OULUN YLIOPISTO
UNIVERSITY of OULU

DEGREE PROGRMME IN ELECTRICAL ENGINEERING

MASTER'S THESIS

REGISTER-TRANSFER LEVEL POWER ESTIMATION AND REDUCTION METHODOLOGIES OF DIGITAL SYSTEM-ON-CHIP BUILDING BLOCKS

Author	Miikka Haataja
Supervisor	Jukka Lahti
Second Examiner	Timo Rahkonen
Technical Advisor	Ari Oja

March 2016

Haataja M. (2016) Register-Transfer Level Power Estimation and Reduction Methodologies of Digital System-on-Chip Building Blocks. University of Oulu, Degree Programme in Electrical Engineering. Master's Thesis, 69 p.

ABSTRACT

This thesis is a study of register-transfer level power estimation and reduction methodologies for digital system-on-chip building blocks. In the theory section, the components of power dissipation for current circuit technology are explained in details, the commonly implemented register-transfer level power estimation methodologies are classified and explained, and finally, commonly used power reduction methods used in system-on-chip development are presented.

In the implementation part of this thesis, register-transfer level power estimation and power reduction methodologies with a state-of-the-art commercial register-transfer level power tool are presented. Results obtained with these methodologies are analyzed for three different system-on-chip building blocks. The experimental results of power estimation accuracy and power saving estimates are presented. The average deviation between register-transfer level and gate-level power estimation were 11 %, and potential total power saving estimates were between 10 % and 29 %.

Keywords: register-transfer level, power estimation, power reduction, system-on-chip.

Haataja M. (2016) Digitaalisen järjestelmäpiirilohkojen rekisterinsiirtotason tehonkulutuksen arviointi ja vähennysmenetelmät. Oulun yliopisto, sähkötekniikan koulutusohjelma. Diplomityö, 69 s.

TIIVISTELMÄ

Tässä työssä tutkitaan rekisterinsiirtotason tehonkulutuksen arviointi- ja vähennysmenetelmiä digitaalisille järjestelmäpiirilohkoille. Teoriaosuudessa esitetään tehonkulutuksen eri komponentit nykyiselle piiriteknologialle, luokitellaan yleisimmät rekisterinsiirtotasolla käytettävät tehonkulutuksen arviointimenetelmät sekä kuvataan yleisesti digitaalisten järjestelmäpiirien suunnittelussa käytettyjä tehonvähennysmenetelmiä.

Kokeellisessa osassa kuvataan rekisterinsiirtotason tehonkulutuksen arviointi- ja vähennysmenetelmä käyttäen kaupallista rekisterinsiirtotason tehotyökalua. Menetelmiä testataan kolmella digitaalisella järjestelmäpiirilohkolla ja saatuja tuloksia analysoidaan tehonkulutuksen arvion tarkkuuden ja tehonvähennyksen arvioiden kannalta. Näiden kolmen järjestelmäpiirilohkon tulokset tehonkulutuksen ja tehonvähennyksen arviosta on esitetty. Rekisterisiirtotason tehonarviointi poikkesi keskimäärin 11 % porttitason vertailuarviosta, ja potentiaaliset tehonvähennysarviot olivat väliltä 10 – 29 %.

Avainsanat: rekisterinsiirtotaso, tehonkulutuksen arviointi, tehonvähennys, järjestelmäpiiri.

TABLE OF CONTENTS

FOREWORD

The purpose of this thesis was to study the register-transfer level power estimation and reduction methodologies for system-on-chip building blocks based on a commercial state-of-the-art register-transfer level power tool. The thesis work was carried out in Nokia Solutions and Networks between the summer of 2015 and the early beginning of 2016.

I would like to thank Ari Oja for acting as a technical advisor for this thesis and for coming up with the idea for this thesis, also thanks for my manager Juha Yrjänäinen for valuable comments and making the thesis work possible. From the University of Oulu, I'd like to thank Chief Engineer Jukka Lahti for supervising this work and Professor Timo Rahkonen for acting as a second examiner. A special thanks to the application engineers of the tools used in this thesis as they provided valuable support for the set up and usage of the tools. I would also like to thank those whose name was not mentioned specifically here as a lot of colleagues from Nokia Solutions and Networks were involved helping me with this thesis work. Last but not least, I would like to thank my family, Mom and Dad, for constant support during all my studies.

Oulu, March 1st, 2016

Miikka Haataja

LIST OF ABBREVIATIONS AND SYMBOLS

CMOS	Complementary Metal-Oxide-Semiconductor
DSP	Digital Signal Processing
DVFS	Dynamic Voltage and Frequency Scaling
DVS	Dynamic Voltage Scaling
EDA	Electronic design automation
FET	Field-Effect Transistor
FSM	Finite-State Machine
FSMD	Finite-State Machine with Datapath
HLS	High-Level Synthesis
IC	Integrated Circuit
ICT	Information and Communication Technologies
ITRS	International Technology Roadmap for Semiconductors
LSB	Least Significant Bit
LUT	Look-Up Table
MOS	Metal-Oxide-Semiconductor
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MSB	Most Significant Bit
ODC	Observability Don't Care
RT	Register-Transfer
RTL	Register-Transfer Level
SoC	System-On-Chip
STC	Stability Condition
WLM	Wire Load Model

C	capacitance
E	potential energy, energy
f	frequency
f_{clk}	clock frequency
I, i	current
V, v	voltage
P	power
T	period-time of a signal
V_{dd}	supply voltage
V_T	threshold voltage

α	switching activity factor
β	gain factor of a MOS transistor
τ	rise or fall time of a signal

1. INTRODUCTION

Power cannot be considered as a design consideration solely of system-on-chips (SoC) for low-power battery operated mobile devices anymore as in recent years, power has become critical and a limiting design constraint for a lot of SoC developments. Increasing performance requirements and complexity of the devices have drastically increased the power dissipation, while demand and need for low-power electronics in this mobile era is stronger than ever before. A high power consumption does not only create challenges for the supply power of an SoC, but also has significant negative impacts in terms of degrading reliability and increasing the cost of packaging and cooling of the device. The increase in power related design issues has imposed numerous challenges on the traditional style of SoC design, as well as to the electronic design automation (EDA) tools used by the industry, that originally have been focusing on optimizing the speed and area of the SoC's.

The scaling of the current dominant circuit technology, complementary metal-oxide-semiconductor (CMOS) technology, has been the driving force for the boom of modern electronics by enabling the performance increase and the device cost reduction with every new generation of devices. Despite of the success of CMOS technology, the rapid technology scaling itself has become one of the reasons for power becoming a major design issue at the first place. As the power dissipation of a transistor does not scale down with the same rate as that of the physical size in CMOS circuits, the power dissipation will only continue to increase when more and more transistors are packed within the same surface areas [1 p. 109]. Due to this, the next generations of high performance chips are consuming ever more power if low-power design measures are not taken. In the 90s, there was seen an exponential increase of power dissipation in high performance chips, and there were even estimations that the power density of SoCs would reach the same level as the sun's surface in the future [2]. Furthermore, there was even cancelation of releasing a high performance processor due to power-thermal issues [1 p. 110], which ultimately led to the development of modern multi-core processors to continue increase performance while managing power. With the start of exponential global demand for low-power battery operated devices, the 90s can be seen as the turning point for the traditional SoC design methodology and the starting point of development for the new low-power design methodologies and technologies. The recent industry publications have been often referring to this phenomenon as the "power wall" [3 p. 2], which reflects the fact that power limits current SoC design. Furthermore, some of the publications have been predicting that the scaling of CMOS technology is coming to an end and new circuit technologies need to be developed and adopted [1].

The rising awareness of the global climate change is the latest driving force for the low-power development, as it has made the energy efficiency as an issue for all electronic devices. In the past, not much attention was paid to the power consumption of the devices operating straight from the power grid, but today, it is a major consideration for a new device. As SoCs are vital building blocks of all modern electronics and large contributors of the total power dissipation, the semiconductor industry has been forced to take steps for reducing power consumption of all produced SoCs. For example, it is estimated that communication networks alone were responsible for about 2 % or 350 TWh of the total worldwide electricity consumption in 2012 [4 p. 1]. This does not only have a large contribution to the greenhouse gas emissions, but it also directly contributes to the operating costs

of the networks as well. The power consumption of the communication networks will be a major challenge in the future, as it is expected that the 25 billion devices connected to the Internet in 2015 will double to 50 billion only in five years [5 p. 3]. This will drastically increase traffic and needed network infrastructure of all communication networks, thus increasing the power consumption and emissions if power efficiency of the new network infrastructure is not increased.

As power has become more important than before, the classical way of only performing circuit-level power estimation and optimizations does not yield sufficient results in terms of SoC design power dissipation. The design decisions made early in the design flow have much higher impact than the later decisions on power [6 p. 1], thus poor power decisions made in the early stages of design flow cannot be overcome in the later stages. Power estimation and optimizations must start as early as possible and be performed throughout the design process to make sure how each design decision affects the power. With this approach, it is possible to design for low-power without performing any costly design iterations late in the design and meet competitive low-power, performance, area and time-to-market requirements. Furthermore the importance of high-level power tools is increasing, as SoC design is constantly developing towards higher abstraction levels and human interaction in gate-level and below in the flow is minimal as the process is highly automated in the modern EDA tools. In the past two decades, the research on high-level power estimation and reduction methods has been very active and we are just starting to see mature state-of-the-art commercial tools available that can produce sufficient results in terms of power estimation accuracy and performing power optimizations. These tools have been gaining the acceptance and adaptation of the industry slowly but steadily.

The aim of this thesis is to study and define a register-transfer level (RTL) power estimation and reduction methodologies based on a commercial RTL power tool for SoC development. Register-transfer (RT) level was chosen as the design abstraction level for this thesis, because the availability of commercial RTL power tools and as RT-level is one of the key points in SoC design flow. On RT-level, the design iteration time for major architectural changes is significantly faster than from gate-level, which enables greater power saving opportunities achieved with less time. The faster design iteration times of RT-level power analysis also enables the designer to explore the power of a design through multiple architecture candidates.

In order to understand the power estimation and reduction techniques, Chapter 2 starts by describing the basic theory behind the power dissipation components in CMOS circuits. Chapter 3 discusses the basic functionalities and theories behind different most common power estimation methods used in order to understand the difficulties and functionality of power estimation tools. Chapter 4 presents the common power reduction techniques for dynamic and static power dissipation. Chapter 5 is the case study chapter of the thesis, where the methodologies and results for RTL power estimation and reduction are presented. Chapter 6 is the discussion chapter and Chapter 7 summarizes the thesis chapter by chapter.

2. POWER DISSIPATION IN CMOS CIRCUITS

Chapter 2 describes the physical phenomenon behind the power dissipation in CMOS circuits. Each power component behind the power dissipation is described in detail using a simple CMOS inverter as an example. Section 2.1 gives a short introduction to CMOS power dissipation, section 2.2 about the dynamic power components and section 2.3 discusses about the static power components. Section 2.4 confers a short summary and classification of the power components.

2.1. Introduction

In CMOS circuits, the power dissipation is typically divided into two main components that are dynamic and static power. The dynamic power refers to the power dissipation that occurs when transistors switch their states in the circuit. On the contrary, the static power is present always when the circuit is powered on and in stable steady state. Unlike the dynamic power, the static power is not dependent on transistors switching their states. [7 p. 1]

Logic gates in CMOS circuits can be thought of as pairs of complementary and symmetrical p-type and n-type field effect transistors (FET), typically metal-oxide-semiconductor field-effect transistors (MOSFET). A simple CMOS inverter can be constructed with two MOSFETs, as illustrated in Figure 1, where on the top is a p-type MOSFET and on bottom is a n-type MOSFET [7 p. 2]. When input voltage V_{In} of the inverter circuit is logical 0, the p-type MOSFET Q_p is conducting and n-type MOSFET Q_n is not conducting and the output voltage V_{out} is at logic level 1. And when the input voltage is at logical 1, the p-type transistor is not conducting, the n-type transistor is conducting and the output is at logical 0. [7 p. 2]

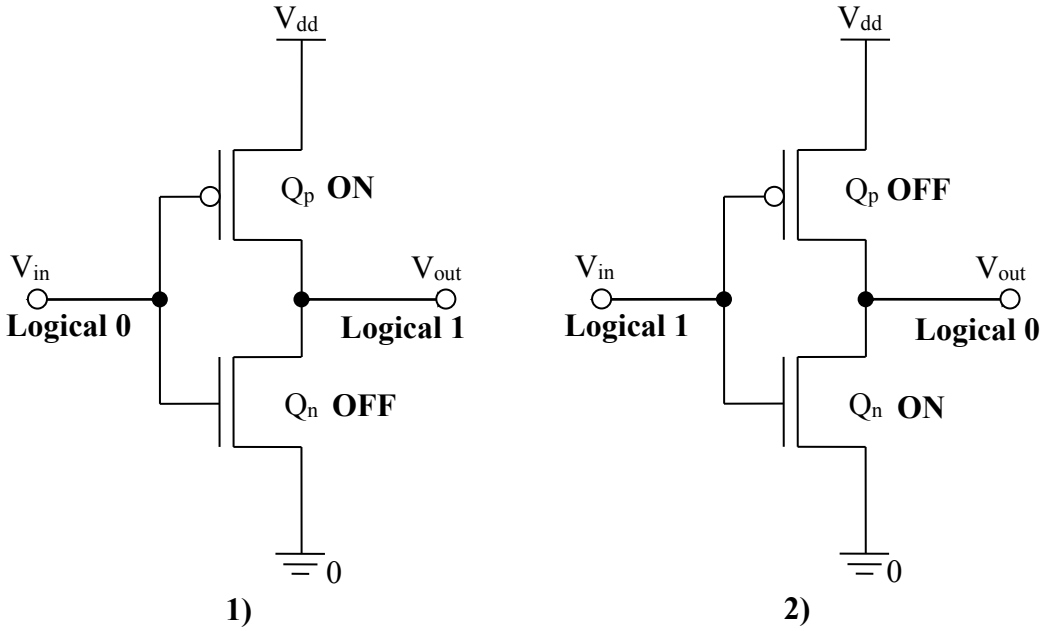

Figure 1. Logic states of CMOS inverter.

2.2. Dynamic Power Components

The dynamic power dissipation consists of two components: short-circuit current and capacitive-load current. The short-circuit current is due to finite rise and fall times of the signals and capacitive-load current is a product of charging and discharging of parasitic load capacitances.

2.2.1. Capacitive-Load Current

When the output voltage V_{out} of the CMOS inverter switches state from logical 0 to 1, a parasitic load capacitance C_{load} is charged from 0 to supply voltage V_{dd}. The loading of the parasitic capacitance causes current $I_{capacitive\text{-}load}$ to flow through the p-type MOSFET, as illustrated in Figure 2 [8]. This current is referred to as the capacitive-load current. The energy stored in the parasitic load capacitance is discharged to the ground, when the output voltage V_{out} the changes state from logical 1 back to 0. The parasitic load capacitances consist of the combination of various intrinsic and extrinsic capacitances, as presented in Figure 3. The main sources of intrinsic capacitances are from diffusion capacitances between the drains and bulk of the MOS transistors as capacitors C_{bn} and C_{bp}, as shown in Figure 3. Extrinsic capacitance is mainly due to the parasitic input capacitance of the driven gate C_{GS} and C_{GD} and the effective interconnect capacitance C_W. The interconnect capacitance tends to dominate in current circuit technologies. [8][9 p. 10-2, 14-1][10 p. 13 – 14][11]

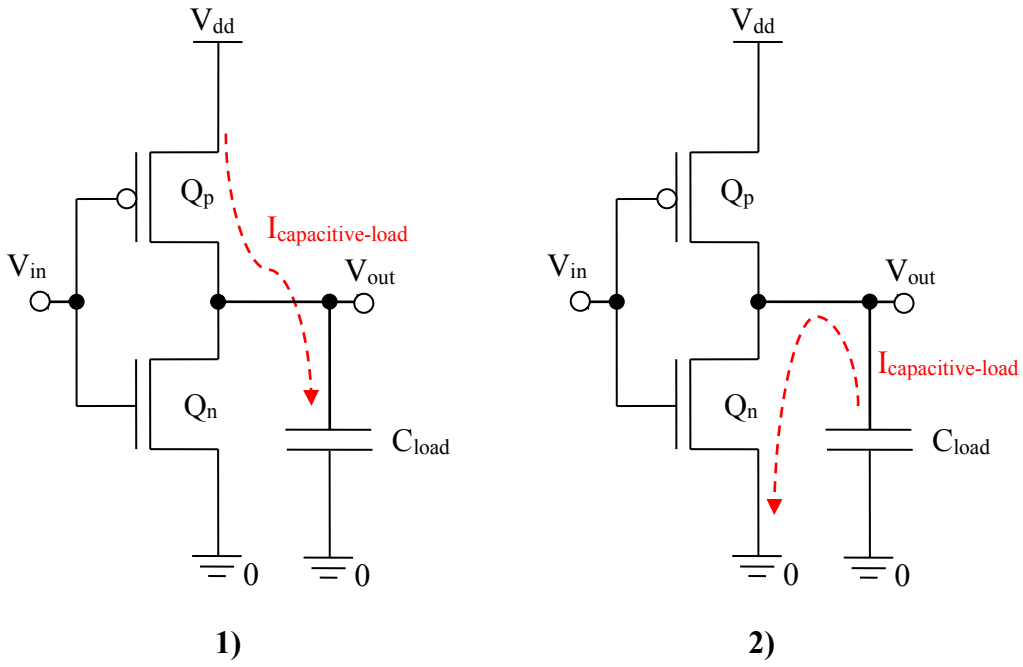

Figure 2. 1) Charging capacitive-load current in CMOS inverter. 2) Discharging capacitive-load current in CMOS inverter.

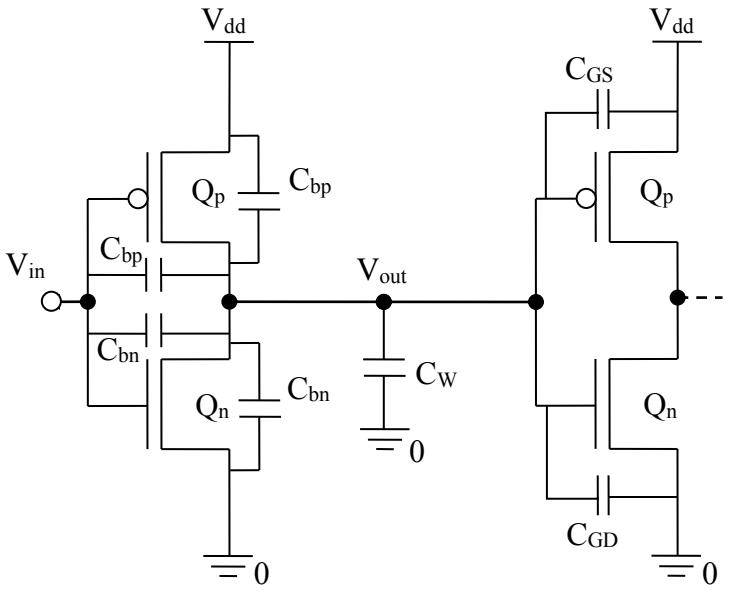

Figure 3. Components of the parasitic load capacitances.

The potential energy drawn from the supply voltage V_{dd} when the parasitic load capacitor is charged can be calculated as

$$E_p = \int_0^\infty i_c(t)v_c(t)dt = V_{dd}\int_0^\infty C_{load}\frac{dv_c}{dt}dt = C_{load}V_{dd}\int_0^{V_{dd}} dv_c = C_{load}{V_{dd}}^2, \quad (1)$$

where i_c is current of the capacitor C_{load}, v_c is the voltage of C_{load}, C_{load} is the sum of the parasitic capacitances and V_{dd} is the supply voltage. The actual energy stored in the capacitor can be calculated with a similar equation as above by replacing the supply voltage with the target voltage of the capacitor. Which results in an equation as

$$E_C = \int_0^\infty i_c(t)v_c(t)dt = \int_0^\infty C_{load}\frac{dv_c}{dt}v_c dt = C_{load}\int_0^{V_{dd}} v_c dv_c = \frac{1}{2}C_{load}{V_{dd}}^2. \quad (2)$$

By comparing the equations (1) and (2), it is seen that the energy stored in the capacitor is half of the total energy drawn from the supply voltage. Which means that the other half of the total energy is dissipated as heat in the p-type MOSFET as the inverter transitions from logical 0 to 1. When the output of the inverter goes back to logical 0, the stored half of the energy is dissipated to heat in the n-type MOSFET. [10 p. 13 – 14][11 p. 174]

Further modifying the equation (1), the power consumed by the switching of parasitic load capacitances can be calculated for the CMOS inverter circuit as

$$P_{switching} = \alpha f_{clk} C_{load}{V_{dd}}^2, \quad (3)$$

where f_{clk} is the clock frequency of the circuit and α is switching activity factor. The switching activity factor represents the number of transitions of each clock cycle, and can be adjusted accordingly to model a different operation state of the circuit. [11 p. 175 – 176]

2.2.2. Short-Circuit Current

The second dynamic power component is the short-circuit current. This current also occurs every time when CMOS inverter switches a state. The finite non-zero rise and fall times of the input and the output signal causes a direct current path to form between supply voltage and ground through the transistors for a short time period, as illustrated in Figure 4. The current generated by this phenomenon is referred to as the short-circuit current $I_{short\text{-}circuit}$. [10 p. 14 – 15]

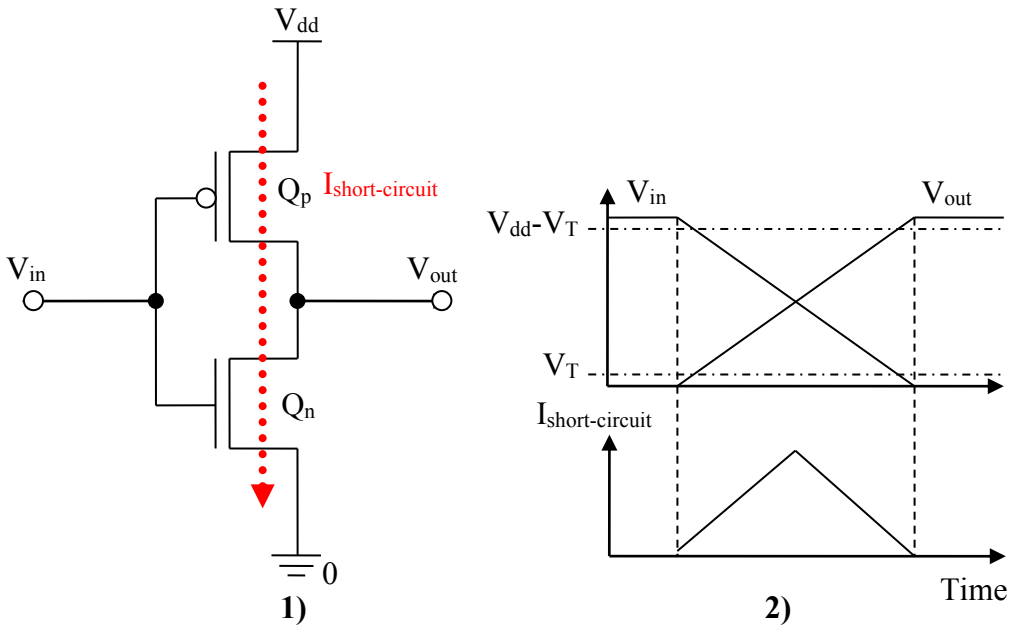

Figure 4. 1) Short-circuit current in CMOS inverter. 2) Short-circuit current in relation to transition of output and input signal.

An average value for the short-circuit current without the effects of the parasitic load capacitance can be calculated with the following equation

$$I_{short-circuit} = \frac{1}{12}\frac{\beta\tau f}{V_{dd}}(V_{dd} - 2V_T)^3, \tag{4}$$

where β is a gain factor of the MOS transistor, f is frequency of the input signal, τ is rise or fall time as they are assumed equal, V_{dd} is supply voltage and V_T is threshold voltage [12 p. 469].

With equation (4), power dissipation of the short-circuit can be calculated for the CMOS inverter circuit with the following expression

$$P_{short-circuit} = I_{short-circuit} \cdot V_{dd} = \frac{\beta \tau \alpha f_{clk}}{12} (V_{dd} - 2V_T)^3, \quad (5)$$

where f_{clk} is the clock frequency of the circuit and α is the switching activity factor similar to equation (3) above [10 p. 15][12 p. 469].

2.3. Static Power Components

Static power is always present in CMOS circuits when they are powered on. It is highly dependent on the characteristics of the target technology as well as temperature of the chip. The total average static power dissipation can be seen as a simplified expression

$$P_{static} = I_{static} \cdot V_{dd}, \quad (6)$$

where I_{static} is the current that the circuit consumes and V_{dd} is the supply voltage [11 p. 182]. For estimating the static power of a circuit, the current I_{static} can be simply measured when powered on and no switching of logic is present.

In terms of physics, the static current I_{static} is a result of the various leakage currents found inside of a MOS transistor. The static current is typically expressed as a sum of these leakage currents

$$I_{static} = I_G + I_{RB} + I_{ST} + I_{GILD} + I_{PT}, \quad (7)$$

where I_G is leakage current from gate-oxide tunneling, I_{RB} is reverse-biased diode leakage current, I_{ST} is current from subthreshold leakage, I_{PT} punchthrough leakage current and I_{GILD} is gate induced drain leakage, current as illustrated in Figure 5. Figure 5 illustrates also the basic structure of the MOS transistor, which is needed for the understanding of the mechanics behind the leakage currents. These same leakage currents can be also identified at the circuit level as the CMOS inverter example presented in Figure 6. In Figure 6, the CMOS inverter's output is at logical 1, which also illustrates that some of the leakage currents in circuit level can have a relation to the circuit's logical state as the transistors are either conducting or not. Due to this, the static power can vary between different logical states of the circuit. [9 p. 3-1 … 3-2][10 p. 15 – 17]

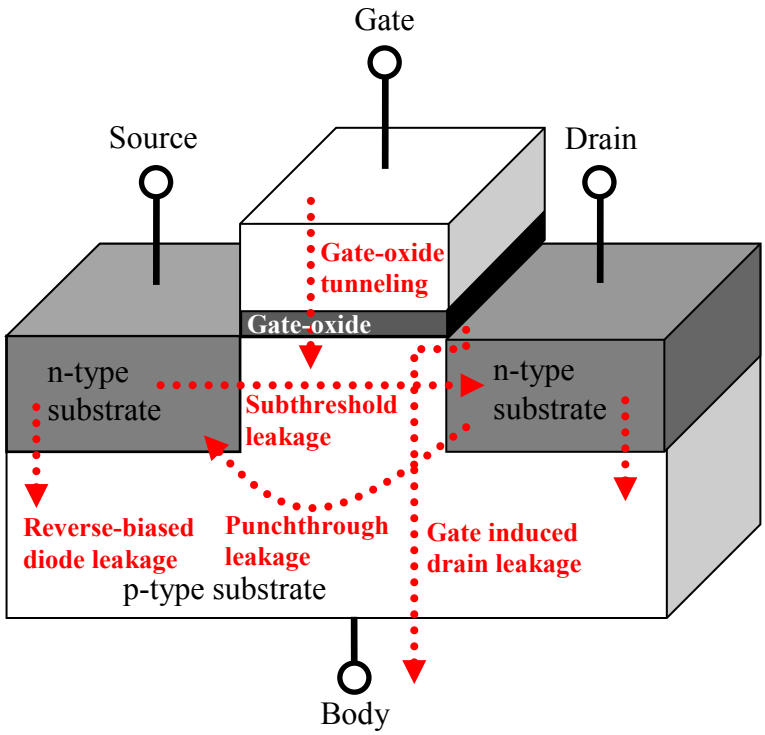

Figure 5. Leakage currents inside MOS transistor.

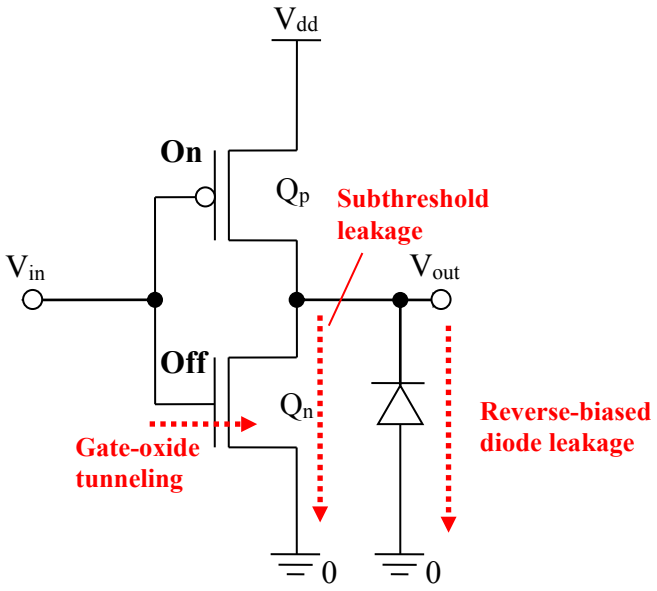

Figure 6. Leakage currents in CMOS inverter.

2.3.1. Gate-oxide tunneling

In gate-oxide tunneling, a high electric field in the thin gate-oxide insulator causes electrons to tunnel through the gate-oxide between the gate and the substrate, which generates the gate-oxide tunneling current I_G. There are two physical mechanisms

behind the tunneling phenomenon, which are called Fowler-Nordheim tunneling and direct tunneling. Direct tunneling is more dominant in nanoscale MOS transistors, where oxide layer thickness is less than 4 nm. A MOS transistor with oxide thickness above 6 nm the Fowler-Nordheim tunneling is the main tunneling mechanism. [9 p. 3-3 … 3-6][10 p. 16 – 17][13]

In Fowler-Nordheim tunneling, electrons conduct from the conduction band of the substrate through the triangular potential barrier of the gate-oxide, as shown in Figure 7. Whereas in direct tunneling, the potential barrier is trapezoidal shape and electrons tunnel directly thought the forbidden energy gap of the gate-oxide. Direct tunneling consists of three major sub-mechanisms: electron tunneling from conduction band, electron tunneling from valence band and hole tunneling from valence band. These mechanisms can be modeled and estimated by using complex models from quantum mechanics [14]. [10 p. 16 – 17][13]

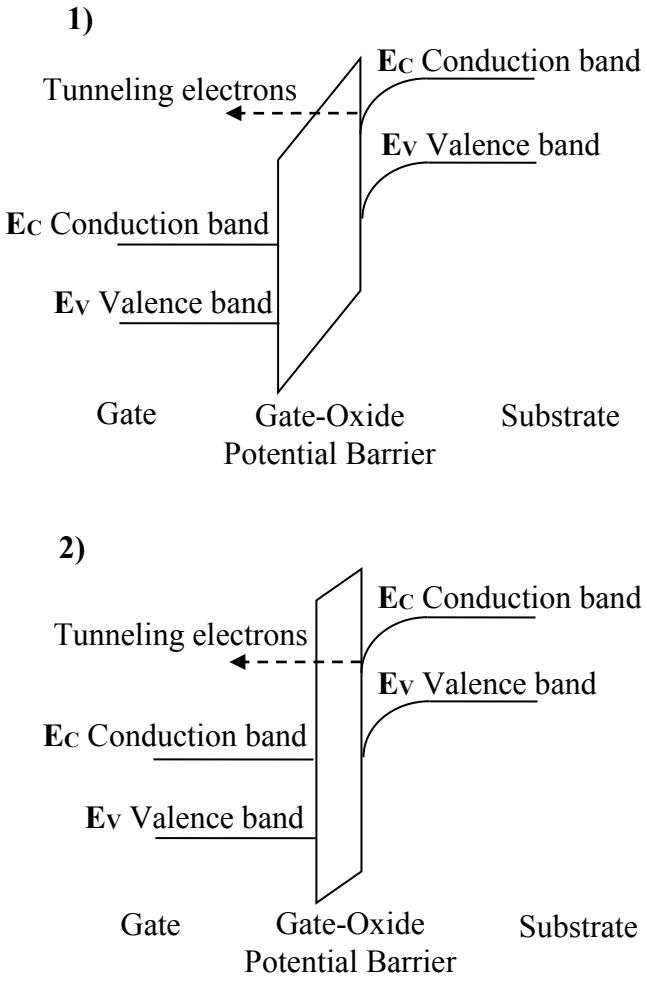

Figure 7. 1) Fowler-Nordheim tunneling. 2) Direct tunneling.

The scaling of CMOS technology decreases the physical feature size of the MOS transistors, thus decreasing the gate-oxide thickness. Thinner gate-oxide results in larger gate-oxide leakage currents, thus making the gate-oxide leakage the dominant contributor to the static power dissipation at 60 nm technologies and below [15]. [9 p. 3-2 … 3-13]

2.3.2. Reverse-Biased Leakage

There is a pn-junction between drain and body and between source and body in the MOS transistor, as in Figure 5 above. When these pn-junctions are reverse biased, there is a reverse-biased leakage current I_{rbias} flowing through the pn-junctions. In circuit-level, this means that there is a parasitic transistor towards the lower voltage potential, as shown in Figure 6 found above. [10 p. 15]

Behind the reverse-biased leakage current, there are two main mechanics: minority carrier diffusion near the edge of the depletion region and thermal generation of electron hole pairs in the depletion region. The reverse-biased leakage current increases exponentially with junction temperature. In heavily doped nanometric MOS transistors, band-to-band tunneling causes a significant contribution to the reverse-biased leakage current. In band-to-band tunneling, the electrons tunnel from the valence band of the p-type substrate to the conduction band of the n-type substrate. [10 p. 15][16]

2.3.3. Subthreshold Leakage

Subthreshold leakage current flows between source and drain of MOSFET when the gate voltage is below the threshold voltage V_T and there is a voltage difference between the drain and the source. This is due to the so-called weak inversion effect, where carriers by diffusion move through the surface of the p-type substrate below the gate-oxide in a similar way as the charge transport effect in bipolar transistors. The subthreshold leakage current correlates exponentially with temperature. [10 p. 17][16 p. 307 – 308]

In short-channel MOS transistors, the source and the drain region overlap slightly the substrate under the gate-oxide. In such devices, the subthreshold current is mainly due to drain-induced barrier lowering, where the threshold voltage vary with the drain voltage enabling the source injecting carries in to the channel formed inside of the substrate.[9 p. 3-6 … 3-7][10][16]

Scaling of CMOS technology is problematic towards the subthreshold leakage, as thinner gate-oxides require lower supply voltages. And lower supply voltages require lower threshold voltages in order not to decrease the timing performance of the circuit, but lowering threshold voltage increases significantly the subthreshold current. [9 p. 3-11]

2.3.4. Gate Induced Drain Leakage

In MOS transistors, gate induced drain leakage currents flow from the drain to the p-type body substrate, when using a relatively high supply voltage. In the region where the gate slightly overlaps the n-type region of the drain, a high electric field is formed between drain and the substrate, which causes gate induced drain leakage. There are several effects that contribute to gate induced drain leakage current, but in today's short-channel MOS transistors, the band-to-band tunneling effect is the most dominant one. In the band-to-band tunneling, electrons directly tunnel from the valence band of the substrate into the conduction band of the drain in the region where the gate and the drain overlaps. Other mechanisms include electron-hole pair generation and trap-assisted tunneling. [9 p. 3-9 … 3-10][10][16]

2.3.5. Punchthrough Leakage

Punchthrough leakage current flows from the drain through the substrate into the source. This forms a parasitic bipolar transistor inside the MOS transistor, whereas source represents emitter, drain is collector and substrate is base. The parasitic bipolar transistor is formed, when the drain voltage is high enough to create a depletion zone into the substrate, which lowers the potential barrier between the source and the substrate generating the punchthrough current. This leakage mechanism is rather negligible in today's nanoscale MOS transistors [9 p. 3-2, 3-11]

2.4. Summary

Power dissipation in CMOS circuits consists of many components created by various physical phenomena, which are hierarchically illustrated in Figure 8. These power components can be compressed into an expression such as

$$\begin{aligned} P_{average} &= P_{dynamic} + P_{static} = P_{switching} + P_{short-circuit} + P_{leakage} \\ &= \alpha f_{clk} C_{load} {V_{dd}}^2 + \frac{\beta \tau \alpha f_{clk}}{12} (V_{dd} - 2V_T)^3 + V_{dd} \cdot (I_G + I_{RB} \\ &\quad + I_{ST} + I_{GILD} + I_{PT}) \end{aligned} \tag{8}$$

in order to estimate the average power dissipation of the example CMOS inverter circuit. As illustrated by equation (8), the task of accurately estimating the power dissipation of a CMOS circuit is not an easy one as many technology and temperature related factors must be taken into account. Typically, the static power is simplified into a single current value that is defined for the target circuit technology.

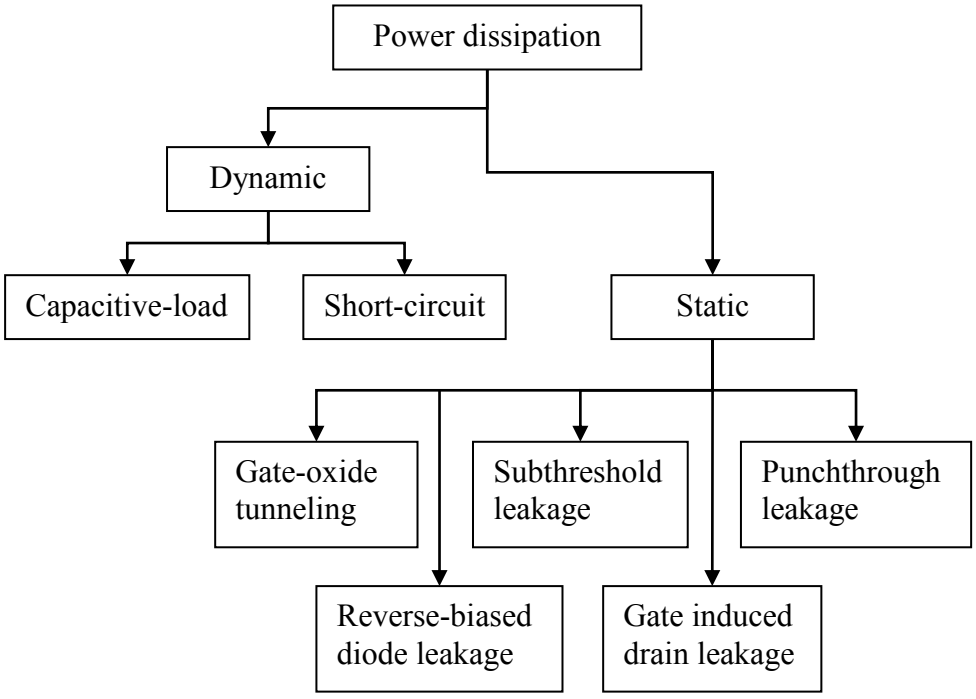

Figure 8. Power dissipation components in CMOS circuits.

Trends in CMOS Power

Traditionally, the dynamic power has been solely defining the total power dissipation of a CMOS circuit, but the static power has been exponentially rising with the scaling down of the technology. This is especially visible in technologies under 100 nm, as an example for certain 65 nm technologies over 50 % of the total power dissipation is caused by the static power dissipation [17]. In CMOS technology, there is a connection between technology scaling and leakage power, which increases leakage currents every time when the technology is scaled down [10 p. 19]. Increasing leakage currents increase the temperature of the chip, thus increasing leakage currents even more, which makes a complete feedback loop that results in an increase of the total power density of a chip [10 p. 19].

New technologies, such as FinFET technologies, have been developed to tackle the problem of the static power dissipation in the current CMOS technologies. In FinFET technology, the transistors are multi-gate 3D structures that rise above the planar substrate, thus giving them more volume with the same surface area used for a single gate. FinFET is based on a double-gate MOSFET with a similar structure rotated and 3D structure added, as illustrated in Figure 9 below [18]. The 3D structure provides better electrical control of the effective channel formed in the transistor, which helps to overcome the short channel effects and reduce the leakage mechanism significantly. This 3D structure wrapped around the channel is called as a fin, where the name FinFET comes. As of today, the FinFET technology seems to be the choice for CMOS technologies below 20 nm. Compared to standard planar MOSFET technology, FinFETs are estimated to reduce static power by 90 % and dynamic power by up to 50 % with 37 % performance increase [18]. All this gain is estimated to come with a 5 % production cost increase [18].

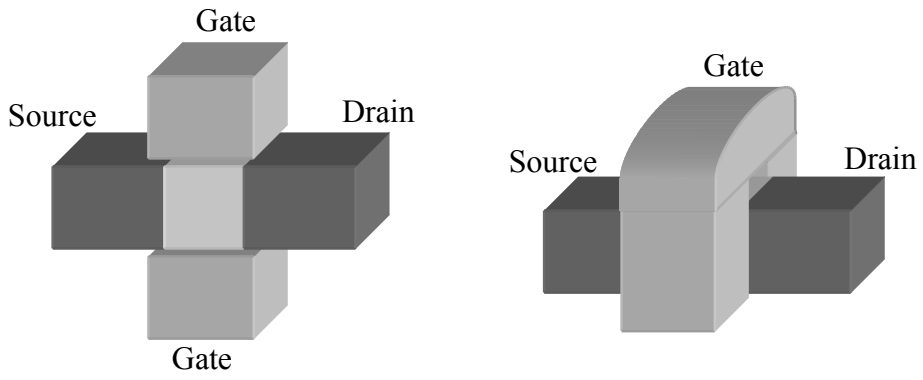

Figure 9. 1) Planar double-gate FET. 2) FinFET.

Future projection of a consumer SoC power consumption trends by International Technology Roadmap for Semiconductors (ITRS) show that the dynamic-static power ratio is developing towards dynamic power dominating again in the future [19]. The projected ratio for 2015 is 2/3 of the total power dissipation is dynamic and 1/3 is static and 2025 3/4 will be dynamic and 1/4 static power [19]. This is somewhat in line with the estimates of the effects by FinFET technologies to the power dissipation.

3. RTL POWER ESTIMATION METHODS

This chapter gives an overview of the common methodologies behind the different type of power estimators typically used in the register-transfer level. The first sections gives an introduction to power estimation, and the following three sections describe the three types of power estimators used in register-transfer level.

3.1. Introduction to Power Estimation

It is good practice to make a distinction between the terms power analysis and power estimation, as it is common to see that they are thought to mean the same thing, which is not true. The term power estimation, refers to the estimating the power with incomplete information about the design. And the term power analysis, in its turn, refers to the analysis of power when there is a full existing design, i.e. physical layout of the circuit, is available. [9 p. 38-2]

In today's SoC design, continuous power estimation and analysis are necessary in every abstraction level of the design, in order to meet the competitive low-power, performance and time-to-market requirements. A variety of different types of power estimation and analysis methods have been developed to the different design abstraction levels, all the way from simulating physical circuit models to estimating power early with spreadsheet calculations. Typically, power estimation accuracy increases when moving down on the design abstraction levels as there are more and more information available, as illustrated in Figure 10 [20 p. 5]. This also slows down the simulation speed in the lower abstraction levels, thus increasing design iteration times. This increases the need of accurate high-level power estimation for performing power exploring and debugging without any major increase of the time-to-market time and the design cost. [20 p. 5]

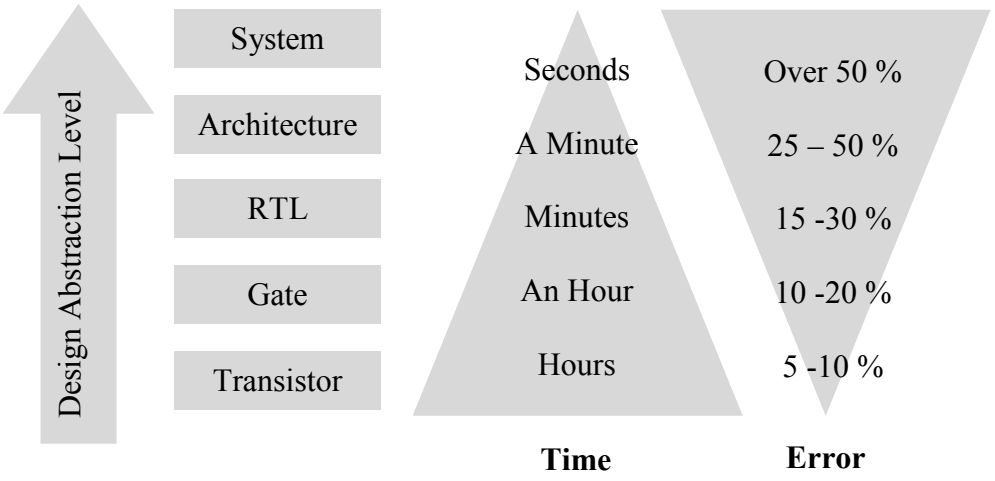

Figure 10. Power estimation speed and error in different abstraction levels of a typical SoC design flow.

As discussed in Chapter 2, the overall power dissipation of a digital circuit tends to be highly dependent on the dynamic power, which highly correlates with the switching activity of the logic. Considering power estimation, this makes it critical to

model the switching activity in a realistic manner in order to achieve accurate and realistic estimation results. As it happens, typically power estimation methods are classified based on the generation method of the switching activity into static, dynamic and hybrid methods. [20 p. 6][21 p. 4935]

The static methods are non-simulative estimation methods, where the power estimation is done without simulation activity. Hence, the switching activity is generated with initial input values and propagated through the circuit. In the static estimation methods, there might not even be a description of the circuit. In that case, the power estimation is typically based on analytical models of the circuit based on the complexity information available for the circuit. The static estimation methods are more suited for power estimation in higher design abstraction levels as they do not require much information as an input. [20 p. 6][21 p. 4935]

The dynamic estimation methods are simulative methods, where the switching activity data is acquired straight from the simulation of the circuit. The switching activity data is either used directly or statistically sampled stimulus to record the power dissipation with a power model of the circuit. The dynamic methods are more common in the lower design abstraction level, where complete models of the circuit exist that can be directly simulated. The dynamic power estimation methods are also more accurate, but they are slow to run and computationally intensive compared to statistical methods. Hybrid methods try to combine the approach from the both of these methods and can usually utilize a high-level simulation with low-level block models of the circuit. [20 p. 6][21 p. 4935]

In register-transfer level, typically there are HDL descriptions of the functional blocks of the design available, but information about gate, circuit and layout level details may not exist at this point of the design flow. This is a major challenge for RT-level power estimation, as how something can be accurately modeled and estimated, if a large portion of the required information is not available at the time being. For an accurate RTL power estimation, it is required to have power model libraries that contain power models, or power description, for each functional RTL block. For example, these power models represent an arithmetic block as a single functional block rather than separate gates, as Figure 11 illustrates, the fundamental difference between the gate-level and RT-level.

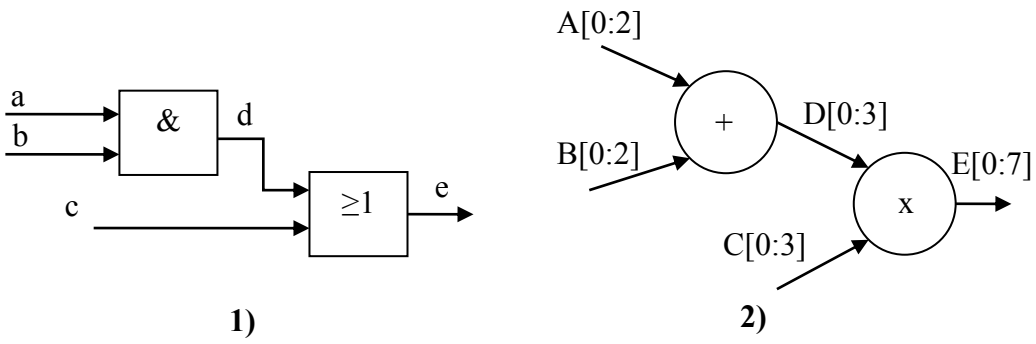

Figure 11. 1) Gate-level model. 2) RT-level model.

The main power modeling methods in register-transfer level can be categorized as analytical, macro-modeling and fast synthesis based methods. The background

behind these estimation methods is described in the following sections below. [22 p. 431 – 432]

3.2. Analytical Methods

Analytical methods are used for estimating power of a design by modeling the physical features of the design that contribute to the total power consumption. These physical methods are such as they were discussed in Chapter 2 above. Often, these estimators are focused on modeling the switching of the physical parasitic capacitances of the circuit as they contribute a large proportion of the total power dissipation and static power can be simply estimated as a pre-evaluated number. The analytical estimators estimate power consumption by considering complexity and average switching activity of the design. These methods require very little information as input and do not require computationally intensive calculations or direct simulations. As for a down side, the power estimates of these techniques may not correlate well with the power consumption of the end product. They are more suited for design power exploration, where complete description of the design does not exist yet. The analytical power estimation methods typically fall under the static estimator class as they do not require simulation. [22 p. 431 – 432][23 p. 30]

3.2.1. Complexity-Based Methods

Complexity-based estimators estimate power of a design by performing a numeric modeling for the complexity of a design. The complexity for a design is typically approximated with gate equivalents, i.e. the number of reference gates that is needed to implement the design. 2-input NAND logic gates are typically used as reference gates. Most design libraries typically provide the number of 2-input NAND gates needed for a specific function, for example, certain width of a multiplier or an adder. The total power required by the design can be then estimated straightforwardly by calculating the average power dissipated by each gate with a simple expression such as [24 p. 150]

$$P = N_{Gate}(E_{aver} + C_L {V_{dd}}^2) f_{clk} \alpha, \tag{9}$$

where N_{Gate} is the number of gate equivalents, E_{aver} is the average energy dissipated by a single reference gate, C_L is the parasitic load capacitance, V_{dd} is the supply voltage, f_{lck} is the clock frequency and α is the switching activity. [23 p. 30 – 31]

The obvious disadvantage of complexity-based estimators is that they consider the switching activity of a circuit as a fixed static factor. In real life, the switching activity is much more dynamic as when the data is processed and propagated within the circuit, thus making the power estimation results inaccurate and non-related to the physical hardware. Also, it's a limitation if the power estimation is based on the power dissipation of a single equivalent gate, since multiple different logic technologies may exist in a single chip. [23 p. 30 – 31]

3.2.2. Activity-Based Methods

To address the problem of fixed switching activity, more complex analytical power estimation methods have been developed that include techniques to model the activity in a more dynamic fashion. These methods often use probabilistic methods and the information theory. For example, entropy from the information theory can be used as a measure of computational work, as a work in [25] summarized by [23]. In this model, power is estimated as proportional to the product of the load capacitance and switching activity, which is then proportional to the product of area and entropy of inputs and outputs [23 p. 31]

$$P \propto Capacitance \cdot Activity \propto Area \cdot Entropy. \tag{10}$$

This method requires an RTL simulation in order to determine the input and output entropies, as well as knowledge of the average load capacity of a single node. [21 p. 4935][23 p. 31]

3.3. Macro-Modeling Methods

In macro-modeling based power estimation methods, power estimation for an RTL design description is made by using separate macro-models for each functional block in the design. A macro-model models the power dissipation of a functional block as a function or a look-up table with variables, for example, in terms of power dissipation in relation to input and output signal statistics. In empirical macro-modeling methods, the parameterization is performed at gate-level or layout-level for similar design implementations in order to increase estimation accuracy in register-transfer level. In essence, the power macro-models try to model the high-level power consumption by characterizing the lower-level implementations of the corresponding design blocks. As such, power macro-modeling requires a characterization phase before the actual power estimation step. In the characterization phase, power macro-models are constructed for each functional component within the design RTL libraries. The power macro-modeling can be average power estimation or cycle accurate power estimation. In cycle accurate power estimation, the power estimation is done in the time domain which requires input data from simulation in order to get accurate results, while the average estimation can be typically performed without any input data. [22 p. 39]

3.3.1. Basic Concepts of Power Macro-Modeling

In order to understand the basic concepts of power macro-modeling, it is suitable to consider five different issues related to the power macro-modeling scheme. The building blocks that the power models are built from are vital consideration as they define the granularity of the whole macro-model. The choice of parameters used inside of the macro-models is critical as they characterize and capture the power dissipation. The data collection method for collecting data from the power models determines how the power macro-model can be used. The construction method of the

power macro-model for an RTL design defines how and when the model can be used in a design flow. [9 p. 39-2]

Building Blocks

In an ideal power macro-modeling scheme, there should be a single power model for each RTL design block and these power macro-models should be based on a single architectural template. A finite state machine with datapath (FSMD) has been suggested for such an architectural template. The FSMD is constructed with a control and a datapath section. The control part contains a state machine for controlling the datapath, and the datapath has various registers, multipliers, adders and other logic blocks for implementing a specific functionality. The idea in using of FSMDs is that any RTL description can be viewed as the interactions of datapaths and controllers for which the FSMD can be configured for. Using a single component as a template can be too ideal and inflexible for real life settings increasing the error of the estimation. [9 p. 39-2 … 39-3]

Parameters

A power model contains a set of parameters that are used to capture the power dissipation of a design block. One example of a high-level power dissipation equation is a sum of parameters as [9 p. 39-4]

$$P = k \sum_i A_i C_i, \tag{11}$$

which represents the dynamic power dissipation equation (3) in Chapter 2, where k is the variable that represents the term fV_{dd}^2, A_i is the switching activity, C_i is the load capacitance and i is the component block from the power model. From this equation, it is seen that the power model should model at least two parameters in register-transfer level, which are activity and capacitance or complexity parameter. Activity parameters represent the switching activity of the components found inside the power models and are straightforward to model in register-transfer level as RTL simulations typically monitor activity with clock cycle accurately. Or alternatively, the switching activity can be also modeled as probabilities of signal states over time. Modeling the load capacitance is much more complex as it is related to the target technology and it is often measured as the complexity of the component. [9 p. 39-3 … 39-5]

Data Collecting

The models for collecting the data returned by the power models can be either average or cycle-accurate models. Equation (11) is one example how average power data can be collected from the power models. For collecting cycle-accurate data, equation (11) can be modified by adding one dimension to the sum as such [9 p. 39-5]

$$P_{total} = k \sum_i A_i C_i = k \sum_j \sum_i A_{ij} C_{ij}, \tag{12}$$

where *j* represents the current cycle index. With this equation, the total power can be traced cycle-accurately over time, which increases the power analysis capabilities for example in terms of analyzing peak power. [9 p. 39-5 … 39-6]

Constructing

A power macro-model can be constructed analytically top-down or empirically bottom-up. Analytical methods model the parameters through formulas and derive the power model directly from the RTL design description. As the analytical models do not require information of previous low-level implementations, they can be used when the design is new and in early stages of the design flow when such information does not exist yet. On the other hand, this approach may not be very accurate as the power model may not relate very well to the end hardware. [9 p. 39-6]

Empirical methods are much more accurate power estimation methods as the power models are based on similar previous implementations. In RT-level macro-modeling, the low-level information is typically derived from similar gate-level implementations, where the power dissipation of a certain component is accurately modeled and statistically analyzed into set of parameters. [9 p. 39-6 … 39-7]

The power models can typically be an equation-based or look-up table based method. Typically, analytical methods are based on equations and empirical methods can be either equation-based or LUT-based. [9 p. 39-7]

3.3.2. A Generic Power Macro-Modeling Flow

RTL power macro-modeling flow can be split into a set of two flows. The first one is characterization flow, which is used to construct the power models. The second one is power estimation flow, which illustrates a generic estimation process and where the characterization flow is a preliminary step before the power estimation.

Characterization Flow

The constructing flow of the power macro-models is called a characterization flow. A generic characterization flow consists out of four stages, which can be illustrated with mathematical expressions as output values [9 p. 39-10 … 39-11]:

1. **Parameters X_1, …, X_n:** Defining of the parameter space for the power models that capture the power dissipation.

2. **Vectors W_1, …, W_n:** Defining all of the used input vectors, i.e. the training set. The key decisions here are the size of the training set that impacts on the simulation times and the statistical distribution in relation to the parameters defined in the previous step.

3. **Samples $P(W_1)$, …, $P(W_n)$:** Characterization of the power model, where the vectors are mapped with corresponding power values. Characterization is made typically with power information from gate-level or post-layout-level models in order to obtain accurate power values.

4. **Models $P = P(X_1, \ldots, X_n)$:** Building the power models from the samples of power obtained in the previous step and storing them either into equation-based or LUT-based models.

Power Estimation Flow

In a generic RTL macro-modeling power estimation flow, three basic stages are performed [9 p. 39-12]:

1. Identification of each component in the architecture templates.

2. RTL simulation for tracing signal activity for the power models built from the architecture templates.

3. Perform power estimation.
    a. Characterization phase: Construction of power macro-models for each component with the characterization flow from above.
    b. Power estimation: Collect and report power dissipation from the power models with using trace from an RTL simulation.

3.3.3. Power Macro-Modeling in Real-Life Applications

The assumption that any RTL design can be modelled only by using a single architectural template is too ideal and inflexible for real-life power macro-modeling. To overcome this problem, real-life examples of RTL power macro-modeling methods utilize a number of fine-grain models called power primitives rather than using a single model. For example, these power primitives are separate logic gates, multiplexers, flip-flops and memories. This brings the granularity of the macro-models closer to the RT-level and the model can estimate the impact of an RTL synthesis better. The power primitives can be constructed with a synthesis to the target technology library or specific power models can be developed and stored beforehand to avoid performing of synthesis. [9 p. 39-13][26][27]

In an example of a commercial RTL power macro-modeling, the power models consists of RTL operators as register, adders, multipliers and multiplexers [9 p. 42-6]. The architecture of this type of power macro-modeling application is illustrated in Figure 12 [9 p. 42-5]. The RTL design is inferred to these components by an RTL inference engine, which produces a micro-architectural netlist that is stored in an internal data base. This step is very similar to what happens first in a logic synthesizer, but instead of producing a full gate-level netlist, the micro-architectural netlist contains technology as unmapped. The power calculation engine requires a data base that contains switching activity data from RTL simulation and power-characterized technology libraries that describe the electrical characteristics of the target technology. For each inferred component, the tool has a predefined power model that is characterized with the electrical characteristics from the technology libraries. This creates a unique power equation for each inferred component. With the stimulus data from the simulation trace and power equations, the power calculation engine can calculate the power of each inferred component and report the power of various design units. [9 p. 42-5 – 42-6]

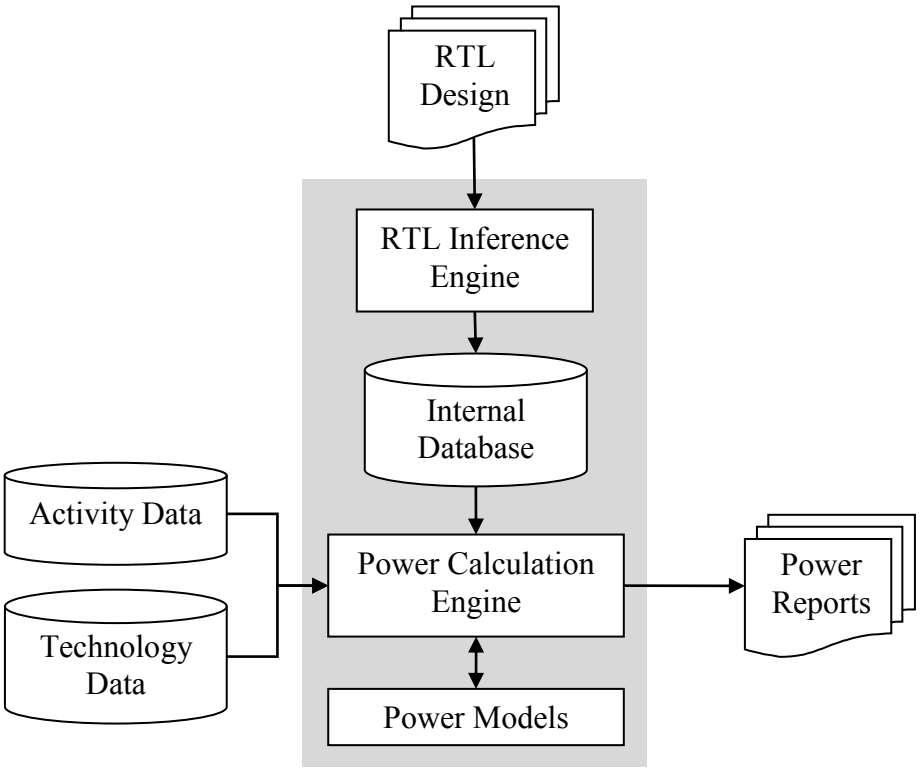

Figure 12. A commercial RTL power estimation application.

3.4. Fast Synthesis Methods

The third common method for power estimation in the register-transfer level is fast synthesis methods. In fast synthesis methods, a limited low-effort RTL synthesis is performed which is significantly faster than a normal logic synthesis. The fast synthesis generates a low-effort netlist suitable to be used in a gate-level power simulator for power estimation with switching activity data provided by simulation or generated by some pre-defined stimulus. Rather than producing a fully optimized gate-level netlist as in full-blown synthesis, the fast synthesis procedure refers the design into block level components, which have cell power information and models available in technology libraries. Also, the number of cells used in the synthesis process can be limited in order to speed up the process and to maintain the accuracy of power estimation. These methods try to reduce the problem of power modeling into a problem of estimating the effects of logic synthesis. The fast synthesis methods offer good estimation accuracy, but much slower runtimes compared to analytical and macro-model methods. One key advance over the macro-modeling methods is that fast synthesis methods do not require a separate power model library as they operate solely with the standard technology libraries. [20 p. 7][26][27]

After the netlist is obtained via the fast synthesis, the fast synthesis methods rely on similar circuit-level simulators for calculating power as in gate-level estimation methods. Rather than having the design partitioned into nets connected to gate components, the design is now partitioned into nets connected to block-level components. The circuit-level simulators need information about the static power and internal power for each component in the netlist, as well as knowledge of load

capacitances and output and input pin capacitances. Typically, static and internal power data is available in the libraries as look-up tables for each cell, as illustrated in Figure 13. The look-up tables should have the relation of input data pattern, delay and dynamic power for each component. As well as for static power calculations, it is required to have information of static power in relation to input data pattern. In order to accurately model the internal power, precise timing models are needed, as internal power depends highly on the timing. The rise and fall times of the signals determine how long the transistors are short-circuited during a state transition, thus forming the short-circuit current, as discussed in section 2.2.2. [28 p. 4, 6, 7]

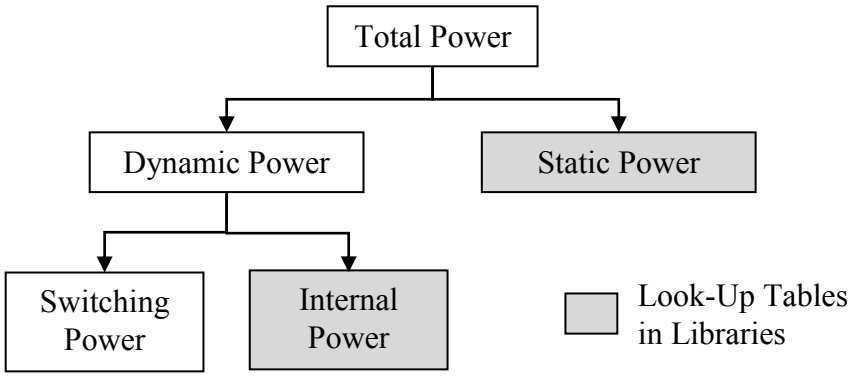

Figure 13. Power information required from the libraries.

The calculation of switching power requires the knowledge of the load capacitances for each net in the design. This information can be provided by wire load model (WLM) or detailed parasitic data extracted from a layout analysis. The data from layout analysis is accurate, but it requires data from the layout of the same target technology to be available. For a new design, such data required for the calibration may not yet exist in RT-level design phase. WLMs are an approximation of the net load capacitances, which are less accurate, but are available pre-hand for the target technology and do not require analysis of the layout. [20 p. 7][28 p. 12]

Power Calculation via simulation

Power is calculated by running an event-driven logic simulation with the set of input vectors obtained from an actual simulation. During the simulation, the power estimation engine monitors the toggle count and the state of each net in the design to calculate the switching, internal and static power components. For switching power, the toggle count of each net is monitored, as [28 p. 15]

$$P_{switching} = \sum_k C_k V^2 N_k, \quad (13)$$

where k is the index number for the different nodes in the circuit, C is the capacity of net k, V is the supply voltage of the circuit and N is the number of toggles per time unit in the net k. Internal power is calculated such as [28 p. 16]

$$P_{internal} = \sum_c \sum_e E(c,e)F(c,e), \quad (14)$$

where c is the number of components in the design, s is the states of component c, $E(c,s)$ is the energy of event e for component c and $F(c,s)$ is the monitored occurrence frequency of the event e for component c. The energy E is precomputed into look-up table for each component and frequency F monitored in the simulation. The static power is simply calculated by multiplication of static power dissipated and time spent in specific states, as [28 p. 17]

$$P_{static} = \sum_c \sum_s P(c,s)T(c,s)\,/T, \quad (15)$$

where $P(c,s)$ is the static power of a component in state s, $T(c,s)$ is the time spent in the state s for component c and T is the input vector period or simulation time window. Then the total power can be simply obtained by summing these three components, as [28 p. 18]

$$P_{total} = P_{switching} + P_{internal} + P_{static}. \quad (16)$$

4. COMMON POWER REDUCTION METHODS

This chapter gives an overview of the common power reduction that an RTL designer should consider and be aware of. The chapter is divided into two major sections, the first one discusses about dynamic power reduction methods, and the second one about the static power reduction methods.

4.1. Introduction

The design decisions made in the high-level of design abstraction can have a huge impact on the total power dissipation of a design, while the low-level design changes have significantly smaller impact, as illustrated in Figure 14 [29 p. 97]. This means that the low-level power optimizations might not be able to overcome the power mistakes made in the higher design levels in order to stay on the power budget of the design. Such a design situation would require costly iterations back to the higher design levels to fix the power mistakes. In order to successfully reduce power of the design as much as possible and to avoid costly design iterations, power optimizations should be made continuously on all design abstraction levels. [29 pp. 96 - 98].

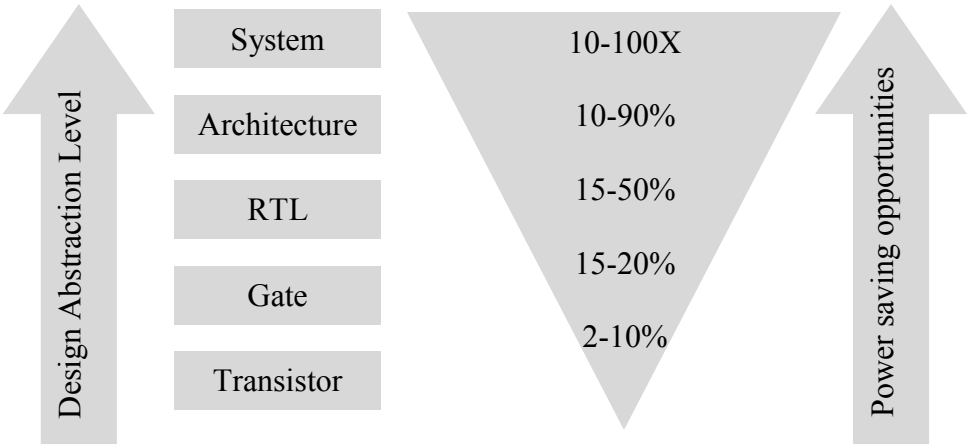

Figure 14. Potential power reduction opportunities by design abstraction levels.

As the power dissipation of a CMOS circuit can be divided into a dynamic and static power component, it is practical also to categorize the power reduction methods into dynamic and static categories. Classification by design abstraction level is far more challenging, as many of the reduction methods span over multiple design levels highly impacting the design considerations. This reflects well to real life, where power reduction should be considered and performed in every design abstraction level and step. Successful low-power design also requires a combination of multiple reduction methods.

4.2. Dynamic Power Reduction Methods

As a large proportion of the dynamic power dissipation of a CMOS circuit comes from charging and discharging the parasitic load capacitances, dynamic power

reduction methods concentrate on limiting the contributing factors in the equation (3) from section 2.2.1. These factors are switching activity, clock frequency, supply voltage and parasitic load capacitances. The parasitic load capacitance is very dependent on the chosen target technology and cannot be affected with design methods other than optimizing logic structures in order to minimize input and output capacitances seen by each logic cell. Most of the dynamic power reduction methods try to reduce the switching activity by gating data and clock signals.

4.2.1. Clock Gating

Clock networks can be responsible over for 50 % of the total power dissipation in a digital circuit [30 p. 1], as they typically are the most active and largest nets found in digital circuits. Therefore, clock gating is the most commonly utilized and researched method for reducing power dissipation, and it is available as an automated design feature in modern EDA tools. The main idea of the clock gating is to eliminate unnecessary input transition generated by a clock signal. In order to gate clocks, special clock gates are inserted to disable the clock signal when possible. For a clock tree, it can be done hierarchically and in multiple stages to disable parts of the clock network. Shutting down parts of the clock network not only reduces power wasted by the clock buffers, but it will also reduce the dynamic power dissipation of cells connected to the clock network. Furthermore, an entire design block can be gated as a coarse-grain gating method. In fine-grain clock gating, single registers or register banks are gated separately rather than gating entire design blocks. [31]

A typical clock gating scheme is presented in Figure 15. In Figure 15 1), there is a flip-flop with an enable signal that synthesis tools will typically produce. This typically requires an enable statement for a register to be coded in the HDL description. The enable signal to the flip-flop is fed through a multiplexer with a feedback loop from the output signal of the flip-flop. Implementing clock gating into the same circuit removes the output signal feedback loop and replaces the multiplexer with a clock gate, as illustrated in Figure 15 2). A clock gate is typically implemented with a latch and an AND gate, as in Figure 15 2). Figure 15 3) illustrates the timing of the clock gating signals, when the enable signal is logical 0, the clock going to the clock input of the register is gated off. This approach eliminates possible glitches that may occur in latch-free clock gates that are constructed only from logic gates. One can also use flip-flops for implementing clock gating which offers better testability, but this approach consumes more area and power than latch-based and logic gate based implementations decreasing achieved power reduction. [30]

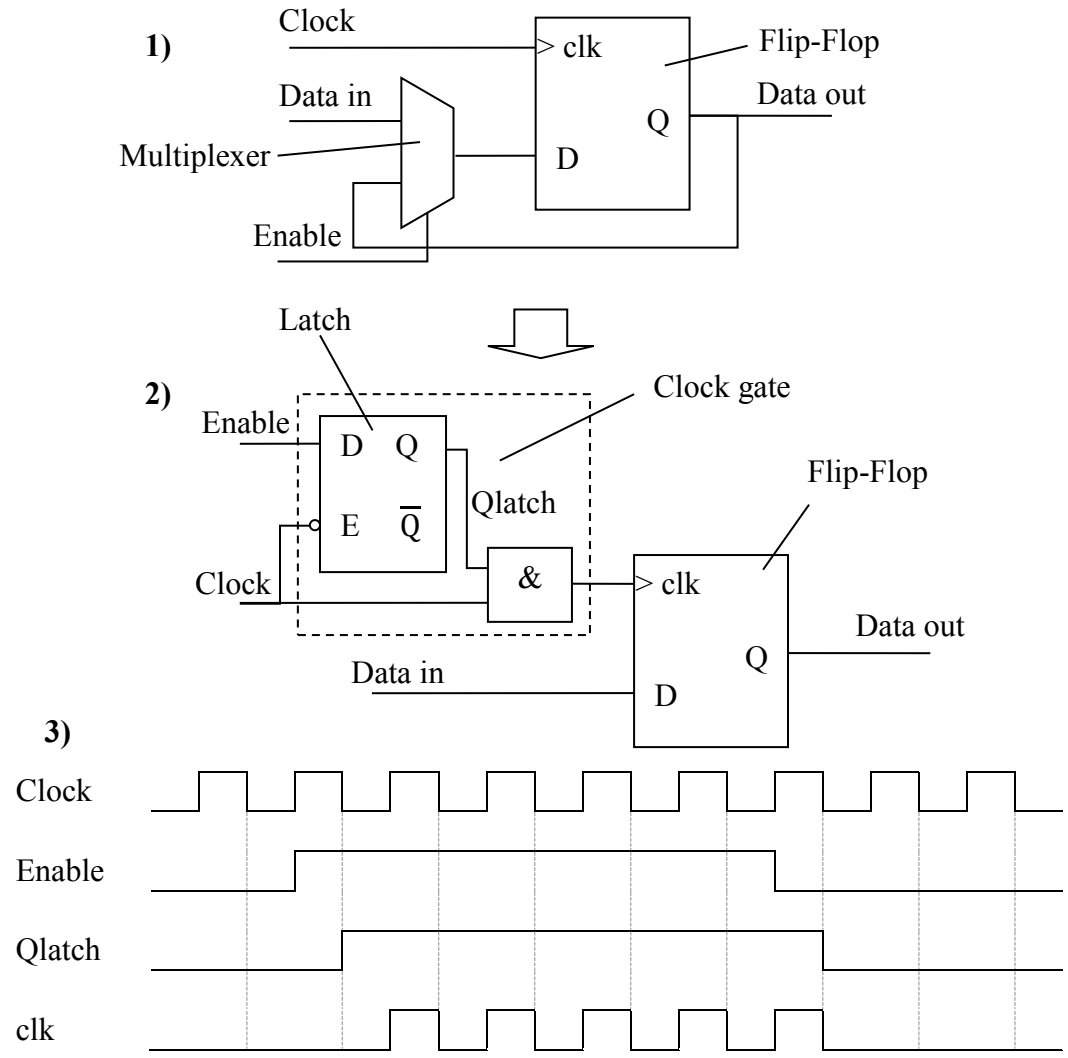

Figure 15. 1) Typical sequential circuit with enable signal. 2) Same circuit clock gated. 3) Timing diagram of the clock gating signals.

There are multiple challenges related to implementing the clock gating successfully, due to the fact that the clock is a critical construct for the synchronous digital circuits. Ideally, in synchronous circuits, the clock signal is distributed unchanged to every location of the circuit. This means that adding clock gates should not alter the waveform, generate glitches or degrade timing of the clock. If clock gating is not implemented properly, the additional logic for generating clock gating can consume more power than what the achieved power savings are [32 p. 176]. This makes the decision critical as to where to insert clock gates and how to control them. Another challenge for clock gating is how to implement testing and verification for clock gating. [30 p. 3]

The efficiency of clock gating is highly dependent on the quality of its enable signal, i.e. how well the redundant clock cycles are gated with the enable signal. It also does not make sense to insert a clock gate for such a register that has constantly data flowing through, as the clock cannot be ever gated off. Inserting a clock gate in such a scenario would only increase the power dissipation, as well as the area of the

design. As such, a simulation is required for a high clock gating efficiency, and it is nowadays a typical feature in advanced clock gating tools.

The results in [34 p. 404] demonstrate typical and realistic effects of clock gating, which show about average 20 % power saving and 3 % area decrease. Clock gating may result in a decreased area in some cases as it can remove the feedback loops from the registers, which reduces wiring and routing overhead reducing total area.

Register XOR Self-Gating

In XOR self-gating, the enable signal for the clock gate is generated routing the register's data input and output via an XOR gate, as illustrated in Figure 16 [35]. This means that the register is gated when the output and input are the same value, i.e. the input data is stable. And when the input is different from the output, the register is ungated and the input value is stored into the register. XOR self-gating is a more advanced clock gating method, and it requires switching data from simulation in order to be implemented efficiently. As with switching data, gating tools are able to calculate if the added area overhead is worth the achieved dynamic power savings for each gating opportunity. [35]

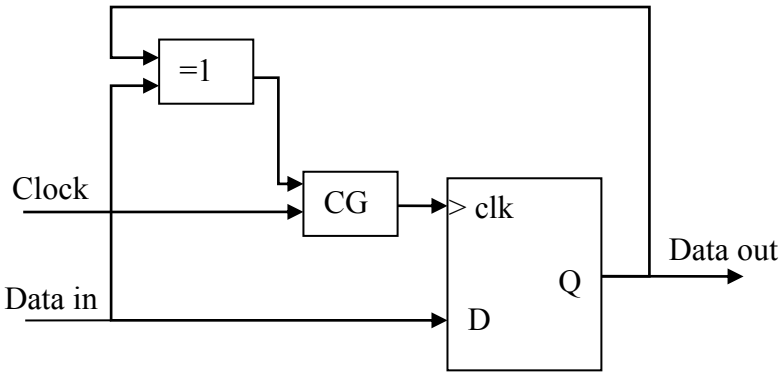

Figure 16. XOR self-gating.

Observability Don't Care Condition

Observability don't care (ODC) conditions are Boolean don't care conditions which are not propagated into downstream into a Boolean network. These ODC conditions can be computed and recognized for signals of an RTL design, and used for clock gating registers. This saves dynamic power by eliminating any unobservable value changes in the registers of an RTL design. This advanced clock gating technique can be found in modern synthesis and power tools. [36]

Stability Condition

Stability condition (STC) is a sequential condition, where the value of the current state is equal to that value of the previous state. The extracting of the STC condition is highly challenging, thus existing clock gating conditions are typically used as STC conditions in a way of enhancing the existing clock gating network in a design. This method scales well to large design and functions on an RTL design, and it can be found in modern clock gating tools. [37 p.659]

4.2.2. Datapath and Logic Optimization for Low-Power

Datapaths and logic structures can be optimized with various methods for low-power. Most of these optimization techniques presented in this section are typically implemented nowadays by logic synthesis tools. But designers should be aware of these techniques as RTL coding style affects how well the synthesis tool can optimize the datapaths. The designers have been responsible for the implementation of these techniques in the past, but modern tools can explore a different solution much faster leading to better results. [33]

Glitch Minimizing

One can reduce dynamic power consumption of a digital circuit by minimizing glitches, as glitches generate unnecessary input transitions wasting power. Glitches are typically produced by different propagation delays in converging combination logic paths [9 p. 11-4]. Minimizing glitches can become important as they can contribute up to 40 % of the total power dissipation in some designs [38 p. 1]. Results from a study in [39 p. 1129] indicate that glitch minimizing yielded impressive 23 % of average power savings in various designs.

At gate-level, glitches can be minimized with delay balancing between the different logic paths to harmonize propagation delays with buffering and gate sizing. One way to minimize glitches in register-transfer level is to add registers between combination logic function blocks, i.e. pipelining. This method usually results in increased power and area overhead because of the added registers. Also, any tight latency requirements can make it impossible to implement any kind of pipelining method effectively. Another register-transfer level method is rearranging operators in logic structures. For example, if there is parity check with a large XOR tree after an addition or a multiplication, there can be glitches from the output of the operator. These glitches can be reduced by reordering, with removing the XOR tree from the output of the operator and inserting XOR tree into both inputs of the operator. Thus, reducing the power dissipation generated by the glitches if the inputs are considered more stable and glitch-free than the output of the operator. [9 p. 11-4 … 11-5]

Operator Isolation

Operator isolation is a power saving method for operators in a datapath that are not fully utilized by the design in every clock cycle. Such operators perform operations that are not needed and are redundant, which naturally waste power. The basic idea of isolating an operator is to eliminate the unnecessary operations performed by the isolated operator. [40]

In operator isolation, an operator is isolated with an enable signal to prevent any state transition of the inputs when the operator is not needed to perform any operations. Figure 17 demonstrates a datapath without operator isolation, the multiplier operator * produces redundant operations when the result from the multiplier is not loaded to register REG_B in every clock cycle and the input signal DATA_A and DATA_B has transitions. In Figure 18, operand isolation is added to the same datapath to prevent these redundant operations. In this example, the operand isolation is implemented with addition of an enable signal for controlling the isolation and two latches, each for every input of the isolated operator. The enable

signal of the register REG_B could be also used as the operator isolation enable signal, if the datapath timing conditions allow this. Operator isolation can be also implemented with logic gates, especially with AND and OR gates. [40]

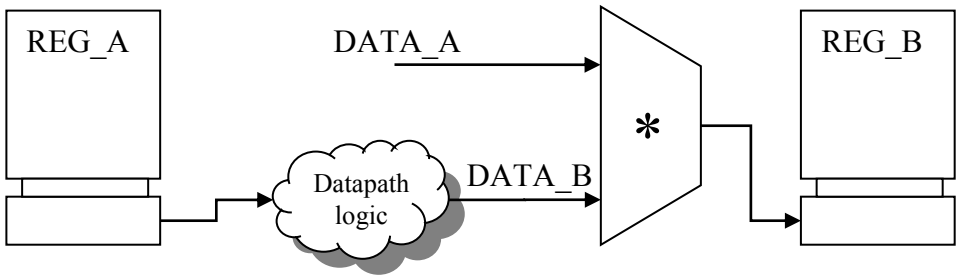

Figure 17. Datapath without operator isolation.

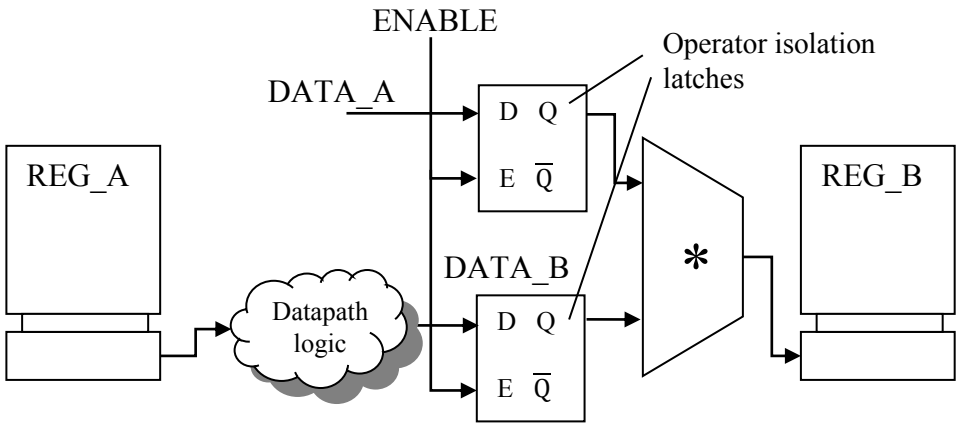

Figure 18. Datapath with operator isolation.

An experimental study in [40] shows up to 30 % power reduction and 20 % area increase with power isolation techniques. It should be noted that this power saving technique is a very case dependent one and in some cases there might not be much power savings at all, because the power savings achieved with operator isolation are proportional to the redundant operations in the datapath. The key for successful power reduction with operand isolation is the identification of redundant operations in datapaths of the design.

Operator isolation can be expanded to cover the isolation of the whole datapath and have a centralized control unit that isolates parts of a datapath in order to optimize power consumption by removing redundant switching from the datapath. This technique is also known as datapath pipelining. [9 p. 11-17 … 11-18]

Pre-Computation

Pre-computation is suitable for the inputs of a combinational logic block, in such a case that some of the inputs values can be predicted before they are required. This reduces switching activity, thus saving power. To generate further power savings, the logic that is generating the original values can be turned off. Figure 19 illustrates the pre-computation scheme [41 p. 75]: the original circuit has a number of inputs that

some of the next states can be easily pre-computed from previous states. The inputs are divided into pre-computation inputs that are used to identify pre-computations and gated inputs that are the inputs which values will be pre-computed. The control logic performs the pre-computation and gates the inputs when needed. The design of the control logic is critical for this scheme as control logic increases power consumption and area, thus control logic should not dissipate more power than pre-computation reduces. [41 p. 74]

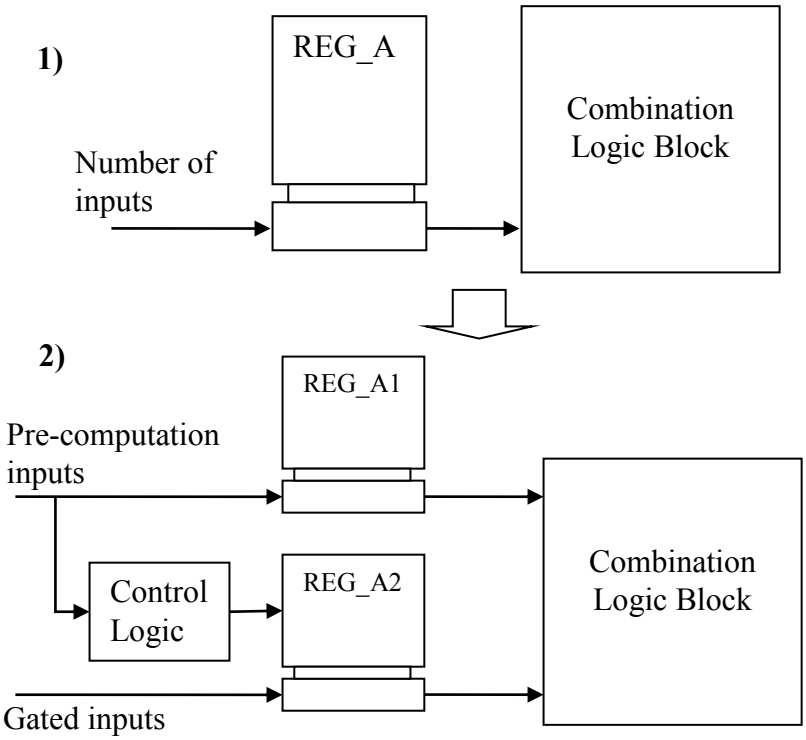

Figure 19. 1) A circuit without pre-computation. 2) Pre-computation scheme.

Other Common Datapath Operator Power Optimizations

There are multiple commonly used techniques for synthesis engines to reduce power of operators inside of a datapath. A common method is to minimize the number of operators by minimizing the number of high level operations needed for the arithmetic functions performed in the datapath. This not only reduces the power greatly, but it also reduces the area of the datapath. This method is often referred to as operator minimization. The number system used in the datapath and the type of the operators also affect the power, for example, the coding system can be switched to some low-power one and the width of the operators can be minimized. [9 p. 9-1]

The inside architecture of an operator can be also optimized for low-power, but this comes usually with area and timing trade-offs which should be taken into account when selecting an operator. Synthesis libraries can have special low-power operators that can be selected, when it is allowed by the datapath latency requirements. Also with resource sharing one can reduce the power and area of a design. In resource sharing, operators are used for multiple different functions where the timing constraints make it possible. This requires some control logic for controlling the shared operators, but the area overhead generated is less significant than each function having own operators. [9 p. 9-2 … 9-14]

Logic Restructuring and Technology Mapping

Logic gate chains and logic tree structures with an input or a net that has high activity should be moved into the end of the chain or tree structure, which reduces the switching activity produced, thus reducing dynamic power dissipation. This optimization method is technology independent and often referred to as logic restructuring. Certain logic structures can be further optimized while being mapped to the physical transistors in order to reduce the total number of transistors needed. This process is often referred to as a technology mapping process as it is a technology dependent optimization method. These optimization methods are typically performed automatically by synthesis tools if a simulation trace is provided. [9 p. 10-17 … 10-19]

4.2.3. Voltage and Frequency Scaling Schemes

Multi-Voltage Partitioning

The main idea of multi-voltage partitioning is to divide the circuit into multiple different supply voltage domains, or as often called, voltage islands, as demonstrated in Figure 20. Low-voltage supply voltage means that the logic is low-speed and low-power and high-voltage, respectively, equals to high-speed and high-power logic. Typically, a design has different performance requirements for each different functional block. If each of these different speeds of functional blocks are implemented with the same speed of logic, power is wasted. With multi-voltage partitioning, the low-performance blocks can be implemented with low-speed logic to reduce the total power dissipation of the circuit while maintaining the performance requirements. Typically, the voltage levels are pre-defined and static for each functional block. [42]

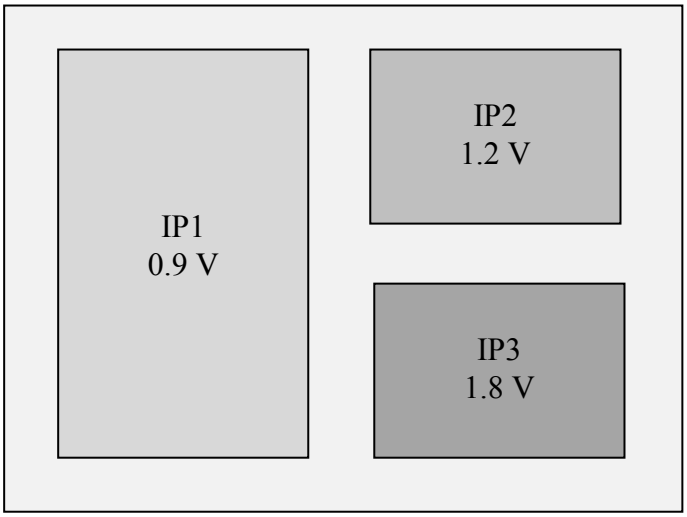

Figure 20. Multi-voltage partitioning in SoC.

Multi-voltage partitioning requires special attention in different levels of design abstraction. In architectural-level, the design is needed to be partitioned into suitable functional blocks that can be implemented with different speed or voltage level of logic. Typically, the RTL synthesis tool can use different libraries for different speed of logic while performing synthesis. Multi-voltage partitioning also presents numerous challenges to the physical design level, which can limit the possible number of different voltage regions as multiple power supplies are needed, multiple power rails and more supply voltage pads. Boundaries between different voltages need to be carefully implemented as level shifters might be needed for signals that are crossing the boundaries, which may be challenging for timing. Multi-voltage partitioning also presents a challenge for verification and testing. [10 p. 29 – 30][42]

Dynamic Voltage and Frequency Scaling

The next step from multi-voltage partitioning is to dynamically or adaptively scale the supply voltage of a whole circuit or separate voltage islands. This scheme is called dynamic voltage scaling (DVS). To further develop this method, the clock frequency can be also scaled as the supply voltage is, this method is then referred to as dynamic voltage and frequency scaling (DVFS). In dynamic voltage and frequency scaling, the voltage and frequency are being dynamically scaled for a different state of the circuit by some controller logic using pre-defined fixed steps, as presented in Figure 21 [44 p. 426]. In a method called adaptive voltage and frequency scaling (AVFS), the voltage is scaled adaptively depending on the workload of the circuit. This scheme deploys the measurement of the work load and a closed control loop for deciding suitable voltage and frequency values for the next cycle. [10 p. 30]

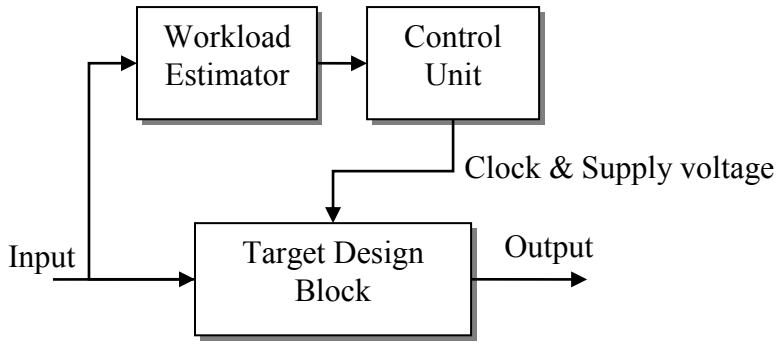

Figure 21. Block diagram of a typical dynamic voltage and frequency scaling.

4.2.4. Design parallelism and splitting

Memory Splitting

Large memories can be split into two smaller memories that have half of the words than the original memory had, as illustrated in Figure 1. The memory operations are now performed in only one of the smaller memories. The overall power dissipation will decrease as a smaller memory have less power dissipation than the split memory and the inactive memory can be potentially gated or put into sleep mode. The

implementation of a memory splitting generates a small area overhead as a penalty. Memory splitting can be implemented with addition of multiplexer to the outputs of the memories, and the memories and multiplexer can be controlled with a bit from the memory address bus, for example, the most significant bit (MSB) or least significant bit (LSB). The output selector mux can be also controlled with the same control bit. This kind of implementation style is illustrated in Figure 22 below. [43 p. 23]

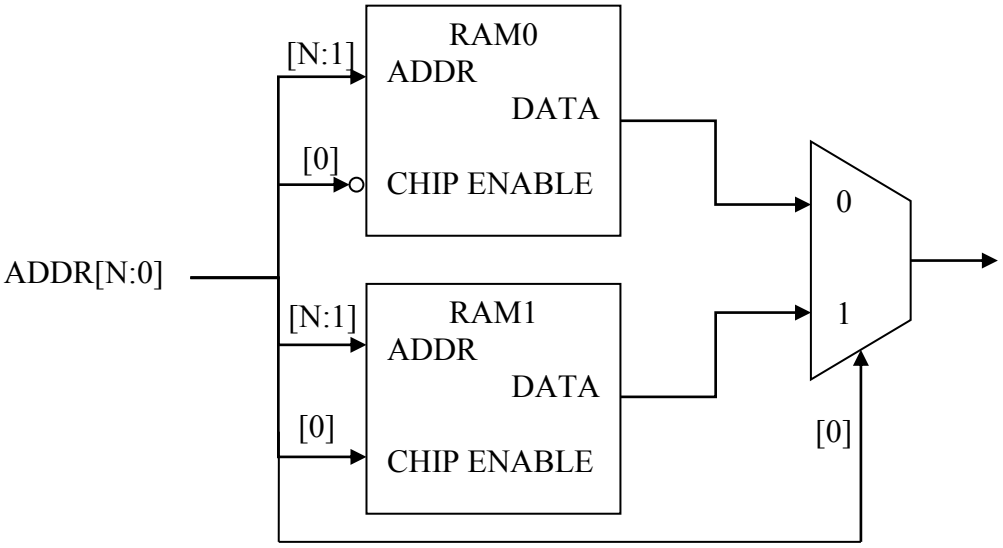

Figure 22. A memory split implementation.

Logic Parallelization

In logic parallelization, a logic block or a datapath can be parallelized into a number of similar parallel slower low-power units that together produce the same result with the same performance as the original single logic block. If a logic block is divided into *N* parallel units, the clock frequency can be divided by the factor of *N*. This enables the supply voltage to be reduced and power reduction to be achieved. Logic parallelization method creates a significant design area overhead related to the number of created parallel units. [9 p. 10-3 … 10-4]

4.2.5. Finite-state machines for low-power

Low-power design goals can be implemented as a part of finite-state machine (FSM) functionality. For example, if there is a long waiting state where the FSM waits for a specific signal before continuing within time that is well defined, the FSM can be easily clock gated for that time in order to save dynamic power. The next three sections discuss some of more complex design methods for low-power FSMs. [9 p. 11-13 … 11-14]

State Coding

By optimum state coding, one can reduce the dynamic power dissipation by minimizing the switching that happens between state transitions. Such state coding

can be achieved with gray coding, which optimizes the bit mask transitions between most probabilistic transitions. State coding can be also used to minimize and optimize the next state coding logic. [9 p. 11-14]

Decomposition

FSMs can be decomposed into a number of smaller FSMs, of which each represents a section of the original FSM. After decomposing, only one of the smaller FSMs needs to be active, while others can be idle and clock or power gated off to save power. A practical way of decomposing FSMs is to identify smaller subroutines in the control loops where the FSM remains most of the time and to decompose those into separate FSMs. Experimental results in [45] show an average power reduction of 31 % with speed increase of 12 % and average cell increase of 48 % [45 p. II-7].

4.2.6. Bus Coding for Low-Power

Buses can be coded to minimize switching activity and to reduce power dissipation. The most commonly used coding method for low-power buses is bus-invert coding. The basic idea of bus-invert coding is that before sending the data, the sender compares it to the current data in the bus and sends an invert signal if the new data can be acquired by inverting the current data. Otherwise, if the data is not suitable for inverting, the sender sends the data to the bus as usual. This scheme requires a computation of Hamming distance between the send and the current data values, from which the decision for inverting can be made. If the Hamming distance is more than the bus width divided by two, the send data can be inverted. The signaling for invert operation requires also a one extra bit in the bus and some extra logic for implementing the needed invert logic for bus receivers and transmitters. [46 p. 49 – 58]

4.3. Static Power Reduction Methods

This section describes the most commonly used design methods targeted for minimizing the static power dissipation component. The static power dissipation is typically very technology dependent, but with these design methods, the design can be optimized for low static power. Most of these methods are implemented on physical levels of the design abstraction, but many of them require special planning and control in system and architectural-level.

4.3.1. Multi-Threshold Voltage Partitioning

Increasing the threshold voltage in CMOS circuits decreases the static power dissipation by reducing subthreshold currents, but it increases the gate delay, thus degrading the performance, as discussed in section 2.3.3. The threshold voltage can be used to reduce the static power dissipation without any major performance trade-offs, by applying low threshold voltage logic on high-performance parts of the design and high threshold voltage logic on other non-performance critical parts of the design. Traditionally, multi-threshold voltage partitioning consists of two different

voltage levels, but more advanced methods have been developed with more than two voltage levels [47 p. 974]. [10 p. 33]

Multi-threshold voltage partitioning adds challenges into design and manufacturing of CMOS circuits. From the design perspective, the challenges are determining which gates to implement with what threshold voltage. This requires accurate simulation and timing analysis, as well as multi-threshold technology libraries for the synthesis. In physical design level, multi-threshold logic does not require much effort to implement as the surface area of different multi-threshold gates is typically identical. Multi-threshold voltage design increases the complexity of the circuit manufacturing process. [10 p. 33]

A typical dual-threshold voltage design flow starts with synthesizing the design with using only high threshold voltage logic. A timing analysis is performed and paths with timing violations are optimized by replacing the necessary logic gates with low threshold voltage ones. Results in [48 p. 208] show average 46 % leakage power savings, with commercial multi-threshold voltage partitioning flow. [48 p. 205]

4.3.2. Multiple Supply Voltages

A similar idea as in the multi-voltage partitioning method, discussed in the section 4.2.3 above, will be discussed next. Rather than implementing the voltage partitioning into whole functional blocks, it is done to separate logic gates. The implementation is to the multi-threshold voltage partitioning discussed above, i.e. the slack of non-critical timing paths, is utilized to save power. The logic in non-critical paths can be connected to lower supply voltage, which reduces power dissipation and performance, while keeping critical logic paths in the original supply voltage level. Signal crossing from high to low voltage levels do not require voltage level sifters, but signal crossing from low into high voltage logic needs simple latch based voltage level converters. When implementing this method, typically, the logic path is given a low-level voltage when the propagation delay is less than the required clock period. [10 p. 31 – 32]

4.3.3. Power Gating

In power gating, separate design blocks are power gated in order to temporarily shut down the idle blocks to reduce the total static power dissipation. Power gating requires system level planning and control in order to successfully manage the timing needed to shut down and power up entire blocks of the circuit, as well as determining when a block is not needed and can be shut down and to control state recovery. Standard HDL coding languages do not support describing of any kind of power gating fabric, which has led to the development of a standardized language to describe the power intent. The standard is specified by the Institute of Electrical and Electronics Engineers (IEEE) and officially known as IEEE 1801, but often referred to as Unified Power Format (UPF) [49].

Power Gates

Power gating can be implemented in CMOS circuits as fine grain power gating or coarse grain power gating. In fine grain power gating, a power gate is added to every logic cell. This generates a large area overhead, but is easily implemented with standard EDA tools as the power gates are integrated within the standard logic cells. The coarse grain power gating approach implements a block level power gating with a small number of power gates for each block, thus reducing the area overhead as compared to the fine grain approach. [10 p. 36 – 37].

The challenge for both approaches is to manage the in-rush currents when the power is switched back on. This requires the power gate transistors to be large in physical size to be able to handle the in-rush currents. The larger gates are slower, thus slowing the on and off switching times, which can have implications on the effectiveness of the power gating. [10 p. 36 – 37]

A typical power gate is implemented with two MOS transistors, one between supply voltage and the other between ground or negative supply voltage, as illustrated in Figure 23 [10 p. 36]. Adding the MOS transistors between the supply voltage and the design block lowers the supply voltage slightly from the original value. The same effect happens on the ground or negative supply voltage side, but there the voltage is slightly increased. These slightly different voltages that the power gated design blocks experiences are called as virtual supply voltage and virtual ground. When the block is turned off, the virtual supply voltage meets the virtual ground voltage causing a virtual supply voltage collapse, as shown in Figure 24 [10 p. 36]. [10 p. 36 – 37]

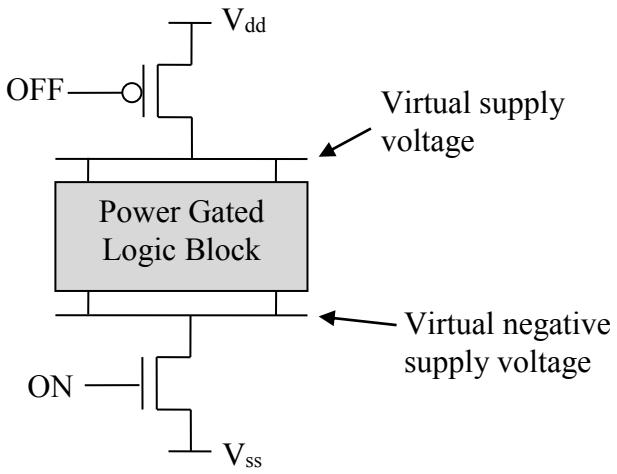

Figure 23. Typical power gating.

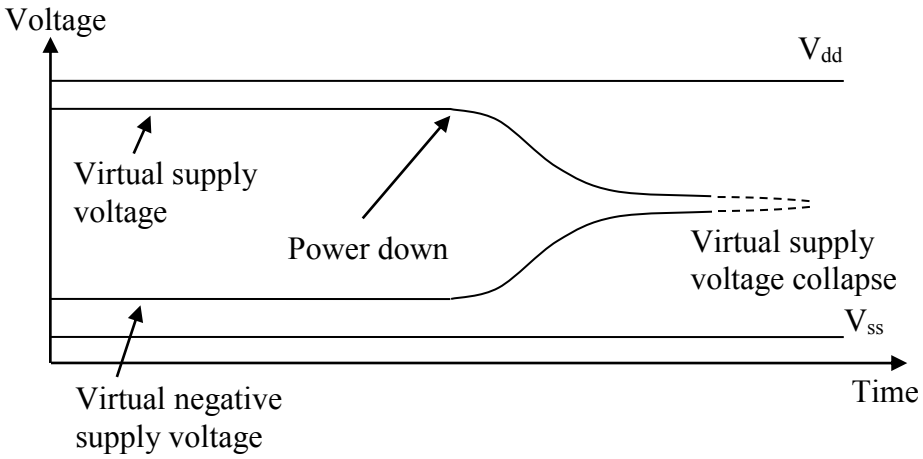

Figure 24. Supply voltages in power gating.

Isolation Cells

When power is switched off from a logic block, the outputs of the block are floating and can cause crowbar current and unwanted logic signaling into the inputs of the powered blocks. This requires isolation between the power gated logic blocks and the other logic structures that remain powered. Isolation cell can be constructed with basic logical port, for example, by using a single AND port with control signal to force the output signal to logical low or OR port for forcing the output to logical high. [50 p. 45 – 50]

State Retention

Logic blocks that contain sequential logic may require saving the current state before being powering off in order to continue functioning properly and to prevent any information loss after powering back on. For state retention, a few methods exist of which, the most common are a scan-chain based method and a retention register method. The scan-chain based method utilizes the test chain for registers used in manufacturing testing. Before powering down a logic block, the information from registers is pulled out via the scan-chain and fed into memory. After powering back up, the information is fetched from the memory and fed via the scan-chain back to the registers. In the retention register based method, there is a shadow register for each register in the block that are powered on during the power down period in order to save and restore the information of each register. [50 p. 50 – 59]

Control

Power gating requires a pre-defined protocol, called power cycle sequencing, in order to successfully power on and off design blocks. Figure 25 illustrates typical power cycle sequencing, with state retention [50 p. 61]. It starts by disabling clock signal and enabling isolation to the outputs and or inputs of the powered down design block. Next, the state retention starts with saving the current state. After the state is saved, a reset is performed followed by powering down the block. Powering back on

is performed similarly in reserve order with replacing save by restore signaling. [50 p. 61]

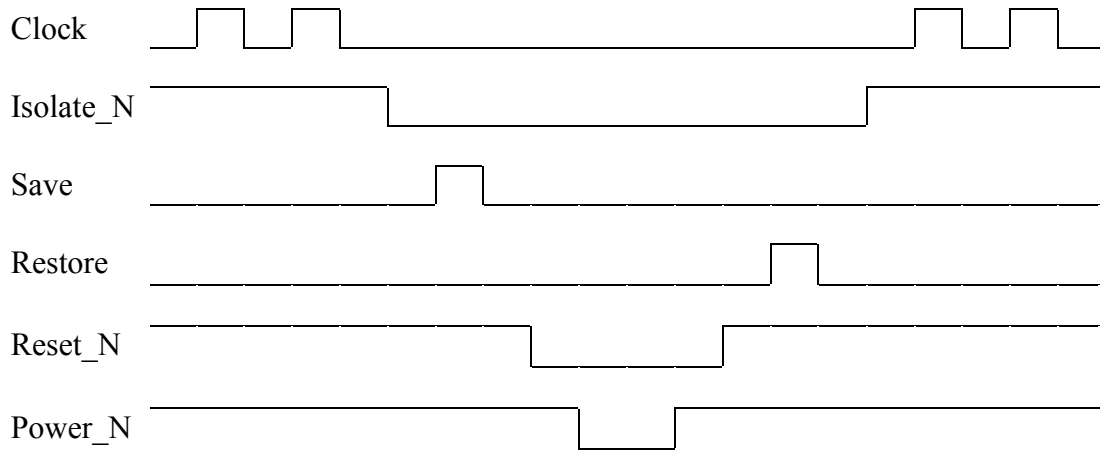

Figure 25. Power cycle sequencing.

4.3.4. Body Biasing

A decrease in threshold voltage increases strongly the leakage currents in MOS transistors, which limits reaching the lowest possible threshold voltage and supply voltage for lowering the power dissipation. This problem can be overcome with a technique called body biasing. In body biasing, a body bias voltage is applied between the body substrate and the ground. With this body bias voltage, the threshold voltage levels can be controlled for controlling the leakage and timing of the transistor. Applying forward body bias voltage, the threshold voltage decreases increasing leakage power dissipation and performance by decreasing propagation delay. Reverse body bias voltage increases the threshold voltage, which reduces leakage power dissipation and increases propagation delay lowering the performance. Body biasing requires additional power supplies and controlling logic that slightly increases area overhead and design complexity. [51 p. 133 – 134]

4.3.5. Minimum Leakage Vectors

Logic gate structures have different levels of leakage in different logical states as the different MOS transistors are on and off in the transistor stack structure. This can be utilized and a set of input vectors for minimizing the leakage currents can be identified. The input vectors for minimizing leakage are referred to as the minimum leakage vectors. A leakage power reduction can be achieved by forcing the inputs of idle combinational logic structures to these minimum leakage vectors. This method requires a technology mapping done to the design in order to identify the minimum leakage vectors. Also some architectural and behavioral level consideration is needed as there is a need for control logic, which has to identify the idle states of logic blocks and force the minimum leakage vectors to the inputs. [52 p. 40 – 41]

5. CASE STUDY: RTL POWER ESTIMATION AND REDUCTION METHODOLOGY WITH A STATE-OF-THE-ART COMMERCIAL RTL POWER TOOL

Chapter 5 is the case study part of this thesis, where an RTL power estimation and reduction were performed by using a state-of-the-art commercial RTL power tool. The first section, section 5.1., discusses the basic background of commercially available RTL power tools. The methodology and results are presented for RTL power estimation in section 5.2., and finally, for RTL power reduction in section 5.3.

5.1. Background to Commercial RTL Power Tools

Typical standard features that are found in commercial RTL power tools include means for analyzing simulation activity files, performing average and time-based RTL power estimation and performing automated or manual RTL power reduction. Analyzing simulation activity files is an important function as the quality of power estimation and reduction results are tied strongly to quality of the simulation test case and the activity file generated from it. The analysis gives also valuable information for the user where potential design problems related to power might be present in the design. In order to run these tools, an RTL design, power characterized libraries and a simulation activity file are typically needed as the setup input. [53 p. 4, 5, 6, 25][54][55][56][57]

More advanced features that can be found in the RTL power tools are means of performing power estimation and reduction without a simulation activity file, gate-level power estimation with gate-level or RTL stimulus, UPF-based power estimation and reduction and means of calibrating the accuracy of the RTL power estimation. As discussed in section 4.3.3, UPF is the IEEE standard for describing the power intent of a design. RTL power tools cannot only read the UPF of a design for the power estimation, but they can also suggest changes to it and regenerate a more optimized one in the power reduction phase. The calibration links the RTL power estimation to the physical power, and typically, it is a design-independent way of characterizing the power parameters of the technology from the physical level into the RT-level. [53 p. 4, 15, 25, 27][54][55] [56][57]

As a summary, the steps of a typical RTL power tool usage flow are illustrated as a flow chart in Figure 26 below. And the features of a typical commercial RTL power tool can be listed in to standard and advanced features:

1. Standard RTL Power Tool Features:
   - Simulation activity file analysis
   - RTL power estimation
     - Average and time-based
   - RTL power reduction
     - Automatic and manual

2. Advanced RTL Power Tool Features:
   - Power estimation and reduction without a simulation activity file
   - Gate-level power estimation
     - Using gate-level or RT-level simulation activity file

- UPF-based power estimation and reduction
- Calibration from gate and post-layout level

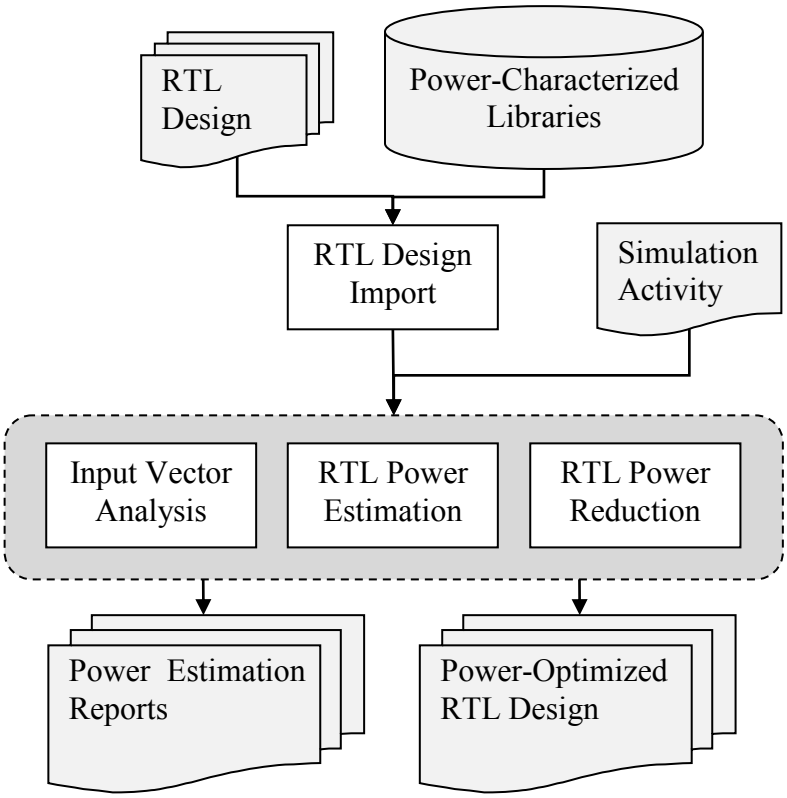

Figure 26. RTL power tool flow overview.

RTL Power Estimation and Reduction in a Design Flow

A typical SoC design flow is usually divided into four different abstraction levels that are ordered from the highest to lowest abstraction as system-level, architectural-level, gate-level and physical-level [58]. These four design abstraction levels are illustrated in Figure 27 [58], with related tools and activities for each level. This also illustrates a typical top-down design flow, where the design process starts from the system-level and goes step by step through the design abstraction levels. The RTL power tools are used in the RT-level design and verification, which are performed in the architectural-level. As highlighted in the figure, the architectural-level is between the system and gate-level design abstraction levels.

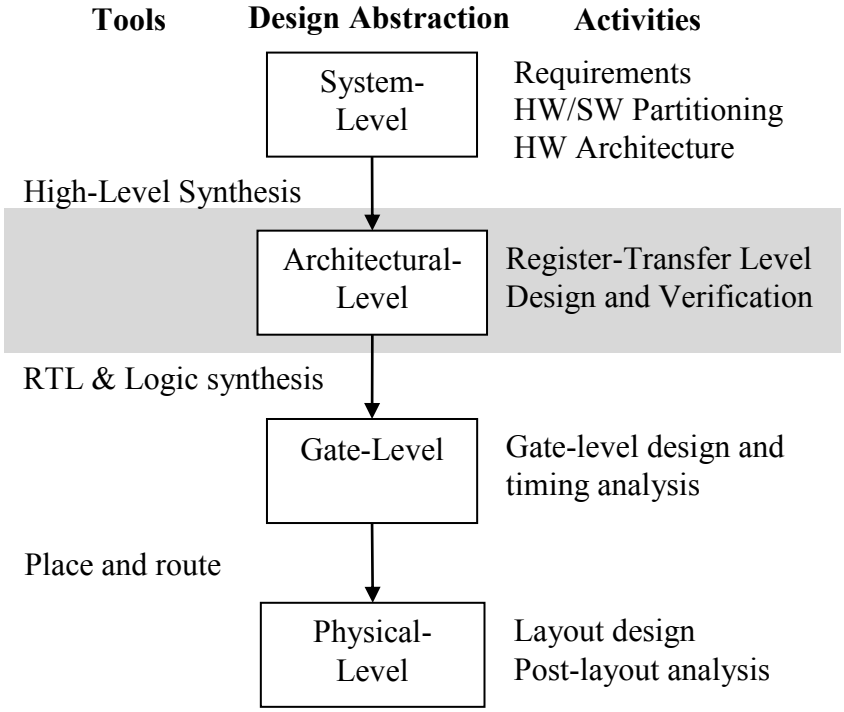

Figure 27. Design abstraction levels in SoC design.

In a typical top-down design flow, important design information for specific design abstraction level is not available until the lower abstraction levels have been reached. This creates lengthy feedback loops back to the upper design abstraction levels for optimizing the design in the higher levels of abstraction. If power estimation results are available only after gate- or physical-level analysis, it may be too late to change the design architecture, which was already planned and documented earlier in the project. Late changes to RTL or system-level design would generate large and time consuming feedback loops to the design flow, which is not desirable. [9 p. 43-2]

In so-called feed forward design flow, the feedback loops are kept inside of specific abstraction levels instead of having cross-abstraction level loops, as illustrated in Figure 28 [9 p. 43-3]. This enhancement to the design flow generates a need for abstraction specific analysis and optimization tools, such as RTL power tools. As highlighted in the flow figure, RTL power tools are used for performing RTL analysis and RTL optimization in the architectural-level. [9 p. 43-3 … 43-4]

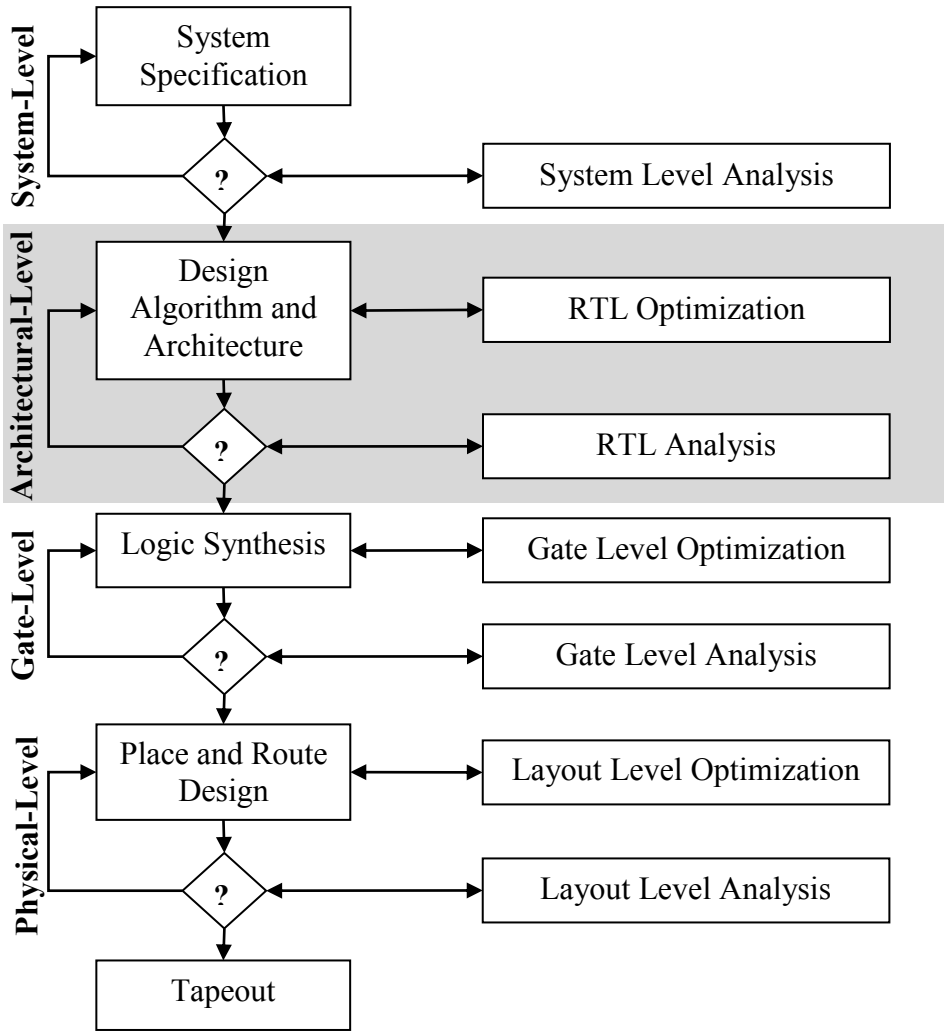

Figure 28. Feed forward design flow.

5.2. RTL Power Estimation

The first section describes the methodology and setup used in this work for the RTL power estimation, the second section presents the results obtained with the methodology and the third section analyzes the achieved results. A commercial RTL power tool was used for performing the RTL power estimation in this work; the details of the tool are not described in this thesis due to confidentiality reasons.

5.2.1. RTL Power Estimation Flow

In the RTL power estimation methodology that is used in this thesis, an HDL description of the RTL design, power-characterized libraries, various synthesis and technology parameters, net capacitance information and simulation activity data are used as inputs. The RTL estimation is performed in two steps: the first step is the RTL design import step, where the RTL design is read in and analyzed. After the RTL design import, the RTL power estimation step is performed, where the actual

power calculations are made. The power calculation uses simulation trace from an RTL simulation to determine the activity in the design. Required power characteristics and information are obtained from the power-characterized libraries. Information of the net capacitance is used to determine the load capacitances in the design, while the various synthesis and technology parameters define more specific information of the synthesis and the target technology. The RTL power estimation phase produces various power reports, where the average power dissipation is reported for the design. A simplified flow chart used for the RTL power estimation in this thesis work is presented in Figure 29 below.

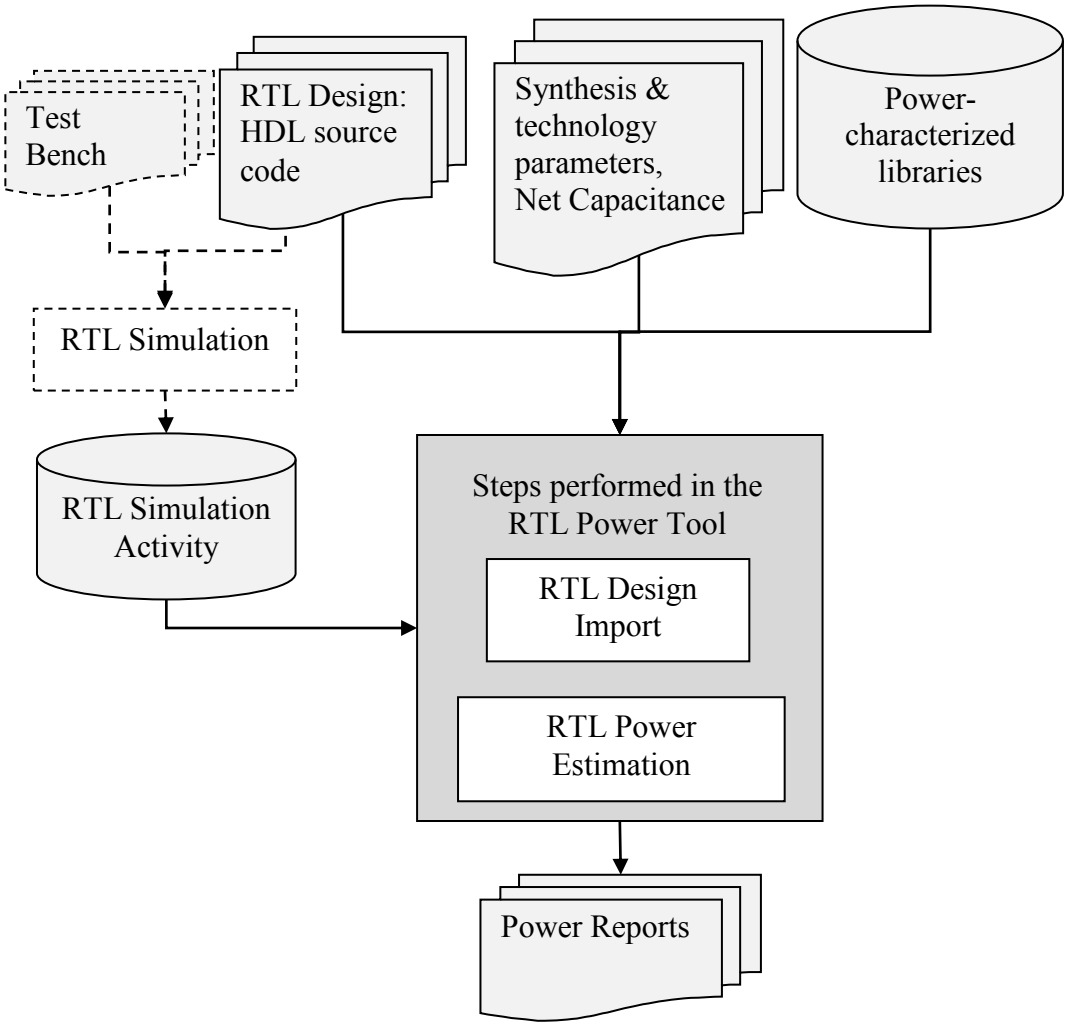

Figure 29. RTL power estimation flow.

Power characterized libraries are the same technology libraries which are used for the full RTL synthesis. Technology and synthesis parameters defined in the power tool setup were the following:

- Simulation activity file type
- Simulation time window
- Clock definitions
- Clock buffer cell

- Clock gating cell
- Clock gating minimum bit width
- Default output load capacitance
- Default signal transition time (slew)
- Don't use cells
- Non-scan or scan chain flops to be used in the synthesis
- Threshold voltage (V_{th}) cell mix
- Memory port definitions.

The following input file formats were used:

- RTL design files and libraries
  - Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL) and Verilog file formats
- RTL simulation activity file
  - Fast Signal Data Base (FSDB) file format
- Power characterized technology and memory libraries
  - Liberty library format (.lib)
- Net capacitance file
  - Wire load model (WLM) format.

5.2.2. Correlation with Gate-Level Estimation

For the power estimation, three test designs were used. The test designs were sub-blocks of a digital signal processing (DSP) system based on 28 nm target technology node. In this section, the designs are referred to by their size as small, medium and large design. The small design is about 100 k equivalent gates, medium 700 k gates and large 5.4 M gates. More details of the test design are presented in Table 1 below, where the relative combinational and sequential logic area is presented for each design.

Table 1. Characteristics of the test designs

Test design	**Size (equivalent gates)**	**Combinational Logic**	**Sequential Logic**	**Target technology**
Small	100 k	58 %	42 %	28 nm
Medium	700 k	66 %	34 %	28 nm
Large	5.4 M	58 %	42 %	28 nm

It should be noted that the reference gate-level power values are also estimates, with deviation from 5 to 10 % off the real values. The gate-level estimates were made with a gate-level netlist obtained by synthesizing the test designs with the same libraries, wire load model and simulation activity files as in the RTL power estimation. As power optimization settings, the logic synthesis included a clock gate insertion with the same bit width and same threshold voltage mix rate as defined in the RTL power estimation setup.

For the small design, the RTL power estimation error against gate-level power estimation for total power is 7 %, dynamic power 11 % and static power 1 %, as illustrated in Figure 30. The error of the RTL power estimation for the medium size design for the total power is 11 %, dynamic power 14 % and static 2 %, as shown in

Figure 31. The same numbers for the large design are 14 % for total power, 13 % for dynamic power and 19 % for static power that is shown in Figure 32. From all of the three designs, the total relative error for the total power is 11 %, 13 % for dynamic power and 7 % for static power. The results are also summarized in single Table 2 below.

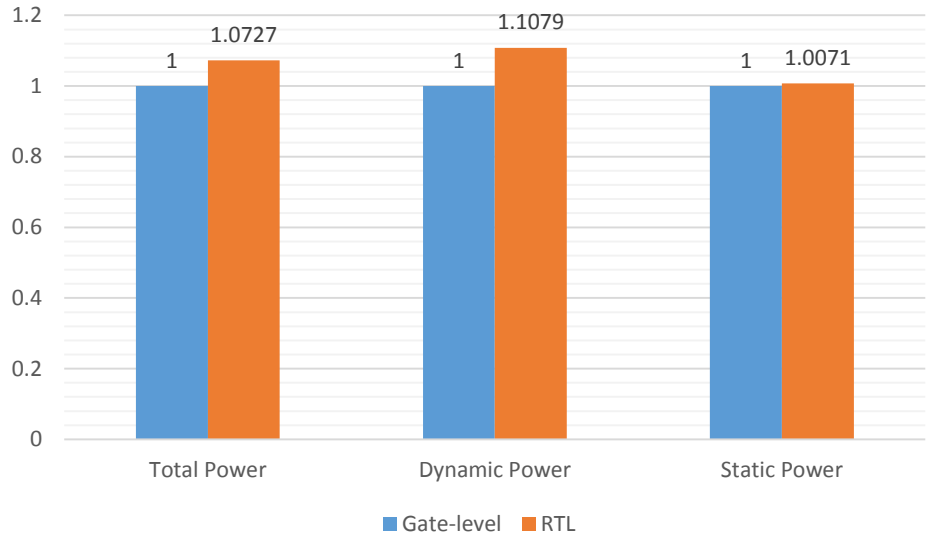

Figure 30. Small design RTL power estimation correlation with gate-level estimate.

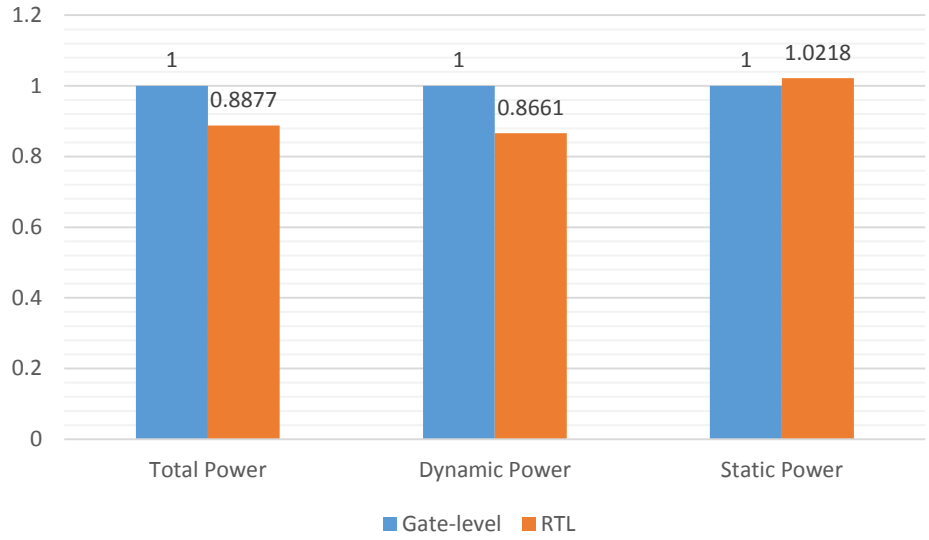

Figure 31. Medium design RTL power estimation correlation with gate-level estimate.

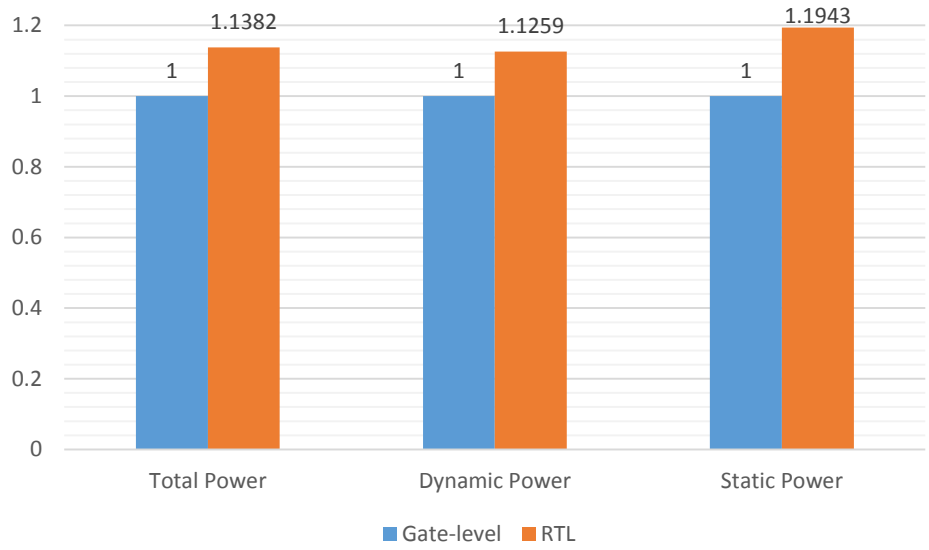

Figure 32. Large design RTL power estimation correlation with gate-level estimate.

Table 2. RTL power estimation deviation against gate-level reference result

Design	**Total Power**	**Dynamic Power**	**Static Power**
Small	7 %	11 %	1 %
Medium	-11 %	-14 %	2 %
Large	14 %	13 %	19 %
Total Deviation	**11 %**	**13 %**	**7 %**

The runtimes of RTL power estimation depend on the complexity of the design and the simulation time window of the switching activity file. The observed runtimes and used simulation time ranges are presented in Table 3 below. These runtimes are only an approximation of the actual runtimes, as they are dependent highly on the various settings in the setup and the characteristics and the current load of the computational infrastructure used to run the power estimation.

Table 3. Runtimes of RTL power estimation

Design	**Simulation Time Range**	**Average Power Estimation Runtime**
Small	33 µs	3 min
Medium	20 µs	35 min
Large	118 µs	5 h

There were also warnings about missing sequence numbering in the simulation activity files, which may affect the accuracy of the power estimation as the order of sequences can be mixed. Also, there were some X-state signals reported in the simulation time window, but they were investigated to be related into the initial reset sequence, thus not affecting the power estimation. As presented in Table 4 below, the annotation rate in the power estimation was between 99 and 100 % for power critical nets and between 93 and 100 % for non-power critical nets. Annotation rate represents how well the simulation activity maps into the design, i.e. how many nets have activity annotated from the activity file. For nets missing the activity, the tool

propagates activity from other nets. If the propagation is not possible, i.e. the net is an input, the tool uses a default activity value.

Table 4. Simulation activity mapping

Design	Annotation rate of the power critical nets	Annotation rate of the non-power critical nets	X-state signals of the total time
Small	99 %	100 %	3.5 %
Medium	100 %	93 %	0.5 %
Large	100 %	100 %	1.3 %

The deviation in the total number of registers between the netlist generated by the fast synthesis and gate-level netlist is negligible for the small and large design, only 0.44 and 1.3 %. But for medium design, there is a slightly increased deviation of 6.0 %. The clock gating percentage is almost identical between the netlists. The synthesis results in terms of the deviation in register count and the difference in register gating coverage are summarized in Table 5 below.

Table 5. Synthesis results in terms of register deviation and register gating

Design	Gated registers reported in RTL power tool	Gated Registers in gate-level netlist	Deviation in total number of registers.
Small	98.5 %	98.6 %	0.4 %
Medium	95.6 %	95.8 %	6.0 %
Large	97.8 %	98.0 %	1.3 %

5.2.3. Analysis of the RTL Power Estimation Results

The simulation file mapping to the design is critical in terms of the power estimation accuracy. Typically, the annotation rate should be over 90 to 95 % in order to consider the results to be accurate and valid. In the results, the annotation rate was reported to be between 99 to 100 % for the power critical nets in the designs. This rate is well above the target and signify the fact that simulation activity mapping is not causing any loss of accuracy, thus making the achieved results viable.

The deviation in the static power can indicate how well the estimated post-synthesis structure of the design matches with the actual structure of the gate-level netlist. As there should be little variation in the static power estimate if the design structures matches, since both gate and register-transfer level power calculation use the same static power dissipation data values from the same libraries. In the results, the deviation in the total number of register between the netlist can give some indication of how well the structures correlate with each other. The deviation in the total number of registers for small and large designs are 0.4 % and 1.3 %. These deviations are very negligible and indicate that structures of the netlists are similar. For the medium design, the deviation is 6.0 %, which can be seen as a sign that there is a slight difference between the gate-level netlist and the structure estimated by the tool.

In the dynamic power estimation, the capacitive power is calculated by the tool based on the capacitive values in the wire load model, while the internal power values are obtained directly from look-up tables inside the power-characterized libraries. The fundamental difference between the level of abstraction in gate-level

and register-transfer level is that instead of a functional block, in gate-level, the block is represented as separate logical gates, which can affect greatly the power estimation correlation between register-transfer and gate-level. Another cause for inaccuracy of dynamic power estimation is how well the RTL power tool can predict where the logic synthesis engine will place clock gates. The gated register percentage is almost identical between the netlist, the results show only the maximum of 0.2 % of deviation. But this does not necessarily mean that the clock gating is similar, as one should further study where the clock gates are inserted in order to draw any conclusions. Further examination of the results revealed that the estimation error was larger in design hierarchies that contained a lot of arithmetic functions. In other words, the power estimation error was larger for datapaths containing arithmetic functions than for datapaths with only registers and combinational logic. This might be due to the differences how the tool processes arithmetical operators compared to the logic synthesis engine.

From these three different test cases, the total average error for total power compared to gate-level reference is 11 %, which is a suitable accuracy level for performing power reductions in RTL level and is a good early estimate of the final power. Smaller than 20 % error marigin can be considered as good enough for the purpose of RTL power reductions, where the focus is on producing consistent and comparable results. The error could be further reduced with calibration and manual fine-tuning of the tool flow. Overall, the results of the RTL power estimation offer very detailed information of the power consumed by the design and from where in the RTL code it originates from. The challenge of this power estimation methodology, and of any other power estimation method that uses signal activity from a simulation, is that the power estimation results are highly dependent on the simulation test case used to generate the activity file. This makes the power estimation results only as viable as the simulation test case is, which makes the selection of the simulation test case critical for power estimation. The second option is to perform the power estimation without a simulation activity file, as discussed in section 5.1, as commercial RTL power estimation tools also support vectorless power estimation. But the challenge in this approach is how to define activity input values that produce viable results.

5.3. RTL Power Reduction

The first section describes the flow used in the RTL power reduction methodology in this work. The results reported by the tool are presented in the second section and third section is focusing on analyzing the results. The RTL power reduction was performed using a commercial RTL power tool and the same test design and input data as in the RTL power estimation above.

5.3.1. RTL Power Reduction Flow

The input files, file formats and parameters used for the RTL power reduction were exactly the same as for the RTL power reduction, discussed more detail in section 5.2.1. The RTL power reduction is based on the data from average RTL power estimation, which is required to be performed before the power reduction. The implementation of the power reduction opportunities is illustrated in the overall flow

in Figure 33 as dashed, since they were not performed as a part of this thesis. The reduction opportunities can be implemented either manually by the designer or automatically by the tool, which is also illustrated in the figure. The tool produces various detailed reports of the found power opportunities and code for implementing the RTL changes.

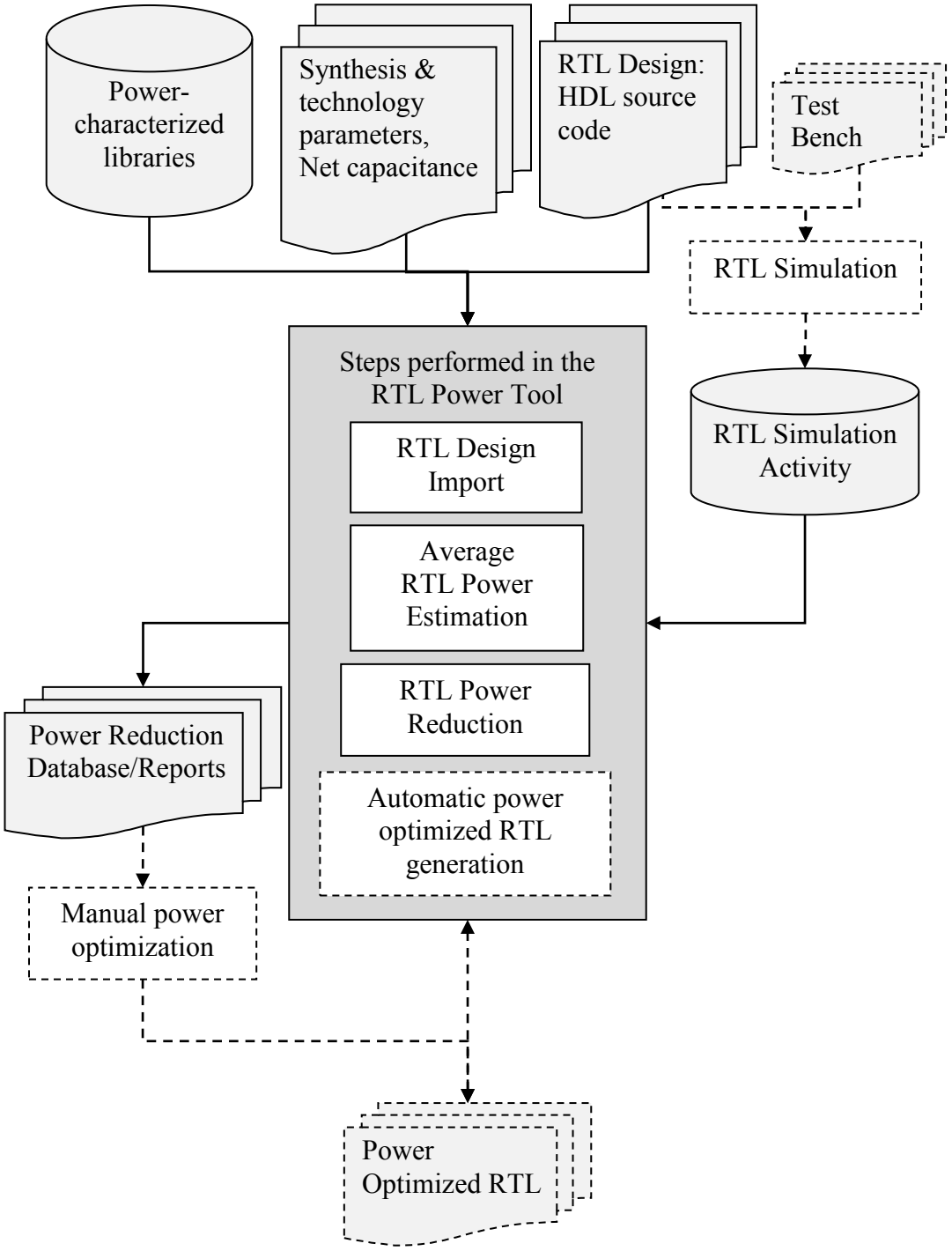

Figure 33. RTL power reduction flow.

5.3.2. RTL Power Saving Estimates

The test design and setup files used to perform power reduction analysis are the same as in the RTL power estimation performed in section 5.2. The results presented in this chapter are estimates of the saved power generated by the RTL power tool, and not results from an actual implementation. In Table 6, the potential total and dynamic power savings are presented for each design. The potential power savings consist of all of the power saving opportunities found in the design, but some of these power saving opportunities can be such that they cannot be actually implemented. Respectively, in Table 7, power saving estimates for the automatic power-optimized RTL generation are presented for each design.

Table 6. Potential power saving estimates

Test Design	**Dynamic Power Savings**	**Total Power Savings**
Small	43.3 %	29.4 %
Medium	13.1 %	10.0 %
Large	24.2 %	20.2 %

Table 7. Power saving estimates for automatic RTL generation

Test Design	**Dynamic Power Savings**	**Total Power Savings**
Small	23.1 %	15.6 %
Medium	2.7 %	2.0 %
Large	8.0 %	6.6 %

The total saved power can be also broken down into saved power by each power reduction technique. Table 8 presents the total saved power and the number of occurrences for each power reduction opportunity found in the small design. Table 9 and Table 10, respectively, present the same to the medium and large design. The total saved power in Tables 8, 9 and 10 is higher than in the Table 6 above, because these tables include all of the reported reduction opportunities, including some overlapping opportunities.

Table 8. Power reduction opportunities for the small design

Reduction Opportunity	**Number of Occurrences**	**% of Total Power**
Inefficient Enables	217	8.0 %
Redundant Memory Access	3	1.5 %
Register XOR Self-Gating	52	13.7 %
Redundant MUX Input Activity	81	0.1 %
Redundant Register Input Activity	10	0.0 %
Memory Splitting	1	1.5 %
Observability Don’t Care	27	9.2 %
Total	**391**	**34.0 %**

Table 9. Power reduction opportunities for the medium design

Reduction Opportunity	**Number of Occurrences**	**% of Total Power**
Inefficient Enables	947	8.5 %
Redundant Memory Access	18	0.2 %
Register XOR Self-Gating	37	1.2 %

Redundant Register Input Activity	9	0.0 %
Redundant MUX Input Activity	247	0.1 %
Redundant Register Input Activity	9	0.0 %
Stability Condition	2	0.0 %
Total	**1269**	**10.0 %**

Table 10. Power reduction opportunities for the large design

Reduction Opportunity	**Number of Occurrences**	**% of Total Power**
Inefficient Enables	1649	3.4 %
Redundant Memory Access	461	0.3 %
Register XOR Self Gating	7426	15.4 %
Redundant MUX Input Activity	3015	0.7 %
Redundant Register Input Activity	2233	0.0 %
Memory Splitting	16	0.2 %
Observability Don't Care	1790	3.9 %
Stability Condition	702	0.5 %
Total	**17292**	**24.4 %**

The power reduction opportunities can be also ordered by the achieved power savings, as visualized in Figures 34, 35 and 36. The reduction opportunities are ordered from the largest to the least by the indicated total power savings and include all found power reduction opportunities for each design. The horizontal axis represents the number of power saving opportunities required to implement in order to achieve the total power saving percentage in the vertical axis. Table 11, Table 12 and Table 13 show the type and generated power savings of the top 10 power saving opportunities for each design. In Table 14, it is presented what the top 10 % of the power saving opportunities contribute in terms of total power and of the power savings for each test design.

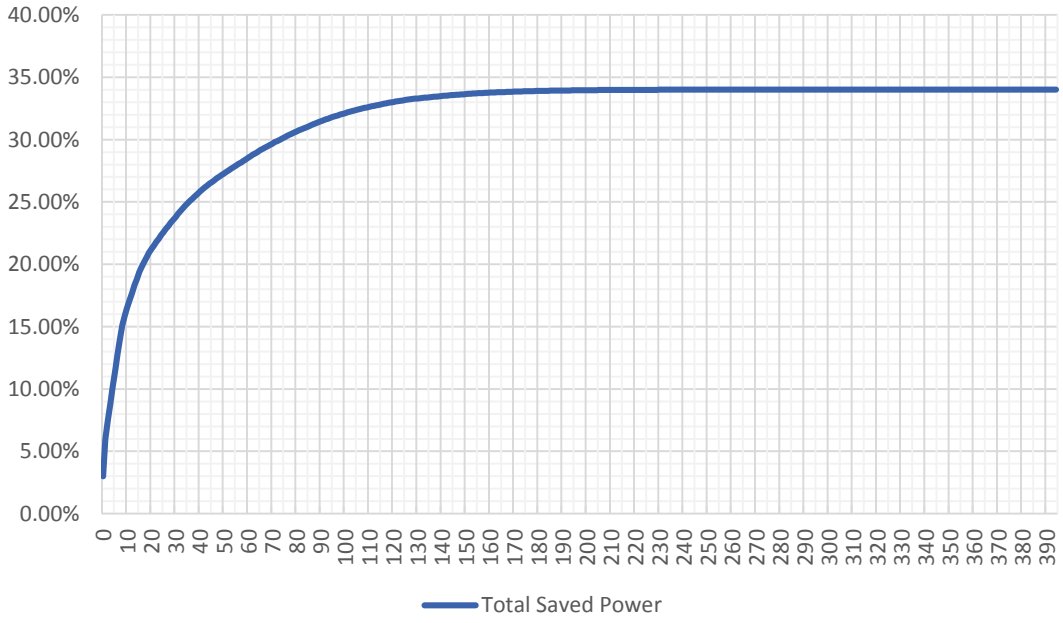

Figure 34. Cumulative power savings of the reduction opportunities for the small design.

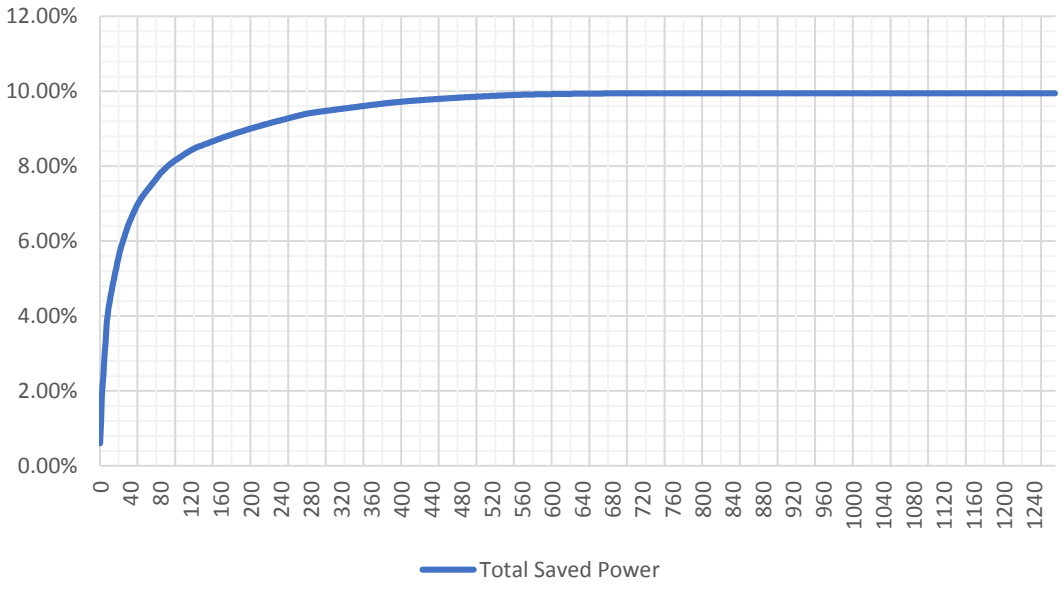

Figure 35. Cumulative power savings of the reduction opportunities for the medium design.

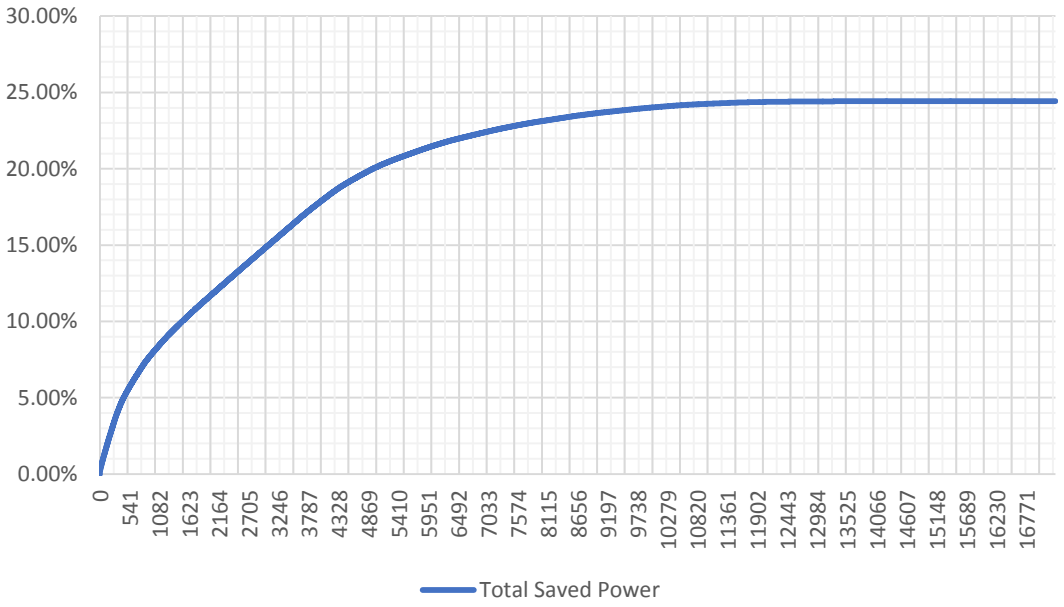

Figure 36. Cumulative power savings of the reduction opportunities for the large design.

Table 11. Power savings of the top 10 power reduction opportunities for the small design

#	**Reduction opportunity**	**% of Total Power**
1	Observability Don't Care	3.0 %
2	Observability Don't Care	3.0 %
3	Memory Splitting	1.5 %
4	Register XOR Self-Gating	1.4 %
5	Register XOR Self-Gating	1.3 %
6	Register XOR Self-Gating	1.3 %

7	Register XOR Self-Gating	1.3 %
8	Redundant Memory Access	1.3 %
9	Register XOR Self-Gating	1.1 %
10	Register XOR Self-Gating	0.8 %
Total		**16.0 %**

Table 12. Power savings of the top 10 power reduction opportunities for the medium design

#	**Reduction opportunity**	**% of Total Power**
1	Inefficient Enables	0.6 %
2	Inefficient Enables	0.6 %
3	Inefficient Enables	0.6 %
4	Inefficient Enables	0.3 %
5	Inefficient Enables	0.3 %
6	Inefficient Enables	0.3 %
7	Inefficient Enables	0.3 %
8	Inefficient Enables	0.3 %
9	Inefficient Enables	0.3 %
10	Inefficient Enables	0.3 %
Total		**3.9 %**

Table 13. Power savings of the top 10 power reduction opportunities for the large design

#	**Reduction opportunity**	**% of Total Power**
1	Register XOR Self-Gating	0.08 %
2	Register XOR Self-Gating	0.04 %
3	Register XOR Self-Gating	0.04 %
4	Register XOR Self-Gating	0.04 %
5	Register XOR Self-Gating	0.04 %
6	Observability Don't Care	0.03 %
7	Observability Don't Care	0.02 %
8	Register XOR Self-Gating	0.02 %
9	Register XOR Self-Gating	0.02 %
10	Register XOR Self-Gating	0.02 %
Total		**0.34 %**

Table 14. Cumulative savings from the 10 % of the power reduction opportunities

Test Design	**# 10 % of Reduction Opportunities**	**% of The Total Power**	**% of The Power Savings**
Small Design	40	25.8 %	75.9 %
Medium Design	127	8.5 %	85.3 %
Large Design	1729	12.1 %	49.5 %

5.3.3. Analysis of the RTL Power Reduction Results

The average estimated total automated power savings for the three test designs is 13 % and the potential is 20 %. The estimated power savings vary between the test

designs, 10 % to 30 % between the potential and 2 % to 16 % between the automated power savings. This variation might be caused by a difference in design and coding style, but the power savings also simply depend on how power efficient the design is already and how much room there is for improvements in each design. The dynamic power savings are higher than the total power savings as the reduction techniques utilized by the tool are for optimizing the dynamic power dissipation, and each implementation of a power reduction opportunity requires additional logic, which increases the total static power dissipation. The estimated power savings are the smallest for the medium size design, and it has less sequential and more combinational logic than the other test designs. This might indicate that the structure of the medium size design differs from the other designs, and thus there is not that much power saving opportunities.

From Figures 34, 35 and 36, it is clear that the major proportion of the potential power savings could be achieved only implementing a fraction of the reduction opportunities. The top 10 % of the power reduction opportunities make the total of 76 % of the potential power savings for the smallest design, 85 % for medium and 50 % for the largest design. It would also be impossible for a designer to manually implement all of the reduction opportunities reported by the tool, because of the sheer number of the opportunities. If automated power optimized RTL generation is not used, the designer could implement the top 10 power savings. In terms of power savings, this would mean 16 % saved power for the small design, 3.9 % for medium design and 0.34 % for the large design. The percentage of the saved power decreases when the design size and total power dissipation increases. This means that the absolute power saving values of the top 10 opportunities are similar, i.e. the same absolute power savings are achieved for each test design if the top 10 power reduction savings would be implemented. For a whole SoC containing several tens of similar building blocks, this would generate meaningful absolute power savings quickly.

The register XOR self-gating, the improving of inefficient enables and the observability don't care appear to contribute the large proportion of the estimated total power savings. The number of opportunities for these power saving techniques is also high in each test design, but a high occurrence in the top 10 power reduction opportunities might suggest these four techniques are the most successful in these test designs. Almost all of the top 10 reduction opportunities are different clock gating opportunities, for example, there is no datapath optimizations. This might be caused by the design or HDL coding style of the test design, thus allowing only gating opportunities to be found. One possible cause for the lack of datapath power optimizations discussed in section 4.2.2 may be that the arithmetic operators are inferred from the HDL code rather than coding the actual functions by hand, which may block the tool finding these types of power reduction opportunities.

The power reduction results correlate highly with the activity data obtained from the RTL simulation. This makes the simulation test case crucial for the power reduction, and the test case should represent the functional state of the design that is the most common in terms of power dissipation. For example, tests with an idle simulation cases were made and then the estimated potential saved dynamic power was over 95 %. This indicates that the tool will try to eliminate aggressively all the redundant activity in the design and focus is on reducing the dynamic power. Implementing the power optimizations from the idle simulation case might be catastrophic to the power dissipation, area and performance when the design is not in

idle operating state. Whether the power optimization are made manually or automatically, the power optimized design should always be verified and the logical equivalence should be checked for ensuring that the power optimizations have not changed the functionality of the design. And the power savings should also be confirmed with power estimation, as there can be variation between the real and the estimated savings. The implementation of the power reduction opportunities and validating the power saving estimates were left out of the scope of this thesis.

As impressive it sounds that the tool could generate automatically a power optimized RTL with 2 to 16 % total power savings, the real potential is in the guidance that the tool can give to a designer. For example, from the reports, a designer can spot problematic sections in the design and perform major algorithmic or architectural design changes, which could have a very high effect on the power dissipation of the design. It should be also noted that the power savings estimated by the tool would be in addition to power optimizations performed in synthesis. For example, clock gating and mixing of multiple threshold voltage cells are implemented by synthesis tools. This means the information from the RTL power tool can be used for optimizing logic synthesis flow for the specific design, for example, forcing the synthesis engine not to insert clock gates in places where they would waste power based on the simulation.

6. DISCUSSION

This thesis work studied and presented common RTL power estimation and reduction methodologies found in the literature. Based on this literature study, an RTL power estimation and reduction methodology was defined using a commercial state-of-the-art RTL power tool. The methodologies were tested with three different test designs and the results were analyzed and presented thoroughly. The power estimation and reduction methodologies and the measured results are the core results of this thesis. The methodology requires a similar setup as a synthesis process – HDL design files, technology libraries and various parameters to control the process. The only addition is that an RTL simulation is required as the simulation trace is needed for calculating power. At the time of RTL design, the simulation activity should be available, as typically the functionality of the design is verified by performing an RTL simulation before running the synthesis.

For the RTL power estimation methodology, the three different test cases yielded an average deviation of 11 % between register-transfer level and gate-level power estimation, which can be considered as an excellent result when considering how rugged the used setup actually was. The methodology demonstrated also consistency as the error was between 7 % and 14 % with the three test cases, which had large differences in gate counts of 100k to 5.3M equivalence gates. This level of accuracy is very suitable for performing any power optimization in the register-transfer level and proves the usability of the used methodology, and is in line what one can expect from an RTL power estimation, as for example in scientific publication [26] an average of 13 % accuracy has been demonstrated.

The RTL power reduction methodology yielded potential total power saving estimates between 10 % and 29 % and dynamic power savings between 13 % and 43 % for the three test design, which would be a significant improvement. And even if only 10 of the reduction opportunities were to be implemented, 16 % of the power dissipation would be reduced for the smallest test case. The results of a power reduction are always challenging to analyze as the power savings are highly dependet on the design and coding style, as well as how much there is room for power reduction. Due to this, comparing the power reduction results to another work is not very sensible, as the settings would be much different and the results would not be comparable with each other. As discussed in Chapter 2, the future trends in circuit technology seem to only increase the importance of dynamic power over static power dissipation, which justifies the power reduction results improving only on the dynamic power.

The aim of this thesis was to study the methodology for register-transfer level power estimation and reduction for digital system-on-chip building blocks based on a commercial RTL power tool. The true motivation behind this was to study and prove the maturity and usefulness of a commercial RTL power tool in real-life use cases. Overall, the methodology and results presented fulfill the requirements set by the aim and scope of the thesis as the average error of the RTL power estimation was proven to be in a suitable level for performing power reductions and meaningful power saving opportunities were found for each test design. And as the information and files required for the setting up the methodologies are present in a typical industry design flow before synthesis, the methodologies can be deployed without any major disruptions to the original design flow.

One criticism to the performed work considering the power reduction methodology is that the suggested power reduction opportunities were not actually implemented and validated, only the estimates of saved power were presented. The implementation was left out of the scope due to the lack of resources as it would have required effort from multiple designers. This is one potential further development of the work, as the results could be further studied and validated by implementing the proposed power reduction opportunities to the test design and running RTL synthesis and gate-level power analysis. And then comparing the gate-level power analysis results between non-power-optimized and power-optimized designs would be possible, which would further validate the power savings and measure more accurate power savings. Also, if the power reduction opportunities would be implemented, one could run a logical equivalence check with the original non-optimized design and study if the power reduction opportunities would alter the functionality of the original design.

Compared to analytical power estimation methods presented in Chapter 3, the used estimation methodology is much more accurate and offers very detailed power analysis features. But the analytical methods are much faster and they can be performed with more limited information, i.e. much early in the design phase, as the used methodology requires synthesis-like setup. One could study how analytical power estimation methods would perform more early in the design flow compared to the later register-transfer level power estimation used in this work. It should be noted that the power estimation setup in this thesis was very simple and did not contain any manual fine-tuning nor calibration. In future work, one could study how performing a calibration from physical design level with the RTL power tool could improve the power estimation accuracy. Furthermore, one could also try to manually fine-tune the parameters of the power estimation setup to further improve the accuracy.

The defined RTL power estimation and reduction methodologies could be also used for RTL designs generated by high-level synthesis (HLS) tools. As a future work, one could study how the presented methodologies perform with an RTL design generated by HLS tool. HLS tools generate an HDL code description from high-level models, which is typically made by using a programming languages like C++ or SystemC. The nature of HDL code and design structures generated by the HLS tools are very different compared to HDL code written by hand. Using an RTL design generated by HLS tool could yield much different results compared to hand written RTL design. Also, the methodologies could potentially offer faster power iteration of the HDL models as the power information is available earlier in the design process.

7. SUMMARY

This thesis is a study of register-transfer level power estimation and reduction methodology for digital system-on-chip building blocks based on a commercial state-of-the-art RTL power tool. The theory of power dissipation in CMOS circuits, background to RTL power estimators and common power reduction methods are discussed. The case study part presents the results and methodology of RTL power estimation and reduction performed with a commercial RTL power tool.

The power dissipation of a CMOS circuit can be divided into a dynamic and static power component, of which dynamic is related to individual transistors switching states and static is always present in the circuit when powered on. Dynamic power component can be further divided into capacitive and short-circuit components. Static component is a sum of gate-oxide tunneling, reverse-biased diode, subthreshold, punchthrough and gate induced drain leakage current.

Power estimation methods can be divided into dynamic and static methods, of which dynamic methods are simulative methods and static non-simulative methods. Register-transfer level power estimation methods fall under three categories, which are analytical methods, macro-modeling methods and fast synthesis methods. Complexity-based analytical methods try to estimate the power dissipation by the complexity of a design and activity-based methods try to model the dynamic nature of activity in a design. In power macro-modeling schemes, power models are constructed for each component of the design and power dissipation can be calculated by monitoring the activity via simulation or probabilistic manners. Fast synthesis methods perform a limited low-effort using the standard technology libraries, which produces a low-effort netlist than can be used in gate-level power simulators.

As power dissipation, power reduction methods can be divided into dynamic power reduction methods and static power reduction methods. Dynamic power reduction methods try mostly to eliminate redundant switching activity in a design, which the clock gating is the most important and widely used dynamic power reduction method. Static power reduction methods try to eliminate the leakage currents in a design. The most important one is power gating, where whole parts of the design are powered down when possible, to reduce static power dissipation. Power reduction methods utilized in the higher design abstraction levels have much higher impact on the total power dissipation than what the methods in lower abstraction level have.

A commercial state-of-the-art RTL power tool was used for performing RTL power estimation and RTL power reduction. The setup for RTL power requires RTL design files, power-characterized technology libraries, various synthesis and technology parameters defined and simulation activity from an RTL simulation. The RTL power estimation and reduction were performed onto three test RTL design of different sizes – small, medium and large. The deviation between RTL and gate-level power estimation was the average of 11% and potential RTL power saving estimates were between 10 % and 29 %. Based on the measurement results, the final conclusion is that the methodology with the commercial state-of-the-art RTL power tool for RTL power estimation and reduction is valid and mature.

8. REFERENCES

[1] Haron Z. & Hamdioui S. (2008) Why is CMOS scaling coming to an END? In: Design and Test Workshop, 2008. IDT 2008. 3rd International, December 20 – 22, Monastir, Tunisia.

[2] Pollack F. (1999) New Microarchitecture Challenges in the Challenges in the Coming Generations of CMOS Process Technologies. In: 32nd annual International Symposium on Microarchitecture, November 15 – 18, Haifa, Israel.

[3] Preeti P., B. V. N. Silpa, Aviral S. & Krishnaiah G. (2010) Power-efficient System Design. Springer.

[4] Lambert S., Heddeghem W., Vereecken W., Lannoo B., Colle D. & Pickavet M. (2012) Worldwide electricity consumption of communication networks. Department of Information Technology, Ghent University, Gent, Belgium.

[5] Evans D. (2011) The Internet of Things – How the Next Evolution of the Internet is Changing Everything. Cisco.

[6] Jalan A. & Khosla M. (2011) Analysis of Leakage Power Reduction Techniques in Digital Circuits. In: India Conference (INDICON), 2011 Annual IEEE, December 16 – 18, Hyderbad, India.

[7] Sarwar A. (1997) CMOS Power Consumption and C_{pd} Calculation. Texas Instruments.

[8] Odland K. (2012; Last accessed June 29, 2015) Designing low-energy embedded systems from silicon to software. URL: http://www.edn.com/design/systems-design/4402309/Designing-low-energy-embedded-systems-from-silicon-to-software

[9] Piguet C. (2005) Low-Power Electronics Design. CRC Press.

[10] Panda P.R., Silpa B.V.N., Shrivastava A. & Gummidipudi K. (2010) Power-efficient System Design. Springer.

[11] Rabaye J, Chandraksan A. & Nikolic B. (2003) Digital Integrated Circuits 2nd Edition. Pearson Education.

[12] Veendrick H. (1984) Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits. IEE Journal of Solid-State Circuits, Volume 19, Issue 4.

[13] Chaudhry A. (2013) Fundamentals of Nanoscale Field Effect Transistors. Springer.

[14] Nian Y., Henson K., Hauser R. & Wortman J. (1999) Modeling Study of Ultrathin Gate Oxides Using Direct Tunneling Current and Capacitance–Voltage Measurements in MOS Devices. IEEE Transactions on Electron Devices, Volume 46, Issue 7.

[15] Patel N. (April 29, 2013; Last accessed June 2, 2015) Grasping the low power fundamentals. EET India. URL: http://www.eetindia.co.in/ART_8800684306_1800000_TA_9f4b574b.HTM

[16] Roy K., Mukhopadhyay S. & Mahmoodi-Meimand, H. (2003) Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits. Proceedings of the IEEE, Volume 91, Issue 2.

[17] Jones R. (2004; Last accessed January 20, 2016) Modelling and Design Techniques Reduce 90 nm power. EE Times. URL: http://www.eetimes.com/document.asp?doc_id=1217859

[18] Kawa J. & Biddle A. (2012, Last accessed January 5, 2016) FinFET: The Promises and the Challenges. In: Synopsys Insight Newsletter, Issue 3, 2012. URL: https://www.synopsys.com/COMPANY/PUBLICATIONS/SYNOPSYSINSIGHT/Pages/Art2-finfet-challenges-ip-IssQ3-12.aspx

[19] International Technology Roadmap for Semiconductors, 2011 Edition, System Drivers. (2011) International Technology Roadmap for Semiconductors (ITRS), p 12.

[20] Pedram M. (1999) Low Power Design Methodologies and Techniques: An Overview, Department of EE-Systems, University of Southern California, Los Angeles, CA, USA.

[21] Yaseer D. & Teresa R. (2006) Statistical Power Estimation for IP-Based Design. In: IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on Industrial Electronics, November 6 – 10, Paris, France.

[22] Ravi S., Raghunathan A. & Chakradhar S. (2003) Efficient RTL Estimation for Large Designs. In: VLSI Design, 2003. Proceedings. 16th International Conference on, January 4 – 8, New Delhi, India.

[23] Paul L. (1996) High-Level Power Estimation. In: ISLPED '96 Proceedings of the 1996 international symposium on Low power electronics and design, August 12- 14, Monterey, CA, USA.

[24] Müller-Glaser K., Kirsch K. & Neusinger K. (1991) Estimating Essential Design Characteristics to Support Project Planning for ASIC Design Management. University of Erlangen-Nürnberg, Erlangen Germany.

[25] Farid N. & Mahadevamurt N. (1996) Towards a High-Level Power Estimation Capability. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 15, Issue 6.

[26] Bruno M., Macii A. & Poncino M. (2005) RTL power estimation in an HDL-based design flow. Computers and Digital Techniques, IEEE Proceedings, Volume 152, Issue 6.

[27] Llopis P. & Goossens K. (1998) The Petrol Approach to High-Level Power Estimation. In: 1998 International Symposium on Low Power Electronics and Design, August 10 – 12, Monterey, CA, USA.

[28] Agrawal V.D. & Srivaths R. (2007) Low-Power Design and Test: Logic-Level Power Estimation. Auburn University, Auburn, AL, USA.

[29] Arora M. (2011) The Art of Hardware Architecture: Design Methods and Techniques for Digital Circuits, Springer.

[30] Shinde J. (2011) Clock gating — A power optimizing technique for VLSI circuits. In: India Conference (INDICON), 2011 Annual IEEE India Conference, December 16 – 18, Hyderabad, India.

[31] Bhutada R. & Manoli Y. (2007) Complex Clock Gating with Integrated Clock Gating Logic Cell. In: Design & Technology of Integrated Systems in Nanoscale Era, 2007. DTIS. International Conference on Design & Technology of Integrated Systems in Nanoscale Era, September 2- 5, Rabat, Marocco.

[32] Garret D., Stan M. & Dean A. (1999) Challenges in Clockgating for a Low Power ASIC Methodology. In: 1999 International Symposium on Low Power Electronics and Design, August 17, San Diego, CA, USA.

[33] Akilla M. & Gopalakrishnan L. (Last accessed January 7, 2016) Techniques for Improving QoR Using analyze_datapath_extraction. Synopsys Inc. URL: http://www.synopsys.com/Company/Publications/DWTB/Pages/dwtb-analyze-datapath-extraction-2014Q2.aspx

[34] Li L., Ken C., SeongMo P. & MooKyung C. (2009) Selective Clock Gating by Using Wasting Toggle Rate. In: Electro/Information Technology, 2009. EIT '09. IEEE International Conference on Electro/Information Technology, June 7 – 9, Windsor, Ontario, Canada.

[35] White M. A. (Last accessed on January 19, 2016) Synopsys Insight: Reducing Power with Advanced Synthesis. Synopsys Inc. URL:

https://www.synopsys.com/Company/Publications/SynopsysInsight/Pages/Art2-reduceadvsynthesis-IssQ4-11.aspx

[36] Cong J., Liu B. & Zhang Z. (2009) Behavioral-Level Observability Don't-Cares and Application to Low-Power Behavioral Synthesis. Cornell University, New York, USA, p. 1.

[37] Fraer R., Kamhi, G. & Mhameed M.K. (2008) A New Paradigm of Synthesis and Propagation of Clock Gating Conditions. In: 45th ACM/EDAC/IEEE Design Automation Conference, June 8-13, Anaheim, CA, USA.

[38] Raja T., Agrawal, V. & Bushnell, M. (2005) Variable Input Delay CMOS Logic for Low Power Design. In: VLSI Design, 2005. 18th International Conference on VLSI Design, January 3 -7, Kolkata, India.

[39] Raghunathan A., Dey S. & Jha, N.K. (1999) Register Transfer Level Power Optimization with Emphasis on Glitch Analysis and Reduction. IEEE Transactions on Computer-Aided Sesign of Integrated Circuits and Systems, vol. 18, no. 8

[40] Munch M., Wurth B., Mehra R., Sproch, J. & Wehn N. (2000) Automating RT-Level Operand Isolation to Minimize Power Consumption in Datapaths. In: Design, Automation and Test in Europe Conference and Exhibition 2000, March 27 – 30, Paris, France.

[41] Alidina M., Monteiro J., Devadas S., Ghosh A. & Papaefthymiu M. (1994) Precomputation-Based Sequential Logic Optimization for Low Power. In: Proceedings of the 1994 IEEE/ACM International Conference on Computer-Aided Design, November 6 – 10, San Jose, CA, USA.

[42] Lackey E., Zuchowski S., Bednar R., Stout W., Gould W. & Cohn M. (2002) Managing Power and Performance for System-on-Chip Designs using Voltage Islands. In: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design, November 10 -14, 2002, San Jose, CA, USA.

[43] Yunlong Z, Qiang T., Li L, Wei W. Ken C. JongEun J., Hyobin J. & Si-Young A. (2012) Automatic Register Transfer level CAD tool design for advanced clock gating and low power schemes. In: SoC Design Conference (ISOCC), 2012 International, November 4- 7, Jeju Island, Korea.

[44] Bhat S., Srigowri S., Rao V. & Pai V. (2014) Implementation of Dynamic Voltage and Frequency Scaling for System Level Power Reduction. In: Proceedings of International Conference on Circuits, Communication, Control and Computing (I4C 2014), November 21 – 22, Bangalore, India.

[45] Benini L., De Micheli G. & Vermeulen F. (1998) Finite-State Machine Partitioning for Low Power. In: Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, May 31 – June 3, Monterey, CA, USA.

[46] Stan R. & Burleson P. (1995) Bus-Invert Coding for Low-Power I/O. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume 3, Issue 1), IEEE.

[47] Rjoub A. & Almanasrah H. (2010) Low Leakage Multi-V_{th} Technique for Sequential Circuits at Transistor Level in Nanotechnology. In: 2010 17th IEEE International Conference on Electronics, Circuits and Systems - (ICECS 2010), December 12 -15, Athens, Greece.

[48] Yen-Te Ho & Ting-Ting Hwang. (2004) Low Power Design Using Dual Threshold Voltage. In: Proceedings of the 2004 Asia and South Pacific Design Automation Conference, January 27 – 30, Yokohama, Japan.

[49] (2013) IEEE Standard 1801 - 2013 for Design and Verification of Low-Power Integrated Circuits. IEEE, New York, NY, USA.

[50] Keating M., Flynn D., Aitken R., Gibbons A. & Shi K. (2007) Low Power Methodology Manual For System-on-Chip Design. Springer.

[51] Manuzzato A., Campi F., Rossi D., Liberali V. & Pandini D. (2013) Exploiting body biasing for leakage reduction: A case study. In: 2013 IEEE Computer Society Annual Symposium on VLSI, August 5 - 7, Natal, Brazil.

[52] Ye Y., Borkar S. & De V. (1998) A New Technique for Standby Leakage Reduction in High-Performance Circuits. In: 1998 Symposium on VLSI Circuits Digest of Technical Papers, June 11 – 13, Honolulu, HI, USA.

[53] PowerArtist™: RTL Design-for-Power. (2014) ANSYS Inc. In: Design and Automation Conference 2014, June 1 – 5, San Francisco, CA, USA.

[54] PowerArtist™, product page. (Last accessed on January 20, 2016) Apache Design Inc., a subsidiary of ANSYS Inc. URL: https://www.apache-da.com/products/powerartist

[55] SpyGlass® Power, product page. (Last accessed on January 20, 2016) Atrenta Inc. URL: http://www.atrenta.com/pg/6/

[56] Calypto PowerPro, product page. (Last accessed on January 20, 2016) Calypto Design Systems Inc. URL: http://calypto.com/en/products/powerpro/powerpro_overview#productinfo

[57] Joules RTL Power Solution, product page. (Last accessed on January 20, 2016) Cadence Design Systems Inc. URL: http://www.cadence.com/products/ld/joules_rtl_power/pages/default.aspx

[58] Lahti J. (2013) Digital Techniques 3: Introduction. Lecture material. University of Oulu, Department of Electrical Engineering, Oulu, p. 7.