally equivalent to the RTL code.

3. **Gate vs. Gate:** The third comparison was between the reference signals obtained by simulating the netlist synthesized from the original RTL code and those from simulating the netlist synthesized from the PowerPro-optimized RTL code. This step validated that the netlist synthesized from the PowerPro-optimized code retained the same functionality as the original netlist.

If all three checks passed, it was concluded that the PowerPro-optimized code and the synthesized netlist maintained the same functionality as the original design. Every set of code generated by PowerPro required the previous verification process to ensure the correctness of the results. Subsequently, the verified optimized RTL code and netlist were used for further post-optimization analysis.

3.6 Post-optimization analysis

After verifying the code, it was essential to validate the improvement achieved through optimization. Since PowerPro was solely utilized to generate the optimized RTL codes, separate engines were used for the optimization and analysis processes to ensure an unbiased evaluation.

The post-optimization analysis flow in this study is shown in figure 3.6. A comprehensive power analysis was performed in three stages to ensure robust and reliable results. This multi-stage approach not only enhanced the accuracy of power metrics but also provided detailed insights into the effectiveness of the optimizations at different levels and aspects of the design.

In this study, post-optimization analysis was structured into three key phases:

1. **RTL analysis:** Spyglass Power was used to compare the changes of the RTL codes and examine power optimizations at both the hierarchy instance and the register levels. During this phase, Python scripts were used to automate the comparison of optimization results, allowing for a thorough and efficient evaluation of how PowerPro optimizations affected specific design components.

2. **Gate-level analysis:** PrimeTimePX was used for gate-level power analysis to achieve highly accurate and definitive power metrics. This step served as the final confirmation stage, ensuring that the results were precise and trustworthy. The gate-level analysis provided a comprehensive view of power consumption under realistic operating conditions, solidifying confidence in the reported power savings.

3. **Estimation of potential optimization opportunities and effects of future manual changes:** Python scripts were developed to identify registers with significant manual optimization potential. A list of these registers was generated and their total bit-width was calculated. The ratio of this bit-width to that of registers already optimized by PowerPro

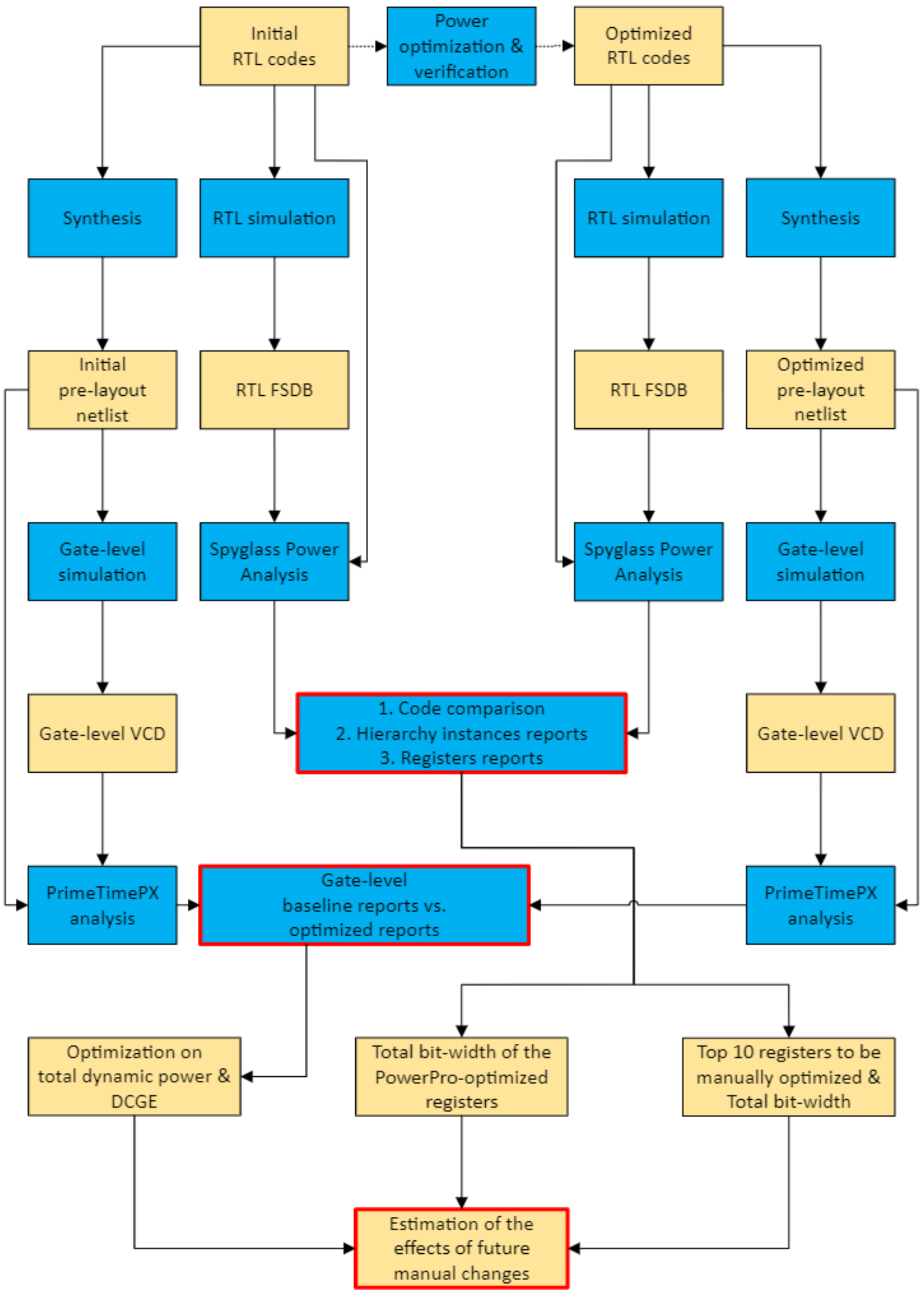

Initial RTL codes
Power optimization & verification
Optimized RTL codes
Synthesis
RTL simulation
RTL simulation
Synthesis
Initial pre-layout netlist
RTL FSDB
RTL FSDB
Optimized pre-layout netlist
Gate-level simulation
Spyglass Power Analysis
Spyglass Power Analysis
Gate-level simulation
Gate-level VCD
1. Code comparison
2. Hierarchy instances reports
3. Registers reports
Gate-level VCD
PrimeTimePX analysis
Gate-level baseline reports vs. optimized reports
PrimeTimePX analysis
Optimization on total dynamic power & DCGE
Total bit-width of the PowerPro-optimized registers
Top 10 registers to be manually optimized & Total bit-width
Estimation of the effects of future manual changes

Figure 3.6: Post-optimization analysis flow

was determined. This ratio and the power savings achieved by PowerPro were used to estimate the potential impact of future manual adjustments. This phase offered a systematic framework for prioritizing manual optimization efforts based on their likely contribution to overall power reduction.

In summary, the first two phases confirmed the optimizations achieved automatically by PowerPro, while the third phase estimated the potential improvements that could be achieved through manual modifications on top of these optimizations. This made it possible to establish a hybrid optimization flow that combines both automatic and manual approaches.

3.6.1 RTL analysis

RTL power analysis was mainly performed using Spyglass Power. Both the original RTL code and the PowerPro-optimized and verified RTL code were simulated at all operating points of the test cases to generate FSDB-format RTL simulation waveforms. These waveforms, along with the respective RTL codes, were then input into Spyglass Power for power analysis.

Spyglass Power provided three main types of information:

1. The GUI of Spyglass Power enabled a comparison of the code before and after optimization, enabling an analysis of the modifications made by PowerPro.

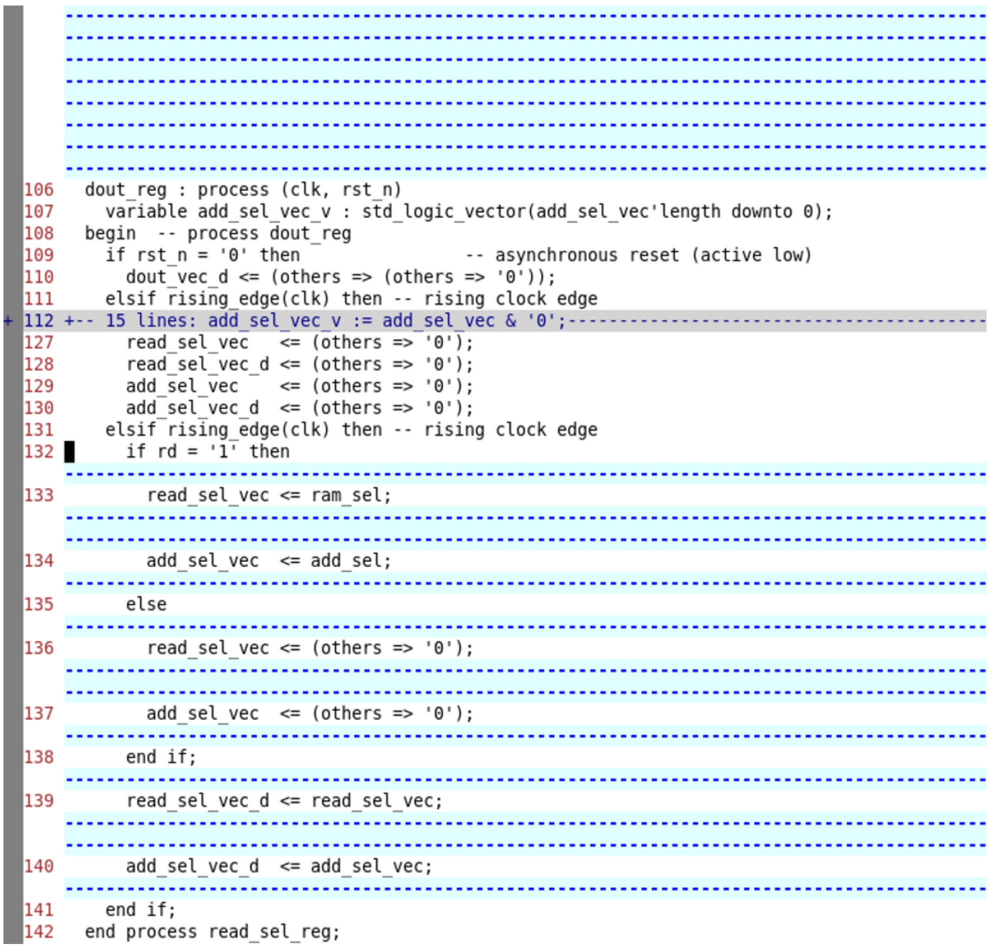
2. After completion of all steps, PowerPro generated a CSV file containing detailed power metrics for all hierarchical instances.

3. The GUI of Spyglass Power included a register view, where power metrics for each register could be examined. These register-level metrics could also be exported as CSV files for further analysis.

To achieve a fast and effective analysis of the optimization effects using the above information, several Python scripts were developed to support the Spyglass Power analysis. These scripts were specifically designed to analyze and compare CSV files exported from Spyglass Power, which contain detailed power consumption data at the register and module levels. By automating the extraction and comparison of power metrics, the scripts enabled a more efficient evaluation of power usage patterns within the design hierarchy.

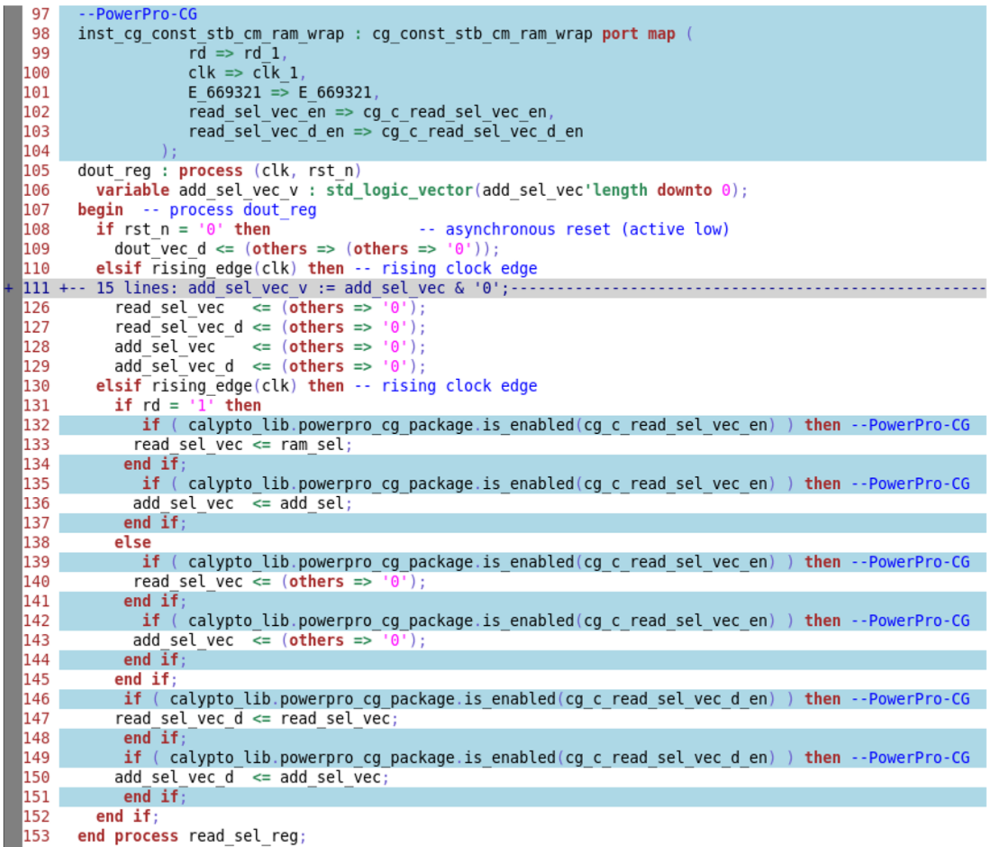
3.6.1.1 Code comparison before and after PowerPro optimization

Compared with the original RTL code and the optimized RTL code side by side in the Spyglass Power GUI, the changes made by PowerPro to the RTL code could be clearly observed and analyzed. Figures3.7 and 3.8 present examples of code segments before and after PowerPro optimization, respectively. From the additional content in Figure 3.8 compared to Figure 3.7, it was observed that lines 97 to 104 in Figure 3.8 introduced clock-gating enable logic generated by PowerPro, encapsulated in a module. Furthermore, lines 132 to 151 in Figure 3.8 showed the injection of the clock-gating condition created by PowerPro into the clocked processes.



```
106  dout_reg : process (clk, rst_n)
107    variable add_sel_vec_v : std_logic_vector(add_sel_vec'length downto 0);
108  begin  -- process dout_reg
109    if rst_n = '0' then                     -- asynchronous reset (active low)
110      dout_vec_d <= (others => (others => '0'));
111    elsif rising_edge(clk) then -- rising clock edge
+ 112 +-- 15 lines: add_sel_vec_v := add_sel_vec & '0';---------------------------------------
127      read_sel_vec   <= (others => '0');
128      read_sel_vec_d <= (others => '0');
129      add_sel_vec    <= (others => '0');
130      add_sel_vec_d  <= (others => '0');
131    elsif rising_edge(clk) then -- rising clock edge
132      if rd = '1' then
133        read_sel_vec <= ram_sel;
134        add_sel_vec  <= add_sel;
135      else
136        read_sel_vec <= (others => '0');
137        add_sel_vec  <= (others => '0');
138      end if;
139      read_sel_vec_d <= read_sel_vec;
140      add_sel_vec_d  <= add_sel_vec;
141    end if;
142  end process read_sel_reg;
```

Figure 3.7: Example of code before PowerPro optimization



```
--PowerPro-CG
inst_cg_const_stb_cm_ram_wrap : cg_const_stb_cm_ram_wrap port map (
            rd => rd_1,
            clk => clk_1,
            E_669321 => E_669321,
            read_sel_vec_en => cg_c_read_sel_vec_en,
            read_sel_vec_d_en => cg_c_read_sel_vec_d_en
         );
dout_reg : process (clk, rst_n)
  variable add_sel_vec_v : std_logic_vector(add_sel_vec'length downto 0);
begin  -- process dout_reg
  if rst_n = '0' then                -- asynchronous reset (active low)
    dout_vec_d <= (others => (others => '0'));
  elsif rising_edge(clk) then -- rising clock edge
+-- 15 lines: add_sel_vec_v := add_sel_vec & '0';-----------------------------------------------
    read_sel_vec    <= (others => '0');
    read_sel_vec_d  <= (others => '0');
    add_sel_vec     <= (others => '0');
    add_sel_vec_d   <= (others => '0');
  elsif rising_edge(clk) then -- rising clock edge
    if rd = '1' then
       if ( calypto_lib.powerpro_cg_package.is_enabled(cg_c_read_sel_vec_en) ) then --PowerPro-CG
      read_sel_vec <= ram_sel;
     end if;
       if ( calypto_lib.powerpro_cg_package.is_enabled(cg_c_read_sel_vec_en) ) then --PowerPro-CG
      add_sel_vec  <= add_sel;
     end if;
    else
       if ( calypto_lib.powerpro_cg_package.is_enabled(cg_c_read_sel_vec_en) ) then --PowerPro-CG
      read_sel_vec <= (others => '0');
     end if;
       if ( calypto_lib.powerpro_cg_package.is_enabled(cg_c_read_sel_vec_en) ) then --PowerPro-CG
      add_sel_vec  <= (others => '0');
     end if;
    end if;
     if ( calypto_lib.powerpro_cg_package.is_enabled(cg_c_read_sel_vec_d_en) ) then --PowerPro-CG
    read_sel_vec_d <= read_sel_vec;
     end if;
     if ( calypto_lib.powerpro_cg_package.is_enabled(cg_c_read_sel_vec_d_en) ) then --PowerPro-CG
    add_sel_vec_d  <= add_sel_vec;
     end if;
  end if;
end process read_sel_reg;
```

Figure 3.8: Example of code after PowerPro optimization

PowerPro clearly marked each segment of added code with comments. If manual insertion of clock-gating conditions were required, the logic for generating enable signals and their source signals could be traced back to the enable logic module created by PowerPro. This shows that the automatically generated clock-gating optimization code by PowerPro was readable and could serve as a valuable reference for future manual modifications.

3.6.1.2 Analysis of optimization effects at the hierarchical instance level

After Spyglass Power completed every run of RTL power analysis, it generated a CSV file named `HierarchicalBrowser.csv`. This CSV file contained detailed data for all hierarchy instances in the design, including various types of activity, power, and power metrics. A Python script was developed to compare the `HierarchicalBrowser.csv` files before and after optimization. By matching the first column (the names of the hierarchy instances), the script identified the corresponding hierarchy instances and calculated the dynamic power savings by subtracting the total dynamic power of the optimized design from that of the original.

The script further sorted the hierarchical instances by the above total dynamic power changes and highlighted the power types with the most significant changes, displaying the results in the terminal. This tool was used to analyze the instances that contributed most significantly to the optimization, providing insights into which parts of the design were most affected by PowerPro's adjustments. This analysis is valuable for understanding the impact of optimization and identifying critical areas where further improvements could yield more significant power savings, thus enhancing overall design efficiency.

3.6.1.3 Analysis of optimization effects at the register level

In Spyglass Power GUI, there was a register view that displayed power, activity, and power metrics such as ROADF and DCGE for each specific register in the design. This register view also allowed the extraction of these data into a CSV file. A Python script was developed to compare the extracted register CSV files before and after optimization. By matching the first column (the names of the registers), the script identified the corresponding registers before and after optimization. The DCGE improvement was calculated by subtracting the original DCGE from the optimized DCGE.

The script filtered all registers with a DCGE improvement of more than 25% and sorted them by register bit-width in descending order. Then it printed these registers, along with their DCGE values before and after optimization, on the terminal. Additionally, the script calculated and printed the total bit-width of the registers optimized by PowerPro.

This script provided a list of registers where the changes from PowerPro led to significant DCGE improvements. These registers were shown to benefit from RTL code modifications to enhance DCGE. Therefore, if the PowerPro optimization results were used as a reference for manual RTL code modifications, these registers should be prioritized. The total bit-width of the PowerPro-optimized registers calculated by the script was further used in Section 3.6.3 to estimate the effects of future manual changes.

3.6.2 Gate-level analysis

After performing a RTL analysis with Spyglass Power, a gate-level analysis was still necessary to obtain a more accurate and realistic assessment of power consumption. RTL analysis provided a high-level view of the power metrics based on the estimated activity and design hierarchy, which was valuable to identify potential areas for improvement. However, it lacked the precision required to account for the actual implementation details. Gate-level analysis, performed after synthesis and using actual switching activity from gate-level simulations, offered a more precise power consumption evaluation under near-final design conditions. This step was crucial to validate the effectiveness of the optimizations and to ensure that the results were consistent and reliable when transitioning from RTL to the final physical design.

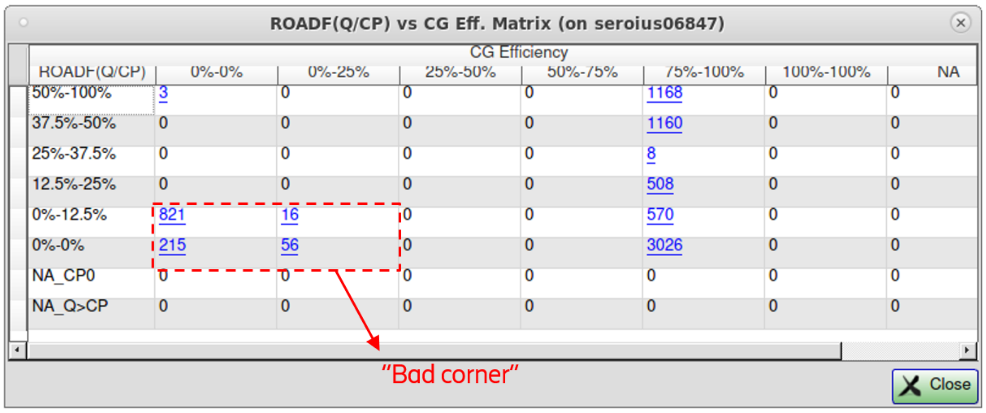
In this study, the legacy synthesis flow used in the original design was used to synthesize the RTL code to produce the pre-layout netlists. During synthesis, some of the original clock constraints were appropriately relaxed for certain parts of the design to prevent excessive negative slack, thereby avoiding adverse effects on subsequent power analysis. The synthesis report directly displayed SCGE, allowing for observation of whether the number of gated registers had increased as a result of the optimizations.

After obtaining the netlists synthesized from both the original design and the PowerPro-optimized design, gate-level simulations were performed using the SystemVerilog test benches generated following the steps in Section 3.4. The VCD waveforms obtained from these simulations were then input into PrimeTimePX for gate-level power analysis. In addition to the VCD files, the netlists, parasitics, and constraints were also provided as input to PrimeTimePX. PrimeTimePX generated results for gate-level dynamic power and DCGE using event-based dynamic power analysis.

PrimeTimePX was a tool used for accurate power estimation through detailed netlist analysis, often employed for signoff-level power analysis. By incorporating real switching activity from gate-level simulations and considering detailed parasitics, PrimeTimePX provided highly precise power metrics. This level of precision was essential for evaluating the final power characteristics of the design under realistic operating conditions and ensured that the optimization results were reliable.

3.6.3 Estimation of potential optimization opportunities and effects of future manual changes

In the register view of the Spyglass Power GUI, there was a window displaying the ROADF vs. DCGE matrix, as shown in Figure 3.9. This figure represented the total Flip-Flop (FF) count of registers that fall into each ROADF and DCGE region. The numbers in the cells indicate the FF count for registers in the corresponding region of ROADF and DCGE.


ROADF(Q/CP) vs CG Eff. Matrix (on seroius06847)

	CG Efficiency						
ROADF(Q/CP)	0%-0%	0%-25%	25%-50%	50%-75%	75%-100%	100%-100%	NA
50%-100%	3	0	0	0	1168	0	0
37.5%-50%	0	0	0	0	1160	0	0
25%-37.5%	0	0	0	0	8	0	0
12.5%-25%	0	0	0	0	508	0	0
0%-12.5%	821	16	0	0	570	0	0
0%-0%	215	56	0	0	3026	0	0
NA_CP0	0	0	0	0	0	0	0
NA_Q>CP	0	0	0	0	0	0	0

Figure 3.9: Register matrix and "the bad corner"

In this matrix, the area highlighted in the lower left corner, where ROADF was less than 12.5% and DCGE was less than 25%, was defined as "the bad corner". Registers in this region had very low ROADF and DCGE values, indicating inefficient clock gating. This inefficiency was characterized by significantly higher switching activity on the clock pins of these registers compared to the activity on their data (D) pins, leading to redundant clock toggling. Therefore, these registers should be prioritized for manual optimization in the future.

In this study, the ROADF vs. DCGE matrix obtained from analyzing the PowerPro-optimized code using Spyglass Power was exported as a CSV file. A Python script was developed to explore this CSV file, focusing on identifying the registers located in "the bad corner" (ROADF $< 12.5\%$ and DCGE $< 25\%$). The script sorted these registers by bit-width in descending order and aggregated the bit-widths of register arrays, also ranking them from high to low. Finally, the script outputs a list of the top 10 registers and register arrays with the highest bit-widths as a summary for the design team. These registers were identified as key targets for further manual optimizations on top of the PowerPro results.

The bit widths of the top 10 registers and register arrays were summed to calculate their total bit width (B_{manual}). This sum was then used to estimate the potential total dynamic power savings (P_{manual}) and DCGE improvement ($\text{DCGE}_{\text{manual}}$) that could be achieved through manual optimization of these top registers. The calculation also required the total bit-width of the PowerPro-optimized registers (B_{PowerPro}), as well as the total dynamic power savings (P_{PowerPro}) and DCGE improvement ($\text{DCGE}_{\text{PowerPro}}$) obtained from the gate-level analysis. The following formulas express these relationships:

$$P_{\text{manual}} = P_{\text{PowerPro}} \times \frac{B_{\text{manual}}}{B_{\text{PowerPro}}} \tag{3.1}$$

$$\text{DCGE}_{\text{manual}} = \text{DCGE}_{\text{PowerPro}} \times \frac{B_{\text{manual}}}{B_{\text{PowerPro}}} \tag{3.2}$$

Here:

- P_{manual}: Estimated total dynamic power savings from manual optimization.
- P_{PowerPro}: Total dynamic power savings achieved by PowerPro.
- $\text{DCGE}_{\text{manual}}$: Estimated DCGE improvement achievable through manual optimization.
- $\text{DCGE}_{\text{PowerPro}}$: DCGE improvement achieved by PowerPro.
- B_{manual}: Total bit-width of the top 10 registers and register arrays.
- B_{PowerPro}: Total bit-width of all registers optimized by PowerPro.

These formulas demonstrate that the estimated improvements are proportional to the contribution of the bit-width of the top 10 registers to the overall bit-width of PowerPro-optimized registers. Since these results are derived based on proportional estimation, they only represent the maximum potential improvements achievable through manual optimization. Further analysis by the design team is necessary to assess the practical feasibility of implementing these manual optimizations in the actual design process.

Chapter 4

Results and Analysis

In this study, two IP blocks are selected for power analysis and optimization using the flow demonstrated in the previous chapter. These IP blocks represent critical components in EMCA, each with distinct characteristics and power consumption profiles. The optimization of these IP blocks presents case studies of power optimization by improving local clock-gating efficiency, enhancing energy performance as reflected in silicon, and providing a comprehensive basis for evaluating the effectiveness of the proposed methodology.

Block 1 is the EMCA DSP, an important IP block in EMCA. It is the main focus of analysis and optimization in this study. Block 2 is the arbiter and router within EMCA, an important component in the system that works collaboratively with the DSP. It is integrated in more significant quantities within the system but is a smaller-scale design compared to the DSP IP block.

4.1 Variation in characteristics of the IP blocks after power optimization

When optimizing power, it is crucial to ensure that the process does not significantly increase the area or other critical characteristics because a larger area increases manufacturing costs and complicates integration. To validate that power optimization does not negatively impact area, the study collected and compared the data of the pre-layout netlist from the blocks before and after the automated optimization process.

Table 4.1 illustrates the variation in characteristics of the two selected IP blocks. For both Block 1 and Block 2, the number of memories remained

Table 4.1: Variation in characteristics of the IP blocks

Block name	Variation in the number of FFs	Variation in the number of memories	Variation in area
Block 1	+0.05%	N/A	-2.64%
Block 2	+0.18%	N/A	+0.92%

unchanged, and the variations in the number of flip-flops (FFs) and area were minimal and well within acceptable limits. These results demonstrate that the power optimization flow applied in this study does not negatively impact key characteristics such as area, confirming its feasibility for practical use.

4.2 Optimization of Block 1

Block 1 is the EMCA DSP, which is the main focus of the analysis and optimization in this study. Hierarchical clock gating has been applied to this block.

4.2.1 Baseline power metrics curves

Building the baseline power metrics curves is the starting point of this study to ensure power profiling and optimization validation across the full spectrum of operating points.

For building these curves for Block 1, a total of 12 test cases are used. These test cases include:

1. **Test Case 0:** Sleep state of the block. Hierarchical clock gating is activated, ensuring that the block stays in its lowest power state.

2. **Test Case 1:** Stall state of the block. Hierarchical clock gating is deactivated, and only local clock gating is working. The block remains in a low-power state.

3. **Test Case 2:** Idle state of the block. Similar to Test Case 1, the hierarchical clock gating is deactivated and only local clock gating is working. The block remains in a low-power state.

4. **Test Case 3 to 10:** Typical test cases that allow the block to work in the intermediate power state. The test cases are ordered by power consumption, arranged from lowest to highest.

5. **Test Case 11:** A special test case that leads to the maximum activity and allows the block to work in the highest power state.

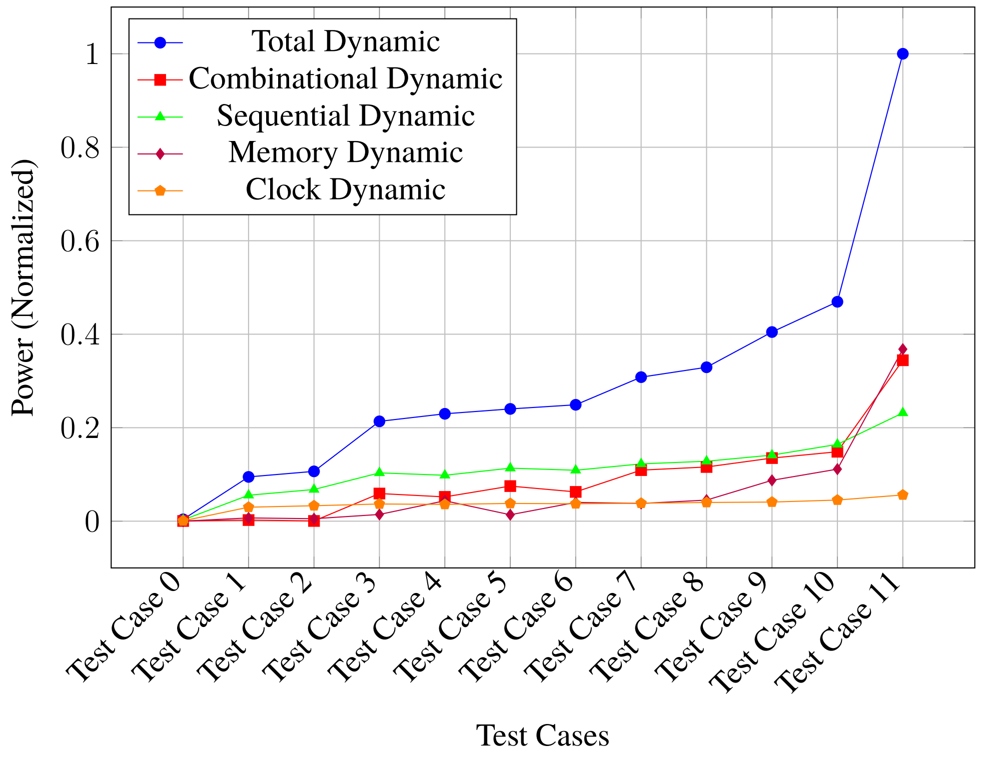


Figure 4.1: Dynamic power curves of the baseline design of Block 1

Figure 4.1 shows the normalized dynamic power curves of the baseline design of Block 1. The curves include the total dynamic power curve and its components, namely combinational dynamic power, sequential dynamic power, memory dynamic power, and clock dynamic power.

The figure illustrates that as the test case numbers increase, the total dynamic power and its components show an overall upward trend. Furthermore, operating points are distributed across power states ranging from low to high, reflecting the power consumption of the baseline design under different power states.

The figure also directly highlights some power issues present in Block 1. Under Test Case 0, hierarchical clock gating is activated. As shown in the figure, the total dynamic power and all its components are nearly zero for this test case. This indicates that the hierarchical clock gating applied to this block effectively gated the clock activity of the entire block, reducing dynamic power to a shallow level.

In contrast, Test Cases 1 and 2 correspond to the stall and idle states, respectively, where hierarchical clock gating is not activated, and only local clock gating is enabled. Ideally, since the block performs no operations in the stall and idle states, effective local clock gating should also reduce the total dynamic power and its components to near-zero levels. However, the figure shows a noticeable increase in total dynamic power and its components for these two test cases compared to Test Case 0, indicating that the dynamic power is no longer near zero.

This observation suggests that insufficient local clock gating results in additional dynamic power consumption. Therefore, optimizing local clock gating should be a priority in the subsequent work.

Figure 4.2 presents the activity and DCGE of the baseline design of Block 1 across Test Cases 0 to 12. It can be observed that as activity increases, DCGE decreases. This trend aligns with general principles since higher activity in a test case typically results in more frequent data updates across registers. Consequently, fewer clock toggles need to be gated, leading to a reduction in DCGE.

Similar to the issues identified in Figure 4.1, Figure 4.2 reveals an identical problem. For Test Case 1, where hierarchical clock gating is effective, DCGE is close to 100%. In contrast, for Test Cases 2 and 3, where only local clock gating is active, DCGE shows a noticeable decline, falling below 95%. This indicates that local clock gating is less effective than hierarchical clock gating in suppressing clock toggles. The resulting redundant clock activity prevents dynamic power from reaching its minimum level.

4.2.2 Power optimization settings

The FSDB files generated from simulating Block 1 in Test Cases 1 to 12 were input into PowerPro, which produced the optimized RTL code. Since PowerPro optimizations do not involve hierarchical clock gating, Test Case 0 was excluded from the input to PowerPro. Instead, Test Case 0 was used only to demonstrate the efficiency of hierarchical clock gating and serve as a comparison point for evaluating the efficiency of local clock gating.

PowerPro supports the use of multiple FSDB files with different weight factors to optimize a block. To investigate whether the weight factors of the

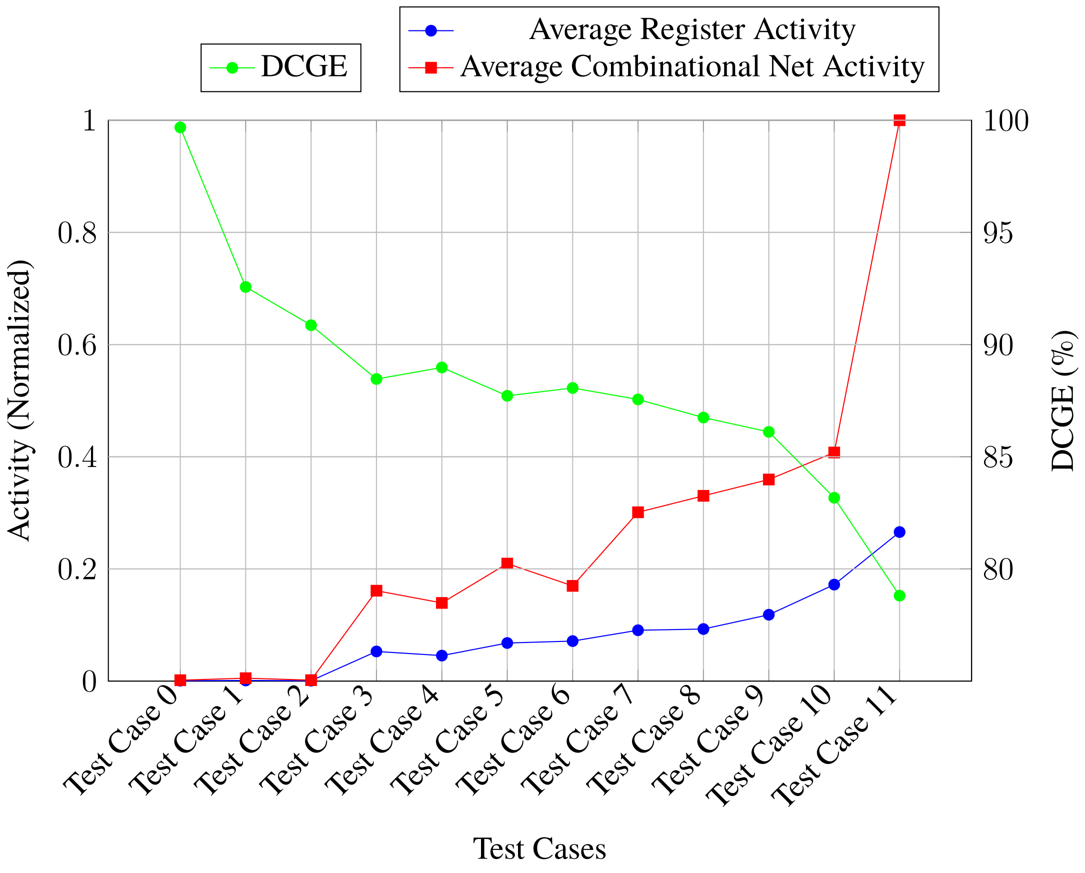


Figure 4.2: DCGE and activity curves of the baseline design of Block 1

FSDB files significantly impact the optimization results and to determine the best strategy to assign these weight factors to achieve the best optimization, this study defined three FSDB mix configurations. These configurations were based on the power characteristics of Block 1 in the baseline design in various test cases. The three configurations are as follows:

1. **Mix1:** Default mix, assign higher weights to the test cases corresponding to the typical workloads of Block 1 as identified by the design team while evenly distributing the weights among the remaining test cases.

2. **Mix2:** Idle/stall extreme mix, assign higher weights to the idle and stall test cases, which exhibit lower power consumption, while evenly distributing the weights among the other test cases. Since stall conditions are likely to occur more frequently than idle in real-world operation, Test Case 2 is given a higher weight than Test Case 3.

3. **Mix3:** Max power extreme mix, assign a significantly higher weight to Test Case 11, which represents the highest power consumption scenario, while evenly distributing the weights among the other test cases.

The specific weight assignments for each test case in these three FSDB mix configurations are detailed in Table 4.2.

Table 4.2: FSDB weight factors input PowerPro

	Mix1	Mix2	Mix3
Test Case 0	0%	0%	0%
Test Case 1	**35%**	**75%**	2%
Test Case 2	**5%**	**5%**	2%
Test Case 3	2%	2.22%	2%
Test Case 4	**30%**	2.22%	2%
Test Case 5	2%	2.22%	2%
Test Case 6	2%	2.22%	2%
Test Case 7	2%	2.22%	2%
Test Case 8	2%	2.22%	2%
Test Case 9	**15%**	2.22%	2%
Test Case 10	2%	2.22%	2%
Test Case 11	**3%**	2.22%	**80%**
Total	100%	100%	100%

The three FSDB mix configurations described above were input into PowerPro, resulting in three sets of optimized RTL code. These optimized RTL codes were then used for subsequent power analysis.

4.2.3 RTL power analysis

The three sets of optimized RTL code generated from the above three FSDB mix configurations were all successfully verified, confirming that they maintained the same functionality as the baseline design of Block 1. These optimized designs were then re-simulated under Test Cases 1 to 12, followed by an RTL power analysis using Spyglass Power. Since Test Case 0 only applies hierarchical clock gating and was not subject to optimization in this study, the power of the optimized designs was not analyzed and demonstrated in all the following steps.

The metrics collected from the three sets of PowerPro-optimized RTL code across different test cases were used to rebuild the curves, which were then

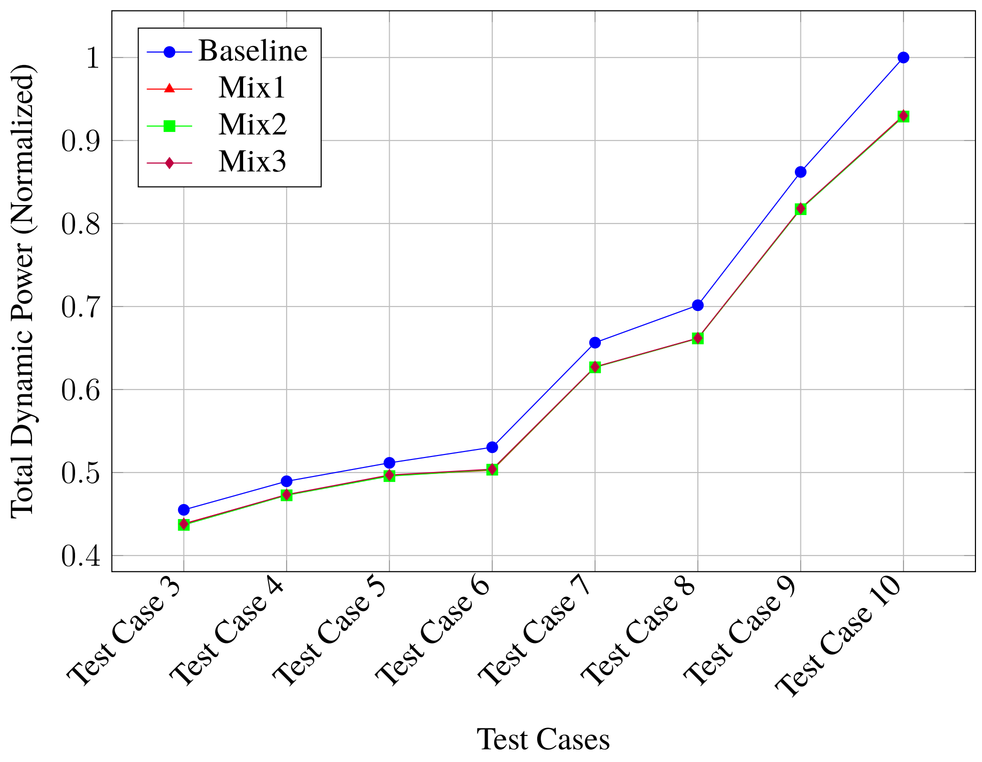


Figure 4.3: Baseline and optimized total dynamic power curves for Block 1 at RTL level (Cropped View)

overlaid on the original baseline curves. Figure 4.3 presents the baseline and optimized total dynamic power curves for Block 1 at the RTL level. To provide a clearer view of the curve changes, the figure shows a cropped view, focusing on Test Cases 3 through 10.

For all three optimized curves, the total dynamic power is consistently decreased compared to the baseline curve in these test cases, indicating that the optimized Block 1 achieved some power savings. The three optimized curves are almost identical, suggesting that different FSDB weight mix configurations did not result in significant differences in optimization effectiveness.

Figure 4.4 illustrates the percentage of total dynamic power savings for each test case. Except for the operating point with the lowest activity, all test cases show noticeable savings in total dynamic power. The slight negative saving observed in Test Case 1 is likely attributable to discrepancies or inaccuracies in the RTL power analysis, representing open items that require further investigation and confirmation through gate-level power analysis.

However, since significant savings are observed in most test cases,

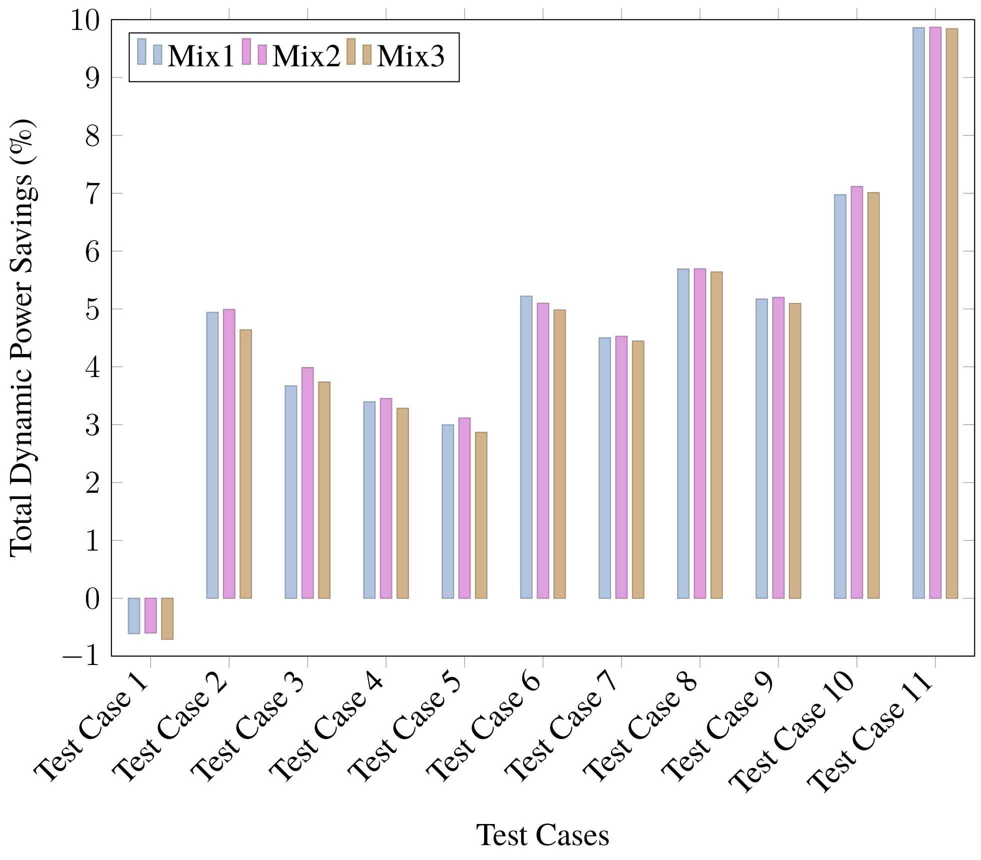


Figure 4.4: Total Dynamic power savings for Block 1 at RTL level

these changes are preliminarily considered more likely to represent actual improvements. Among the three mixes, Mix 2 (idle/stall extreme mix) demonstrates slightly better improvements than the other two mixes for most operating points.

Table 4.3: Total dynamic power savings for Block 1 at RTL level

	Mix1	Mix2	Mix3
Average total dynamic power saving	4.71%	4.77%	4.62%
Max total dynamic power saving	9.86%	9.87%	9.84%

Table 4.3 presents the average and maximum total dynamic power savings for each FSDB mix configuration. Among the three mixes, Mix 2 achieves the highest average and maximum power savings. However, the differences

compared to the other two mixes are minimal.

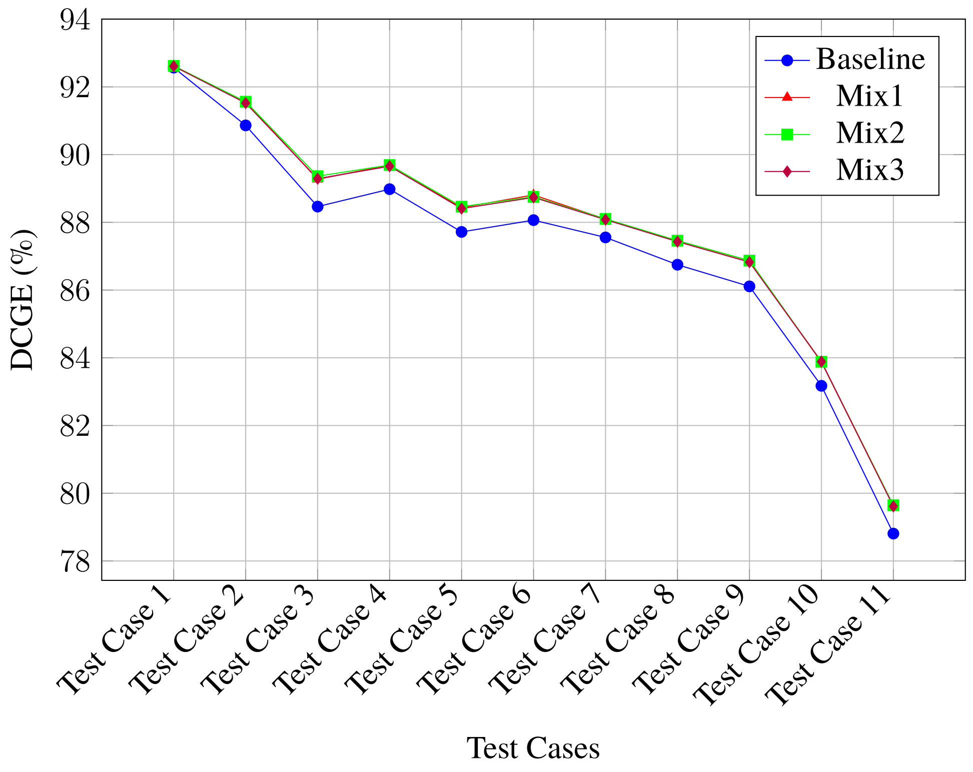


Figure 4.5: Baseline and optimized DCGE curves for Block 1 at RTL level

The observation that weights do not significantly impact the optimization effect can be attributed to the nature of the structural-based optimization by PowerPro. The primary optimization opportunities are inherently identified within the RTL code itself, while the FSDB files mainly serve to select the optimal opportunities from those already identified.

Figure 4.5 shows the baseline and optimized DCGE curves for Block 1 at the RTL level. Except for Test Case 1, the optimized design curves obviously increase, appearing above the baseline curve. This indicates that the DCGE of the optimized design has improved in most test cases.

Similar to the total dynamic power curves, the DCGE curves for the three FSDB mix configurations are nearly identical. This suggests that the weight factors assigned do not significantly influence the improvements in DCGE.

Figure 4.6 illustrates the percentage of DCGE improvement for each FSDB mix configuration. Except for Test Case 1, all other test cases demonstrate noticeable DCGE improvement. For most operating points, Mix 2 (idle/stall

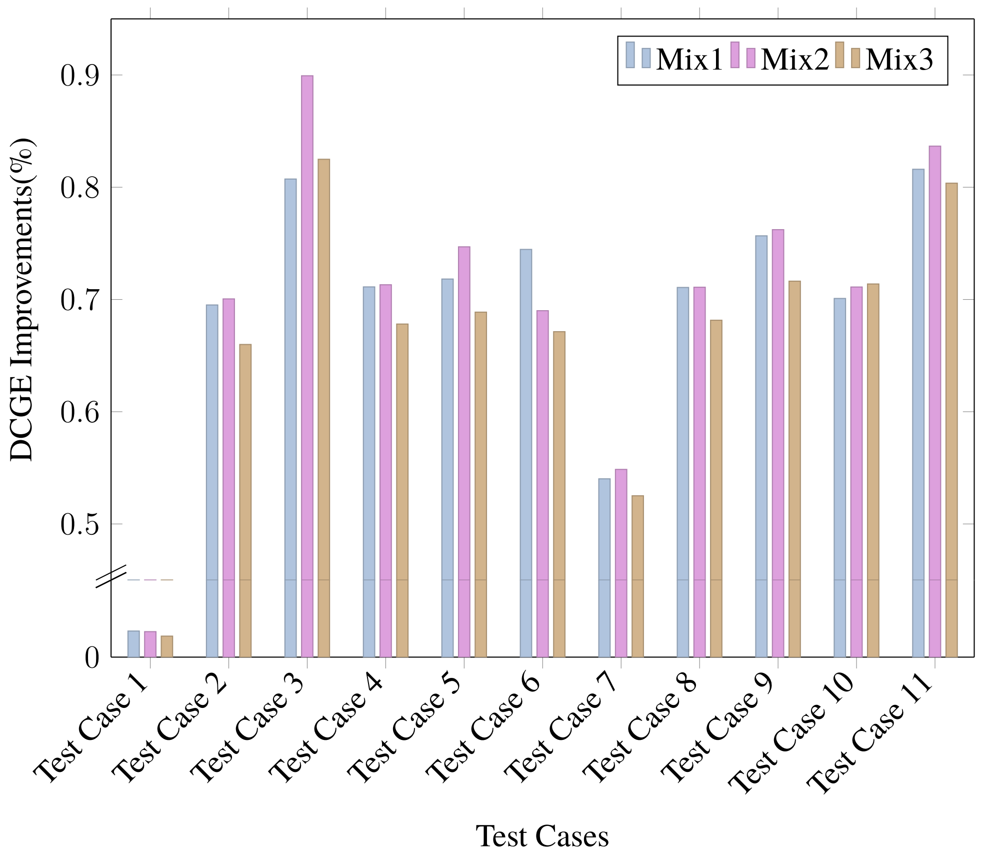


Figure 4.6: DCGE improvements for Block 1 at RTL level

extreme mix) achieves slightly better improvements compared to the other two mixes.

Table 4.4 presents the average and maximum DCGE improvement for each FSDB mix configuration. Among the three mixes, Mix 2 achieves the highest average and maximum DCGE improvement, with minimal differences compared to the other two mixes.

This aligns with the conclusions drawn from the dynamic power savings analysis: Weights do not significantly impact the optimization effect because PowerPro performs structural-based optimization.

In summary, the PowerPro-optimized code demonstrated improvements in both total dynamic power and DCGE in the RTL power analysis. While the FSDB weight factors had little impact on the overall optimization effectiveness, Mix 2, the idle/stall mix, achieved the best results among the three configurations. Specifically, it delivered an average DCGE improvement

Table 4.4: DCGE improvements for Block 1 at RTL level

	Mix1	Mix2	Mix3
Average DCGE improvement	0.66%	0.67%	0.64%
Max DCGE improvement	0.81%	0.90%	0.82%

of 0.67% and an average total dynamic power saving of 4.77%.

Notably, these results highlight that even a slight percentage improvement in DCGE can lead to significant total dynamic power savings at both the IP and SoC levels.

4.2.4 Gate-level power analysis

To further validate the results of the RTL power analysis, the optimized code was synthesized and subjected to gate-level power analysis. Since Mix 2 demonstrated slightly better improvements compared to the other FSDB mix configurations in the RTL power analysis, the PowerPro-optimized RTL code corresponding to Mix 2 was selected for synthesis. The baseline RTL code was also synthesized for comparison.

The timing constraints for the synthesis for both sets of codes were relaxed to 1.07 times the original clock period to prevent excessive negative slack, which could otherwise affect the accuracy of the gate-level power analysis. All other synthesis settings were kept identical for both designs.

After synthesis, the clock gating summary in the synthesis report showed that the optimized code achieved approximately 1% higher SCGE compared to the baseline code. This indicates an improvement in the proportion of registers gated by clock gating, confirming that the optimization effectively enhanced clock gating.

The metrics collected from the gate-level power analysis of the two netlists were used to build the baseline and optimized curves at the gate level. Figure 4.7 presents the baseline and optimized total dynamic power curves for Block 1 at gate level. The figure displays a cropped view to provide a clearer comparison of the curves, focusing on Test Cases 3 to 10.

The optimized total dynamic power curve consistently decreases and appears below the baseline curve in all test cases. This indicates that the dynamic power savings achieved through optimization remain evident and widespread in gate-level power analysis.

Table 4.5 compares the results from the RTL power analysis and the gate-

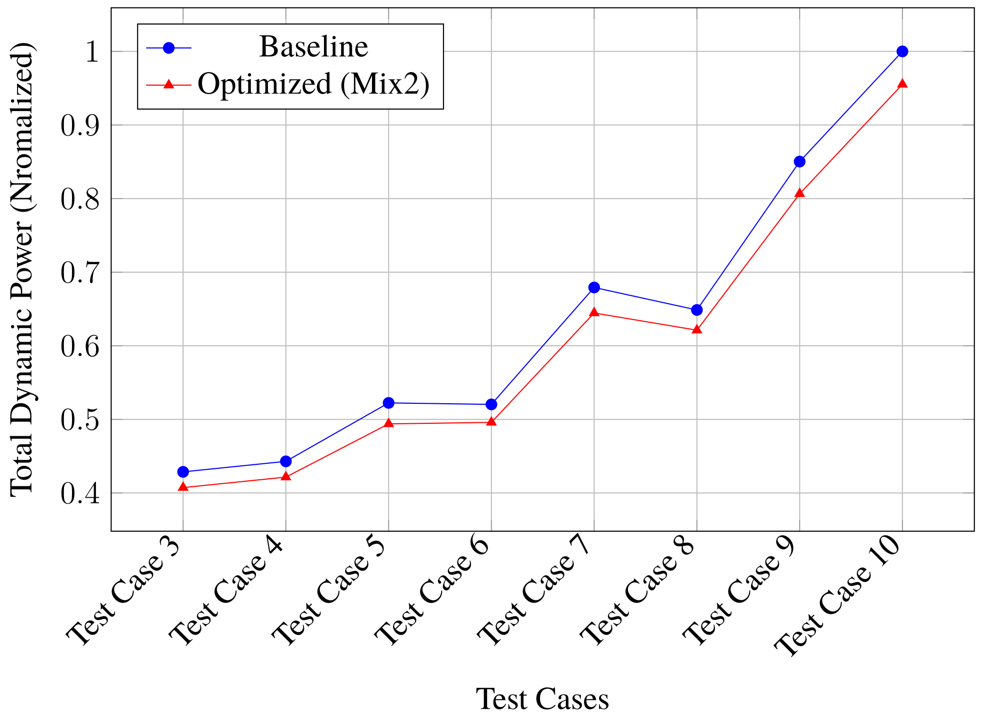


Figure 4.7: Baseline and optimized total dynamic power curves for Block 1 at gate level (Cropped View)

level power analysis. Since gate-level results were derived from the optimized synthesized code using Mix 2, the RTL power analysis results for Mix 2 were used for comparison.

In terms of the average total dynamic power savings, the gate-level result is 4.49%, which shows a minimal difference from the 4.77% observed at the RTL level. Notably, while Test Case 1 remains the operating point with the most minor total dynamic power savings at the gate level, it achieves a 1.64% total dynamic power saving, representing a minor positive optimization. This confirms that the -0.60% saving observed at the RTL level was likely due to inaccuracies in the RTL power analysis, and the actual change at the gate level was a negligible power variation that resulted in a small positive optimization.

Overall, the results confirm that PowerPro delivered significant favorable optimizations for most test cases at both the RTL and gate levels for Block 1.

For DCGE, Figure 4.8 presents the baseline and optimized DCGE curves for Block 1 at the gate level. Across all test cases, the optimized curve consistently increases, appearing above the baseline curve. This indicates that the synthesized netlist of the optimized code achieved enhanced clock gating,

Table 4.5: Total dynamic power savings for Block 1 at RTL vs. Gate Level

	RTL results	Gate-level results
Average total dynamic power saving	4.77%	4.49%
Max total dynamic power saving	9.87%	6.33%
Minimum total dynamic power saving	-0.60%	1.64%

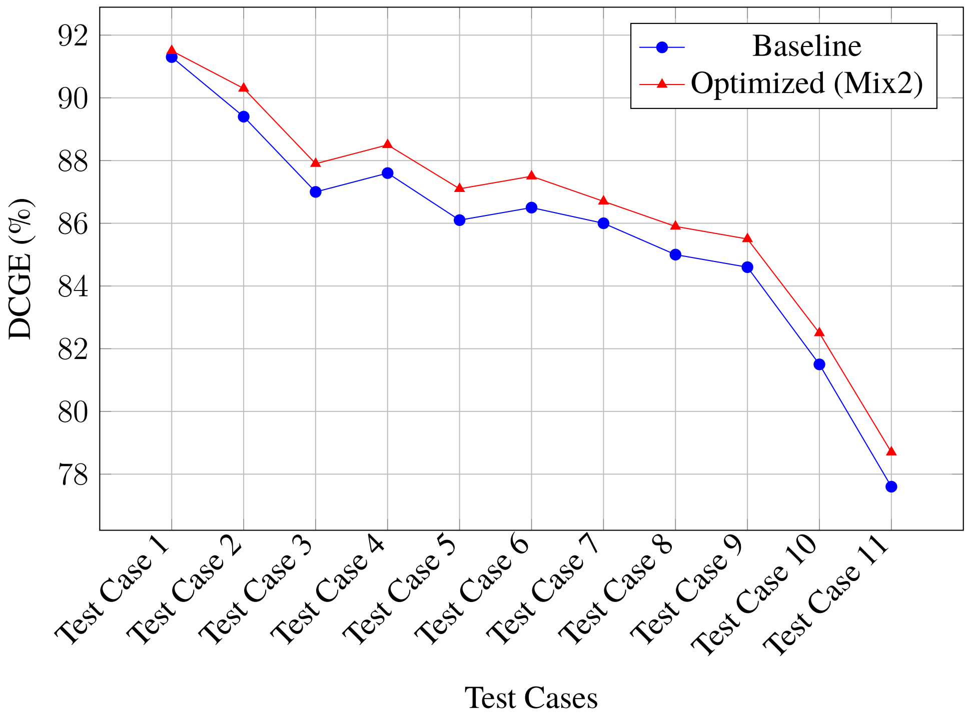


Figure 4.8: Baseline and optimized DCGE curves for Block 1 at gate level

suppressing more clock toggles. This improvement aligns with the observed reductions in total dynamic power at the gate level.

Table 4.6 compares the DCGE improvements observed in the RTL and gate-level power analyses. At the gate level, all test cases exhibited DCGE improvements, with the smallest improvement being 0.2%. The average DCGE improvement was 0.84%, and the maximum reached 1%. These results are consistent with the observations at the RTL level, confirming that optimizations brought by PowerPro significantly enhanced DCGE across all test cases.

In summary, for the gate-level power analysis of Block 1, the netlist

Table 4.6: DCGE Improvements for Block 1 at RTL vs. Gate Level

	RTL results	Gate-level results
Average DCGE improvement	0.67%	0.84%
Max DCGE improvement	0.90%	1.0%
Minimum DCGE improvement	0.05%	0.2%

synthesized from the PowerPro-optimized design demonstrated an average total dynamic power saving of 4.49% and an average DCGE improvement of 0.84% compared to the original design. These results represent the final confirmed optimizations and will serve as the basis for the subsequent calculations in the estimation of manual optimization.

4.2.5 Estimation of manual optimization

After completing the RTL and gate-level power analyses, the method described in Section 3.6.3 was employed to identify manual optimization opportunities that remain beyond automated optimizations from PowerPro. From the register matrix in the register view of the Spyglass Power GUI for the PowerPro-optimized code, a list of registers was extracted. Using the script, the top 10 registers and arrays were filtered out for potential manual optimization based on the criteria of ROADF less than 12.5%, DCGE less than 25%, and the largest bit-width. The total bit-width of these registers and arrays (B_{manual}) was calculated to be 1285.

Furthermore, as described in Section 3.6.1.3, the total bit-width of the PowerPro optimized registers (B_{PowerPro}) was determined to be 700. From the results in Section 3.6.2, these registers were confirmed to contribute an average total dynamic power saving (P_{PowerPro}) of 4.49% and an average DCGE improvement ($DCGE_{\text{PowerPro}}$) of 0.84% .

Using this data and Equations 3.1 and 3.2, it was estimated that manual optimization could provide an additional improvement of up to 8.24% in dynamic power savings ($DCGE_{\text{manual}}$) and up to 1.54% in DCGE ($DCGE_{\text{manual}}$). This suggests that, when combined with the automated PowerPro optimizations, manual efforts could ideally achieve a total of 12.73% in dynamic power savings and 2.38% in DCGE improvement. These results highlight the potential of integrating manual optimization with automated tools to further optimize power by improving clock gating.

4.2.6 Clock gating threshold analysis

This section operates independently of the optimization performed on Block 1, which has been described in the previous sections in this chapter. It focuses on analyzing the impact of different clock gating thresholds on the power and DCGE of the block. Both Spyglass Power and PowerPro were configured with clock gating thresholds ranging from 4 to 16, in increments of 2. Using these varying thresholds, PowerPro was employed to optimize the RTL code, followed by a RTL power analysis of both the baseline and post-optimized designs. The total dynamic power and DCGE data were collected for each threshold value and line graphs were constructed to visualize the results.

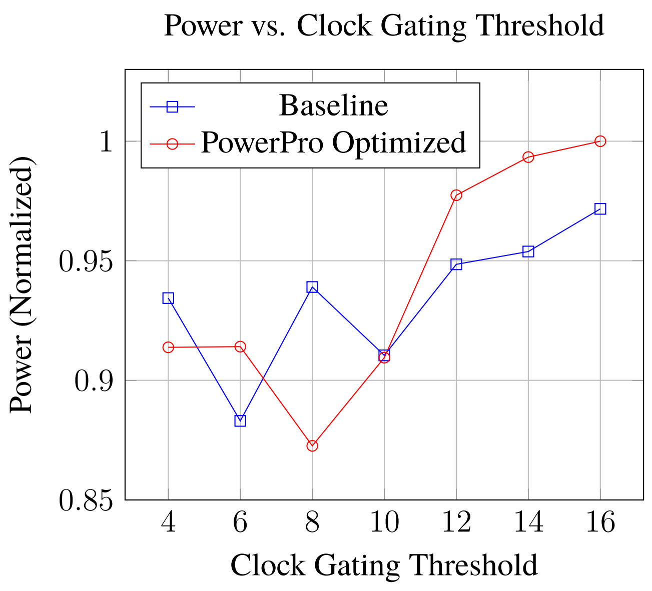


Figure 4.9: Impact of clock gating threshold on total dynamic power

Figure 4.9 illustrates the variation in total dynamic power with different clock gating thresholds for both baseline and optimized designs. In the curves, power values fluctuate with the threshold for both the baseline and optimized designs without exhibiting a uniform trend. For the baseline design, a threshold of 6 results in the lowest power, while for the optimized design, a threshold of 8 achieves the most significant power savings and the lowest power after optimization.

The principles of clock gating can explain this behavior. For the same design, if the clock gating threshold is too high, synthesis tools may transform the originally inserted clock gating enable conditions into redundant reiterating multiplexers to maintain logical correctness. This not only fails to stop the clock toggles effectively, but also introduces additional logic, increasing power consumption. Contrarily, if the clock gating threshold is too low, an excessive number of modules may be gated, resulting in a higher quantity of clock gating logic. The clock gating logic itself consumes power, and when too much of it is introduced, the additional power overhead can offset or even exceed the power savings achieved through clock gating. This trade-off highlights the importance of selecting an appropriate clock gating threshold to balance the benefits of reduced clock toggles against the overhead of additional clock gating logic.

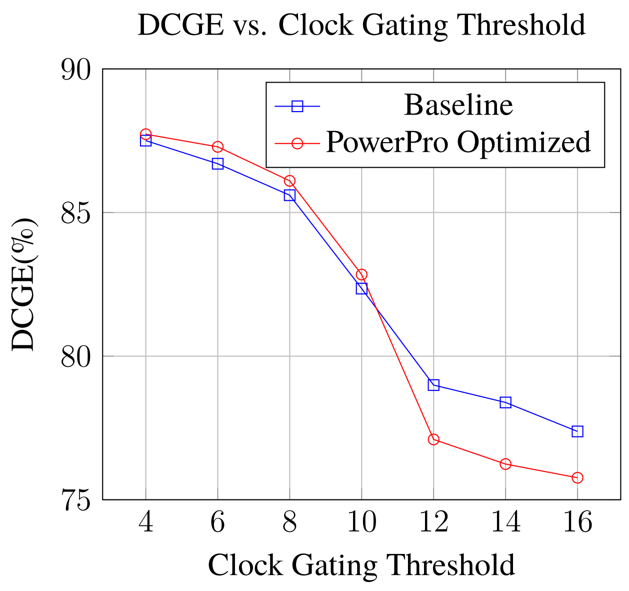


Figure 4.10: Impact of clock gating threshold on DCGE

Figure 4.10 shows the variation in DCGE with different clock gating thresholds for both the baseline and optimized designs. In both cases, DCGE decreases as the clock gating threshold increases. A higher clock gating threshold means that fewer registers meet the clock gating condition, reducing the overall number of gated registers. Consequently, more registers remain active, leading to increased clock toggles relative to data toggles, which decreases the DCGE.

The above analysis is insufficient to determine the most suitable clock gating threshold for this block, as such a decision requires a more detailed gate-level power analysis. However, its significance lies in highlighting that, as shown in Figure 4.9, simply changing the clock gating threshold for the same design can result in up to a 10% variation in power consumption.

This demonstrates that beyond modifying the code to enhance DCGE, synthesis script or constraints optimization can unlock additional clock gating opportunities with minimal effort. For example, the synthesis report should be examined for the ungated registers caused by the “minimum bit-width not met” condition. These registers have enable conditions for clock gating defined in the code, but due to insufficient bit-width, their clock signals remain ungated during synthesis. This not only fails to reduce clock toggles but also introduces redundant reiterating multiplexers, which unnecessarily increase power consumption.

To address this, designers should consider adjusting clock gating conditions or modifying the clock gating threshold to prevent such occurrences and thereby further reduce power consumption.

4.3 Optimization of Block 2

Block 2 is the arbiter and router within EMCA, an important component in the system that works collaboratively with the DSP. It is a smaller-scale design compared to the DSP IP block, and hierarchical clock gating has not been applied to this block.

4.3.1 Baseline power metrics curves and optimization settings

Since Block 2 is a relatively small block, its maximum power consumption cannot reach exceptionally high values. Therefore, fewer test cases were selected for power analysis and optimization compared to Block 1, allowing for interpolation and the construction of metrics curves with fewer data.

Figure 4.11 demonstrates the dynamic power curves of the baseline design of Block 2. Test Cases 1a to 5a are arranged in ascending order of total dynamic power. Unlike Block 1, this module does not utilize hierarchical clock gating, so there are no test cases where the block can enter a complete sleep state.

Figure 4.12 presents the DCGE and activity curves of the baseline design of Block 2. As activity increases, DCGE decreases, which is consistent with expected behavior. Across these five test cases, the measured DCGE values range from 84% to 86%.

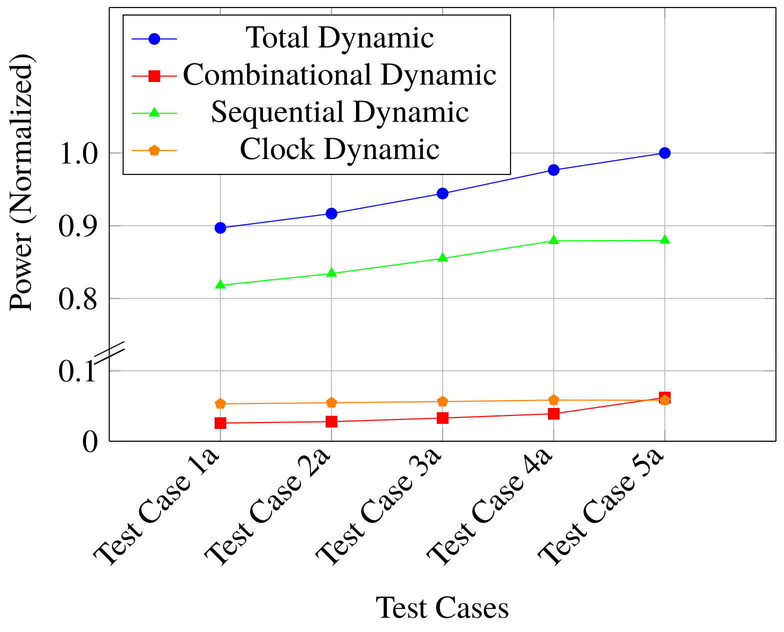


Figure 4.11: Dynamic power curves of the baseline design of Block 2

In the above analysis, Test Cases 1a through 5a did not exhibit significant variations in power levels. Furthermore, the analysis of Block 1 demonstrated that different combinations of FSDB weight factors did not have a substantial impact on the optimization results of PowerPro. Based on these findings, the FSDB weight factors chosen for PowerPro optimization for Block 2 in this study were evenly distributed across the five test cases. Each FSDB file was assigned a weight factor of 20%.

4.3.2 RTL power analysis

After PowerPro optimized the baseline design using the original design and the five FSDB files, the generated code was analyzed with Spyglass Power to collect relevant metrics and rebuild the curves.

Figure 4.13 presents the baseline and optimized total dynamic power curves for Block 2 at the RTL level. It can be observed that the optimized curve shows minimal changes compared to the baseline curve. This may be attributed to inaccuracies in RTL power analysis for smaller modules. Therefore, exact power values and optimization effects should be further verified through gate-level power analysis.

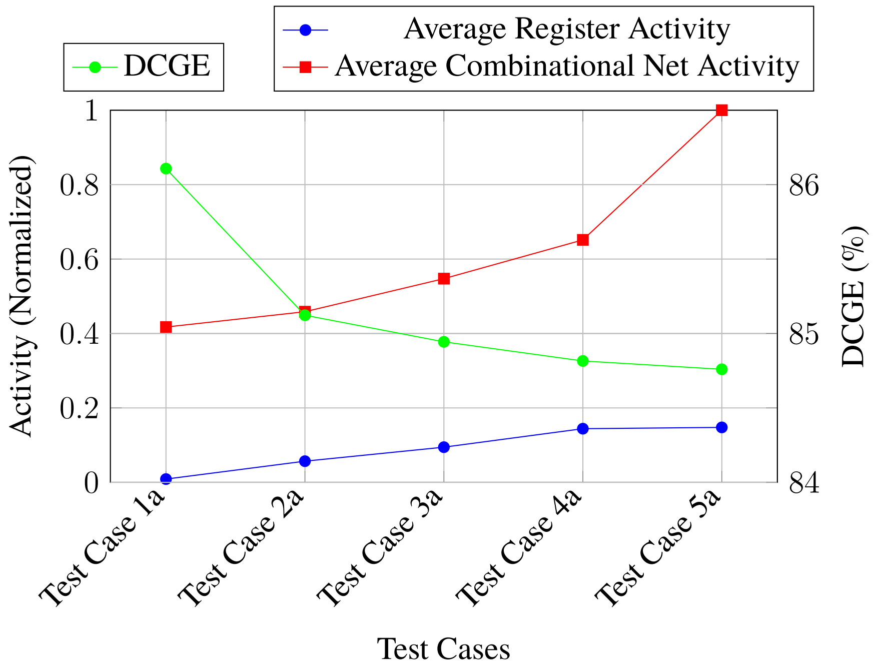


Figure 4.12: DCGE and activity curves of the baseline design of Block 2

Figure 4.14 shows the baseline and optimized DCGE curves for Block 2 at the RTL level. It is evident that the optimized curve consistently increases, appearing above the baseline curve, indicating a significant improvement in the DCGE of Block 2 after optimization.

4.3.3 Gate-level power analysis

To further validate the results of the RTL power analysis, the optimized code for Block 2 was synthesized and subjected to gate-level power analysis. For Block 2, the original timing constraints were retained during synthesis, as neither the baseline nor the optimized design exhibited any negative slack.

According to the synthesis report, SCGE improved by 1.72%, indicating a significant increase in the number of gated registers after optimization. This shows that the clock gating of Block 2 was noticeably enhanced.

Figure 4.15 illustrate the baseline and optimized total dynamic power curves for Block 2 at gate level. Across all test cases, the optimized curve consistently lies below the baseline curve, with a noticeable downward shift. This indicates that the optimized netlist for Block 2 achieved significant reductions in total dynamic power under these test cases. These results suggest

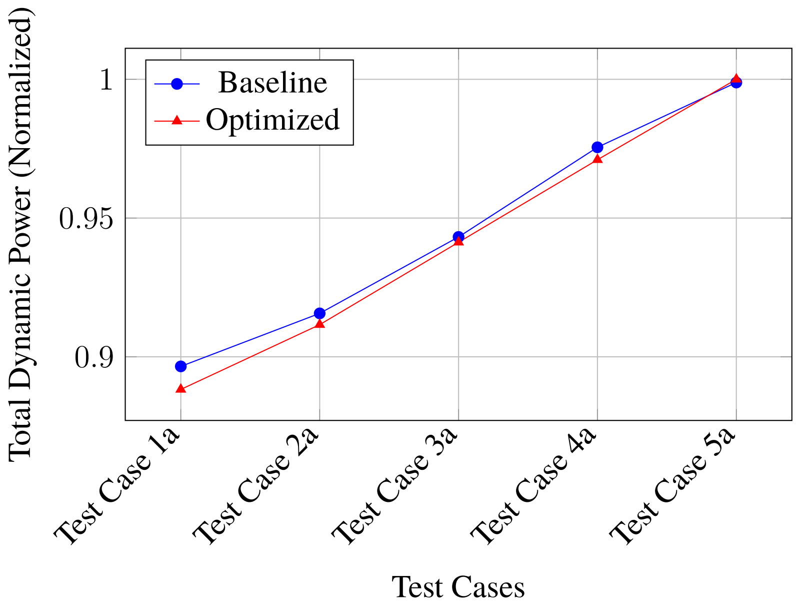


Figure 4.13: Baseline and optimized total dynamic power curves for Block 2 at RTL level

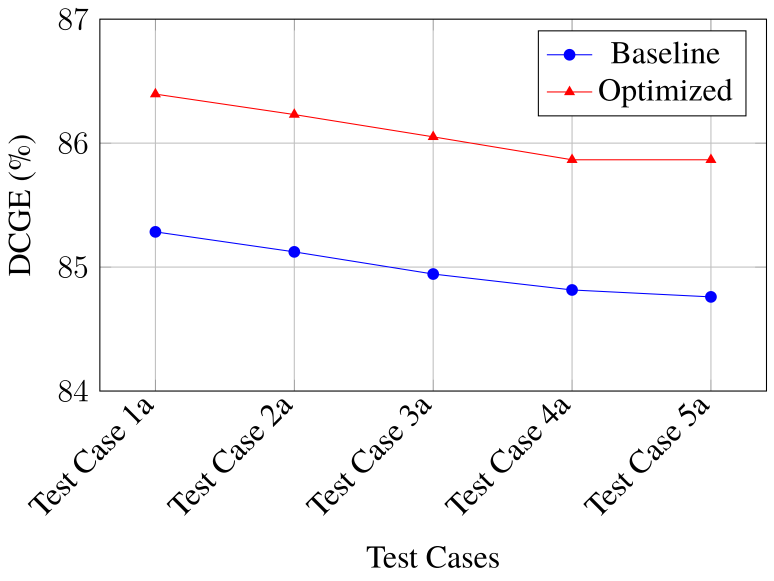


Figure 4.14: Baseline and optimized DCGE curves for Block 2 at RTL level

that the lack of observable total dynamic power optimization in the RTL power analysis was probably due to inaccuracies inherent in the RTL power analysis

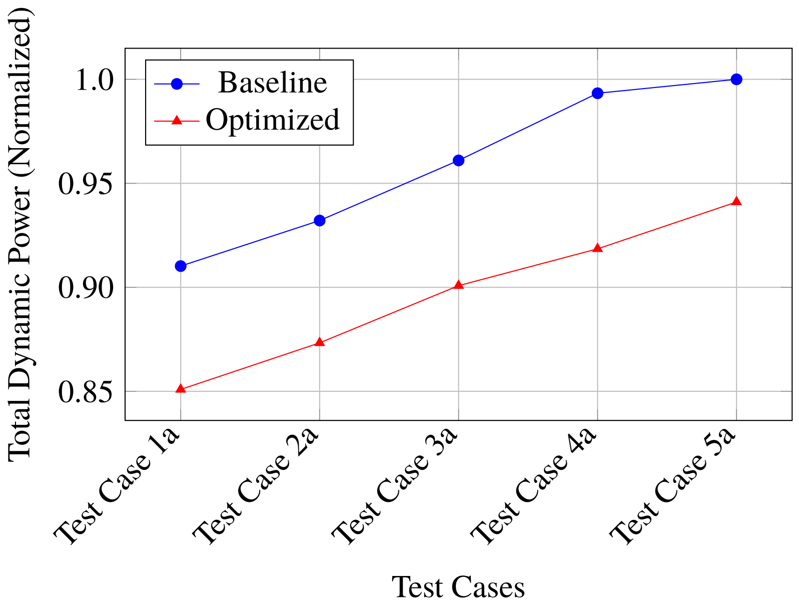


Figure 4.15: Baseline and optimized total dynamic power curves for Block 2 at gate level

rather than the absence of actual improvements.

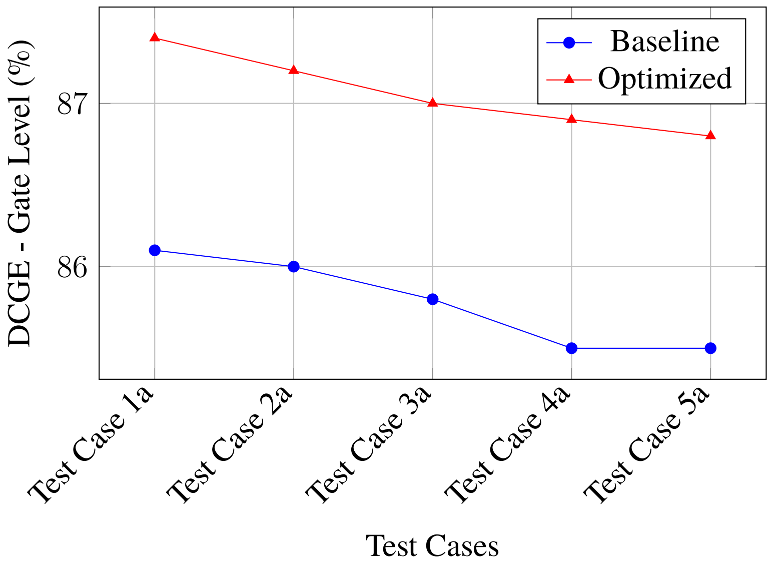


Figure 4.16: Baseline and optimized DCGE curves for Block 2 at gate level

Figure 4.16 presents the baseline and optimized DCGE curves for Block 2 at gate level. Across all test cases, the optimized curve consistently rises and lies above the baseline curve. This indicates that the optimized netlist for Block 2 achieved an increase in DCGE, demonstrating enhanced clock gating efficiency in these test cases.

Table 4.7: Optimization results for Block 2 at RTL vs. Gate Level

	RTL results	Gate-level results
Average total dynamic power saving	0.41%	6.52%
Average DCGE improvement	1.10%	1.28%

Table 4.7 compares the optimization results for Block 2 observed in the RTL power analysis and the gate-level power analysis. At the gate level, the average total dynamic power saving is 6.52%, significantly higher than the result observed at the RTL level. Since Block 2 is a relatively small module, the actual power values at the RTL level may be more prone to inaccuracies. Therefore, the results of the gate-level power analysis are considered more reliable.

The average DCGE improvement at the gate level is 1.28%, which shows a minimal difference compared to the RTL result. This further confirms the consistency of the optimization's impact on clock gating efficiency.

4.3.4 Estimation of manual optimization

The method used for the estimation of manual optimization for Block 2 is consistent with that used in Block 1.

Using the Python script, the top 10 registers and arrays were filtered out for potential manual optimization, and the total bit-width of these registers and arrays (B_{manual}) was calculated to be 355.

The total bit-width of registers optimized by PowerPro (B_{PowerPro}) was determined by the Python script to be 109. These registers were confirmed to contribute an average total dynamic power saving (P_{PowerPro}) of 6.5% and an average DCGE improvement ($DCGE_{\text{PowerPro}}$) of 1.3% .

Using these data and Equations 3.1 and 3.2, it was estimated that manual optimization could provide an additional improvement of up to 21.2% in dynamic power savings ($DCGE_{\text{manual}}$) and up to 4.2% in DCGE ($DCGE_{\text{manual}}$).

Using the above data, when manual optimizations can be applied to Block 2 in addition to the automatic optimization, it could ideally achieve a total of

27.7% in total dynamic power savings and 5.5% in DCGE improvement.

Chapter 5

Conclusions and Future work

5.1 Conclusions

This thesis has demonstrated a systematic approach to the hybrid power optimization of IP blocks within the EMCA architecture, focusing on improving the efficiency of the local clock gating. The methodology, which combines automated optimization using PowerPro with subsequent manual optimization opportunities, has shown significant potential to improve energy efficiency while preserving the functional integrity of the design.

The hybrid optimization flow successfully improved the clock gating efficiency and saved power. For Block 1 (DSP), it is estimated that up to 12. 73% dynamic power savings and 2. 38% DCGE improvement will be achieved in total if manual optimization can be applied. For Block 2 (arbiter and router), up to 27.7% dynamic power saving and 5.5% DCGE improvement in total can be estimated if manual optimization can be applied to the top 10 registers. These demonstrated improvements in energy efficiency contribute to the overall power savings in EMCA.

The findings highlight the importance of improving the efficiency of local clock gating, particularly for blocks like the DSP, where the hierarchical clock gating already offers substantial energy savings. Additionally, analysis of clock gating thresholds emphasized the importance of fine-tuning synthesis constraints to further enhance clock-gating performance.

The hybrid optimization flow is practical and easy to implement. By treating the design as a black box, engineers outside the design team can easily use the entire flow and complete the analysis and optimization of a block in as little as a few hours.

5.2 Future work

The manual optimization phase in this study was limited to generating a list of registers with high potential for optimization. Future efforts should focus on implementing these manual optimizations, validating the effectiveness of the identified registers, and further improving the clock gating efficiency and saving overall power.

In the automated optimization phase using PowerPro, this study primarily focused on the largest hierarchical instances within the IP blocks that could be directly input into the tool as top-level modules. Future work could explore optimizing smaller sub-blocks individually, comparing the optimization effect achieved at this finer granularity. This would provide valuable insight into the scalability and flexibility of the optimization flow.

Lastly, the flow can be extended to other IP blocks within the EMCA architecture. Expanding the application of the flow would not only validate its effectiveness across various designs but would also contribute to improving the overall power efficiency of the EMCA.

References

[1] "Breaking the energy curve," Ericsson, 2020. [Online]. Available: https://www.ericsson.com/en/about-us/sustainability-and-corporate-responsibility/environment/product-energy-performance [Pages xi, 1, and 2.]

[2] B. Ekelund, "Semiconductor challenges in the 5g and 6g technology platforms," in *2023 International Electron Devices Meeting (IEDM)*. IEEE, 2023, pp. 1–5. [Page 2.]

[3] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolić, *Digital integrated circuits: a design perspective*. Pearson Education, Incorporated., 2003. [Page 5.]

[4] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," *Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, 2003. [Page 6.]

[5] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-v power supply high-speed digital circuit technology with multithreshold-voltage cmos," *IEEE Journal of Solid-state circuits*, vol. 30, no. 8, pp. 847–854, 1995. [Page 6.]

[6] T.-H. Kim, J. Liu, J. Keane, and C. H. Kim, "Circuit techniques for ultra-low power subthreshold srams," in *2008 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2008, pp. 2574–2577. [Page 6.]

[7] A. Keshavarzi, S. Ma, S. Narendra, B. Bloechel, K. Mistry, T. Ghani, S. Borkar, and V. De, "Effectiveness of reverse body bias for leakage control in scaled dual vt cmos ics," in *Proceedings of the 2001 international symposium on Low power electronics and design*, 2001, pp. 207–212. [Page 6.]

[8] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, “Low-power cmos digital design,” *IEICE Transactions on Electronics*, vol. 75, no. 4, pp. 371–382, 1992. [Pages 7 and 8.]

[9] H. J. Veendrick, “Short-circuit dissipation of static cmos circuitry and its impact on the design of buffer circuits,” *IEEE Journal of Solid-State Circuits*, vol. 19, no. 4, pp. 468–473, 1984. [Page 7.]

[10] D. Hodges, H. Jackson, and R. Saleh, *Analysis and design of digital integrated circuits*. McGraw-Hill, Inc., 2003. [Pages 7, 8, and 9.]

[11] F. N. Najm, “A survey of power estimation techniques in vlsi circuits,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, 1994. [Page 8.]

[12] Y. Leblebici and S.-M. Kang, *CMOS digital integrated circuits: analysis and design*. McGraw-Hill New York, 1996. [Page 8.]

[13] T. Pering, T. Burd, and R. Brodersen, “The simulation and evaluation of dynamic voltage scaling algorithms,” in *Proceedings of the 1998 international symposium on Low power electronics and design*, 1998, pp. 76–81. [Page 9.]

[14] I. Savvidis, “Auto local cg boost: Eliminating redundant dynamic power at ip level for energy efficiency,” Ericsson, Tech. Rep., 2022, internal Document. [Page 10.]

[15] *PowerPro User Manual*, Version 10.4 ed., Mentor, June 2020. [Page 11.]

[16] A. Priya, T. Agrawal, and M. Saxena, “Early-stage rtl power estimation and exploration.” [Page 11.]

[17] N. Koduri and K. Vittal, “Power analysis of clock gating at rtl,” Technical Report, Atrenta Inc., San Jose, Calif. [Page 14.]

[18] A. Krishnaswamy, “Getting ahead with early power analysis,” *Semiconductor Engineering*, August 9 2018. [Online]. Available: https://semiengineering.com/getting-ahead-with-early-power-analysis/ [Page 15.]

[19] N. Balachandran, “Low power memory controller subsystem ip exploration using rtl power flow: An end-to-end power analysis and reduction methodology,” 2020. [Pages xi, 15, and 27.]

[20] Y. Wang, “Differential energy analysis for improved performance/watt in mobile gpu,” in *Design Automation Conference (DAC)*, 2018. [Pages xi and 16.]

[21] I. Savvidis, “The quest for easy power-aware sw development: A novel approach to dsp code profiling for energy/performance tradeoffs,” in *Design Automation Conference (DAC)*. Ericsson, 2018. [Pages 16 and 18.]

[22] M. Zhang, “Power-aware software development for emca dsp,” 2017. [Page 16.]

[23] I. Savvidis and M. Do, *Power Analysis Flow Rollout*, Ericsson, 2023, internal Document. [Page 22.]

[24] I. Savvidis, *VCD2RPT++ ActivityExplorer Flow Guide*, Ericsson, 2015, internal Document. [Page 23.]

[25] I.Savvidis, “Dump, convert and replay: A targeted methodology to mitigating power simulations effort,” in *Design Automation Conference (DAC)*. Ericsson, June 2019. [Page 24.]

[26] Z. Zhang, “Energy efficient ericsson many-core architecture (emca) ip blocks for 5g asic,” 2021. [Pages 30 and 32.]

[27] *PowerPro Reference Manual*, Version 10.4 ed., Mentor, June 2020. [Page 30.]

[28] I. Savvidis, *PowerPro template-based optimization flow*, Ericsson, 2022, internal Document. [Page 32.]

TRITA – EECS-EX 2025:78
Stockholm, Sweden 2025

www.kth.se