NTNU

Norwegian University of
Science and Technology

Clock Tree Power Analysis

Knut Austbø

Master of Science in Electronics
Submission date: June 2016
Supervisor: Per Gunnar Kjeldsberg, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Project Assignment

Candidate name: Knut Austbø

Assignment title: Clock Tree Power Analysis

Assignment text:

In some chips and power modes as much as half of the chip's power consumption can be associated with the clock trees, i.e. the circuit structures used to distribute the clock signal. A standard practice to optimize for power is to gate clocks, that is, to turn off clocks when they are not needed. There are challenging trade-offs between how and where to do the clock gating and what power performance that gives for the different power modes. This Master's project builds on using a power-benchmarking framework, and enhance it with dedicated clock power classification. A number of different clock gating strategies should be benchmarked on a representative test case, and a detailed assessment of the results is expected. The student needs to know basic digital design and power analysis in digital logic. Initial understanding of clock tree structures is preferable.

Responsible professor: Per Gunnar Kjeldsberg, IET, NTNU.

Supervisors: Johnny Pihl, Nordic Semiconductor ASA.

Co-supervisors: Jan Egil Øye and Are Aarseth, Nordic Semiconductor ASA.

Abstract

The buffered clock tree structure is commonly used to distribute the clock signal to the memory elements in digital circuits. Since the clock signal is used as a temporal reference, it has to be distributed to the registers with decent timing characteristics and low skew. In order to achieve this, buffers and inverters are inserted in the clock tree, typically by a synthesis tool.

The clock tree is a major contributor to the power consumption. This is a result of a combination of high switching activity, due to the high frequency of the clock signal, and high total load in the buffers, registers and other cells. In order to reduce the power, clock gates are inserted in the clock tree, disabling the clock signal for unused logic. In this project, clock skew, buffer area and power consumption are considered the most important clock tree parameters.

The clock-tree synthesis functionality of the Synopsys® IC Compiler™ tool has been explored. The goal of this exploration is to illustrate trade-offs and examine how the different options affect the synthesis results. Synthesis results on a generated test design and a real Bluetooth Smart design show cost reduction when the skew target is increased from the default zero skew target. By increasing the skew target to 0.5 ns for the generated design, the buffer area is reduced by 2 168 μm^2 (58%), while the dynamic and static power are reduced by 0.41 mW (10%) and 0.94 nW (21%), respectively. The reductions are less significant in the Bluetooth Smart design, however, comparable in absolute values. By increasing the skew target to 1.0 ns, the buffer area is reduced by 1 284 μm^2 (21%), while the dynamic and static power consumption are reduced by 0.19 mW (3.3%) and 0.7 nW (0.5%), respectively. The increased clock skew of the Bluetooth Smart design does not show any significant increase in hold time violations cost.

A multi-level module-level clock gating strategy has been implemented in the Bluetooth Smart design. In this strategy, an additional layer of clock gates has been inserted between the existing module-level and the clock source. Results indicate a dynamic power reduction of about 0.04 mW (18%) in a selected low-power scenario, without any significant cost increase, compared to the unmodified structure. However, the control circuit for the new level has not been implemented. Further work is therefore needed to determine if the tested strategy is beneficial.

Sammendrag

Bufrede klokketrær er en mye brukt strategi for å distribuere klokkesignalet til minneelemntene i en digital krets. Ettersom klokkesignalet blir brukt som en tidsreferanse, er det viktig at signalet blir distribuert med god karakteristikk og liten tidsforskyvning. For å oppnå dette settes bufre og invertere inn i trestrukturen. Dette er vanligvis gjort av et synteseverktøy.

Klokketreet står for en stor andel av det total effektforbruket i et digital system. Årsaken til dette er en kombinasjon av høy vekselrate, på grunn av høy klokkefrekvens, og stor totallast i klokketreets bufre, registre og andre byggeblokker. Typisk blir klokkeporter satt inn i klokketreet for å redusere effektforbruket, ved å slå av klokkesignalet til ubrukt logikk. I dette prosjektet er klokkeforskyvningen, bufferarealet og effektforbruket ansett som de viktigste parameterne.

Klokkesyntesen i verktøyet Synopsys® IC Compiler™ har blitt undersøkt. Målet med denne undesøkelsen er å observere vekselvirkninger, samt å utforske hvordan innstillingene påvirker synteseresultatet. Synteseresultater fra et kunstig design og et reelt «Bluetooth Smart»-design viser at man kan oppnå reduserte kostnader ved å tillate høyere klokkeforskyving enn verktøyets standardmål (ingen forskyvning). Ved å øke forskyvningen til 0.5 ns oppnås en reduksjon i buffer areal på 168 μm^2 (58%), samt en reduksjon i dynamisk- og statisk effektforbruk på henholdsvis 0,41 mW (10%) og 0,94 nW (21%), i det kunstige designet. For det reelle designet er reduksjonene mindre betydningsfulle, men fortsatt sammenlignbare i absoluttverdi. Ved å tillate 1.0 ns forskyvning reduseres bufferarealet med 1 284 μm^2 (21%), mens det dynamiske- og statiske effektforbruket reduseres med henholdsvis 0,19 mW (3,3%) og 0,7 nW (0,5%). Undersøkelser viser at den økte forskyvingen ikke gir i høyere holdetidskostnad for det reelle designet.

En flernivå klokkeportstrategi på modulnivå har blitt implementert i det reelle «Bluetooth Smart»-designet. I denne strategien er et nytt nivå med klokkeporter innført mellom de eksisterende klokkeportene og klokkekilden. Resultater viser en effektreduksjon på omtrent 0,04 mW (18%) i en utvalgt strømsparingstilstand, dette uten noen betydningsfulle kostnadsøkninger. Styringskretsen for det nye nivået er imidlertid ikke implementert. Videre arbeid er derfor nødvendig for å avgjøre om denne strategien er lønnsom.

Preface

This Master's Thesis has been written as a part of my 5-year M.Sc. degree in Electronics, with specialization in design of digital systems, at the Norwegian University of Science and Technology (NTNU). This thesis is a continuation of a previous semester project, where a clock-tree analysis tool was created to aid the designer in the clock-tree synthesis process [1]. The goal of this thesis has been to use features of the analysis tool to examine the clock tree synthesis in Synopsys® IC Compiler™, as well as evaluate a clock gating strategy with multiple levels of module-level clock gates.

Nordic Semiconductor ASA has contributed to this Thesis with a workplace and a computer. In addition, I have been given access to a local server with licenses for the required Synopsys® tools. On this server, I have been able to work without any major interruptions and without interfering with design work by Nordic Semiconductor employees. The only issue during this project was a failure of a hard disk controller. This failure lead to about one week of downtime, before the component was replaced. However, this issue was not critical for the progress of the Thesis. I would like to thank Nordic Semiconductor and my supervisors at Nordic Semiconductor, Jan Egil Øye, Are Aarseth and Johnny Pihl, for the help and feedback during this project.

During the semester, I have also been doing some part time work for Nordic Semiconductor. Among the different tasks, I have finalized the analysis tool from the previous project and created a user guide. The tool is now usable by Nordic Semiconductor employees.

I would like to thank Professor Per Gunnar Kjeldsberg for good academic supervision throughout the semester. In addition, I would like to thank my fellow students at NTNU for a good working environment. Lastly, I would like to thank my family, friends and girlfriend for the support during the project. A special thanks to my brother Bjørn, for helping me proofreading this Thesis.

Knut Austbø

Trondheim, 17 June 2016

Table of Contents

List of Figures

List of Tables

Abbreviations

ADC	Analogue-to-digital converter
CMOS	Complementary metal–oxide-semiconductor
CPU	Central processing unit
CTS	Clock tree synthesis
DAC	Digital-to-analogue converter
DFT	Design for test
DRC	Design rule constraints
ECO	Engineering change orders
ESD	Electrostatic discharge
FPU	Floating point unit
HDL	Hardware descriptive language
I/O	Input/output
ICG	Integrated clock gate
IET	Department of Electronics and Telecommunications
IoT	Internet of Things
ISM	Industrial, scientific and medical
Mbps	Mega bit per second
MOSFET	Metal–oxide–semiconductor field-effect
MUX	Multiplexer
NAND	Not and (logical function)
NMOS	N-channel MOSFET
NTNU	Norwegian University of Science and Technology
PLL	Phase locked loop
PMOS	P-channel MOSFET
QoR	Quality of results
RTL	Register transfer level
SoC	System on chip
XOR	Exclusive or (logic function)

1 Introduction

One of the leading trends in today's electronics is the Internet of Things (IoT), a trend where all kinds of devices are connected in networks. This is a rapidly growing trend with estimates of about 40 billion IoT devices by 2020 [2, 3]. The Internet of Things has emerged as a result of progress in chip technology. Devices are becoming increasingly smarter and power efficient. In addition, new System on Chip (SoC) solutions with integrated radio enable wireless communication with smart devices.

The power consumption is one of the most important parameters of IoT devices. Low power consumption has several benefits, e.g. increased the time between each battery recharging or battery replacement. Therefore, modern devices have several features to reduce the power consumption. Dependent on the application, the device operates in different activity modes. Typically, the device spends most of its time in an idle or low power mode, from which it wakes up to perform a task in a high activity mode. In order to do make the device as efficient as possible, it is therefore important to consider the power consumption in several different modes.

One of the major contributors to the power consumption is the clock tree. The task of the clock tree is to synchronize the digital circuit, as the clock signal is used as a temporal reference in the circuit. In other words, the clock edge has to arrive simultaneously at every memory element in the design. In order to distribute the clock signal across the chip with decent timing characteristics, a synthesis tool is used to implement a balanced clock tree. This tool inserts buffers in the clock tree in order to improve timing performance. However, these buffers increase the total load in the clock tree. In addition, due to its high frequency, the clock signal is typically the fastest toggling signal in the circuit. The combination of high toggle rate and high load is resulting in very high power consumption.

Clock gates are inserted in the clock tree to reduce the power consumption. These gates are used to stop propagation of the clock signal to unused circuitry, thus reducing power consumption. However, inserting clock gates increases the complexity of the clock tree and makes it more difficult for the synthesis tool to achieve good timing performance.

1.1 Project Delineation and Contributions

In a previous project, it was observed inexplicably large variations in the results after clock tree synthesis in Synopsys® IC Compiler™ [1]. This observation indicates the need of a

thorough exploration of the synthesis tool. The target of this exploration is to obtain a better understanding of the tool functionality. It is considered important to find out how the synthesis settings influence the synthesis results and potential performance trade-offs. In addition, the effects of the design size and complexity should be examined. The insertion of clock gates in the clock tree lead to increased clock complexity, and can cause imbalance in the clock tree. An assessment of the cost of inserting clock gates is therefore required.

Based on the results of the synthesis exploration, a set of synthesis recommendations has been developed. These recommendations cover the synthesis process and its settings from preparing the design for synthesis to post synthesis debugging. The purpose of this guide is to aid the designer in the synthesis process and enable him to achieve good synthesis results.

In order to minimize the power consumption in the clock tree, a multi-level module-level clock gating strategy has been examined for a real Bluetooth Smart design. The idea behind this strategy is to extend the current single-level module clock gating with an additional level closer to the clock root. However, due to the limited time of this project, the work has been limited to the synthesis stage only. This means that the control circuitry for the new level has not been implemented, as this is done in RTL.

The main contributions in this project can be summed up in the following three parts:

1. Exploration of the clock tree synthesis and optimization functionality in Synopsys® IC Compiler™.
2. Creation of a set of recommendations for the synthesis process in Synopsys® IC Compiler™.
3. Examination and evaluation of a purposed multi-level module-level clock gating strategy for a real Bluetooth Smart design.

1.2 Report Structure

This report has been divided into the following chapters:

Chapter 2: Theory presenting the useful background theory for this project. Clock trees and CMOS power consumption are examples of presented topics.

Chapter 3: Previous Work and Design Tools presenting relevant research and literature. In addition, the tools used in this project are described.

Chapter 4: Methodology describing the basic methodology in this project. The two test designs used in this project are presented.

Chapter 5: Clock Tree Synthesis Exploration presenting the methodology and results of exploring the functionality of the clock tree synthesis in Synopsys® IC Compiler™. A detailed assessment of the results is also provided.

Chapter 6: Clock Gating Experiment examining the benefits and drawbacks of implementing an additional level of module-level clock gating.

Chapter 7: Clock Tree Synthesis Recommendations presenting a set of useful recommendations for clock tree synthesis in Synopsys® IC Compiler™.

Chapter 8: Conclusion summarizing the results of this project, in addition to suggesting topics for further work.

Chapter 9: Bibliography listing references used in this report.

Additional information, e.g. raw data from synthesis results, are attached to this report in the Appendices A to D.

2 Theory

This chapter presents the background theory in this project. First, common terminology, used throughout this report is presented in Section 2.1. Then, a short introduction to the Bluetooth Smart technology is given in Section 2.2. CMOS power consumption is described in Section 2.3. Section 2.4 presents the buffered clock-tree structure, while clock gating strategies, used to reduce the power consumption of this structure, are presented in Section 2.5.

2.1 Terminology

Expressions commonly used throughout this report are presented below:

Cell: Cells are the building blocks of the circuit design. A cell can consist of multiple gates and sub-cells.

Net: The nets are the interconnections between the cells. A net is driven by one cell output only, but can be connected to several cell inputs.

Fan-out: The fan-out of an output pin is the collection of cell inputs connected to the output net.

2.2 Bluetooth Smart

Bluetooth Smart, or Low Energy (LE), is a wireless technology located in the 2.4 GHz ISM band [4, 5]. The technology has 40 channels with 2 MHz bandwidth and uses adaptive channel hopping to avoid interference from other 2.4 GHz technologies such as Wi-Fi, ZigBee and regular Bluetooth. Bluetooth Smart is especially designed for low power applications and offers a bit rate of 1 Mbps with a low latency of 3 ms. The output power is limited to 10 mW (10 dBm) which gives an open field range up to 100-150 meters [5].

Bluetooth Smart has a low data throughput [5]. Therefore, it is not suited for transferring large quantities of data, e.g. video or audio. Instead, it is used to transmit states, e.g. small quantities of data such from sensors, such as temperature and heart rate. Due to its low power functionality, Bluetooth Smart is well suited for many IoT applications. In these applications, the device spends most of the time asleep, only waking up to transmit its state. Typically, the device operates on battery, and low power consumption is very important.

2.3 Power Consumption

The power consumption in digital CMOS can be divided into the dynamic power consumption related to switching activity, and the static consumption that is activity independent [6].

2.3.1 Dynamic Power

The dynamic power consumption is divided into two groups: switching power and internal power [6]. The switching power is the average power dissipated by the nets and is the main contribution to the total dynamic power consumption. Equation (1) shows a formula of the switching power for a single net, where V_{DD} is the supply voltage and C_L is the net load capacitance. The switching activity is given by the clock frequency f_{CLK} and the switching factor α. This factor is the average number of "0-to-1" and "1-to-0"-transitions by the net in one clock cycle.

$$P_{SW} = \frac{\alpha}{2} C_L V_{DD}^2 f_{CLK} \tag{1}$$

The secondary source to dynamic power consumption is the internal power, P_{IN}. This is the average dynamic power dissipated internally in a cell. Activity on the inputs of the cell causes switching in the internal nodes, and creates short circuit currents through the cell. For a short period when a net is transitioning, both the PMOS and the NMOS transistors are conducting, resulting in a short circuit current from V_{DD} to ground. The power dissipated by this effect, P_{SC}, is given in Equation (2), where t_{TR} is the transition time and $\overline{I_{SC}}$ is the average short circuit current.

$$P_{SC} = \alpha t_{TR} \overline{I_{SC}} V_{DD} f_{CLK} \tag{2}$$

The impact of the internal power is dependent on the complexity of the cell. Simple cells, like an inverter or a two-input NAND gate, have few transistors and internal nets. In these cells, the internal power is typically less than 10 % of the switching power of the output [6]. However, for complex cells with several levels, such as multistage clock buffers, the internal power may exceed the output net power.

In order to calculate the total dynamic power consumption, the switching power is summed over all nets and added to the sum of the internal power of all cells, as shown in Equation (3).

$$P_{DYN} = \sum_{n}^{Nets} P_{SW,n} + \sum_{c}^{Cells} P_{IN,c} \tag{3}$$

2.3.2 Static Power

Similar to the internal power consumption, the static power consumption is also calculated per cell in the design, as shown in Equation (4). The cause of the static consumption is leakage currents I_{LK} in the cells, as the transistors do not work as perfect switches [6]. Although this consumption is switching independent, it may be dependent on the state of its input, output and internal nodes.

$$P_{ST} = V_{DD} \sum_{c}^{Cells} I_{LK,c} \tag{4}$$

The main sources of leakage currents in CMOS transistors are [6]:

- Sub-threshold leakage: The current flowing from drain to source when operating in the sub-threshold region.
- Gate leakage: Oxide tunnelling and hot carrier injection is causing current flow from the gate to the substrate.
- Gate induced drain leakage: Current from the drain to the substrate due to high V_{DG} voltage.
- Reverse bias junction leakage: Leakage caused by carrier drift in the depletion region.

Due to technology scaling, the static power is becoming increasingly important. There are many strategies for reducing static power, e.g. power gating, Multi-V_T designs and transistor stacking. However, as the dynamic power consumption is the dominant in the clock tree, this project does not implement any strategies for reducing static power consumption.

2.4 Clock Trees

The task of the clock distribution network is to control the data flow in synchronous digital systems [7]. The clock is used as a temporal reference for movement of data. Therefore, it is very important that the clock is distributed, sharp and clean, to every part of its domain, without any major timing differences. This is a complex task as the clocks are the signals with the highest frequencies within the design, and both the fan-out and physical distance can be large. A single clock source might drive several thousand registers all across the chip. In addition, technology scaling is making this task even more difficult as the lines become more resistive as the dimensions are reduced.

There are several different ways to organize the clock network [7]. Some common structures are illustrated in Figure 2.1. This project focuses on buffered clock trees. In these trees, buffers are inserted at the clock source and along the clock paths, organized in a tree structure. The clock source is referred to as the root of the tree, while the registers (or sinks) are the leaf nodes. Clock buffers are inserted to improve reliability and timing characteristics, as described in Section 2.4.2. The most important characteristics of the clock trees are described in the following section.

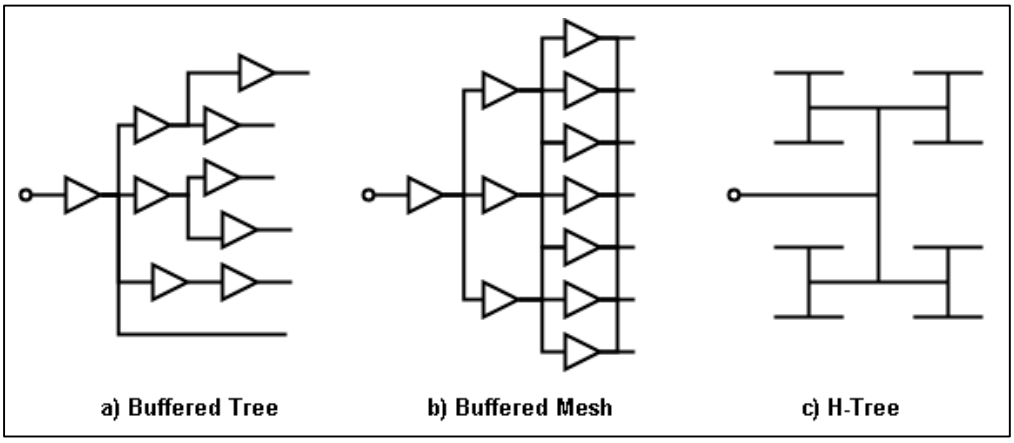

Figure 2.1: Common clock distribution structures.

The nodes in the tree are switching state twice every clock cycle, resulting in very high switching activity. However, the clock is not always needed on the entire chip at all time, as some logic might be unused and not all registers changes value. In order to reduce the power consumption, the clock can be turned off where it is not needed [6]. This can be done with Internal Clock Gating Cells (ICGs), described in Section 2.4.3. As presented in

Section 2.5, there are several strategies in placing the ICGs. Figure 2.2 shows an example of a clock tree with ICGs.

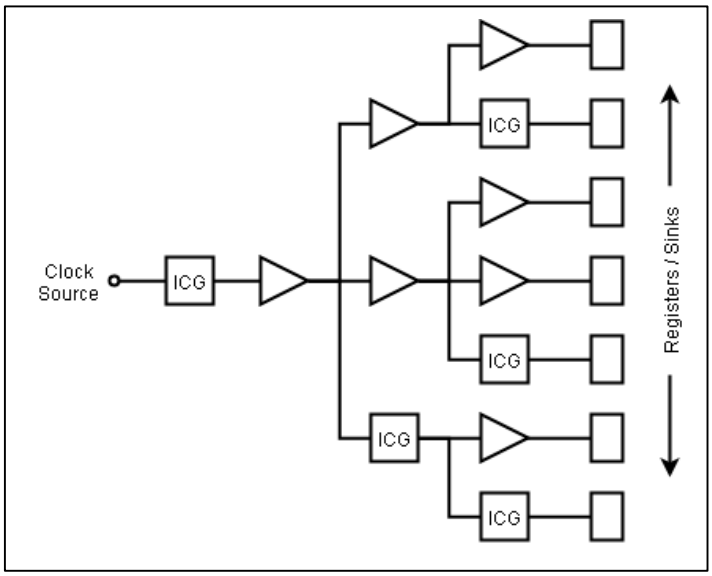

Figure 2.2: Buffered clock tree with ICGs.

An ICG has the ability to stop the clock signal from propagating and thereby cancel all switching activity and dynamic power consumption in its subtree. ICGs are commonly used to reduce the dynamic power consumption of the clock tree, but does not have any major effect on the static power consumption.

Power gating is a common method to reduce the static power consumption. In order to stop leakage currents, the power rail is switched off for unused logic. A side effect of this is that all register states are lost. To be able to keep the register contents, their state has to be saved in retention registers before the power is gated. The states can be reloaded once the power is restored. Power gating is not the focus of this project, and is therefore not further examined.

In some cases, the clock signal may be derived from other clocks domains. A slower clock can be created by using clock dividers, and a faster clock can be generated with phase-locked loops (PLLs). In both cases, the derived clock tree becomes a subtree of the original tree. A clock tree can also have several clock sources. However, only one source is active at a time, selected by a clock multiplexer (MUX). A typical example is the selection of an internal RC-oscillator versus a high precision external crystal oscillator as the clock source.

2.4.1 Characterization

In order to be able to characterize the performance of a clock tree, the following parameters are defined [1]:

Longest path (latency): This is a measure of the longest time delay from the clock source to any leaf node in the tree. A high latency is not desired, as it makes the clock unreliable and prone to timing mismatch. High latency is also an indication of high power consumption, as the clock tree is deep and has several stages of buffering.

Clock skew: The skew is a measure of the difference in timing between two leaf nodes. The clock skew is often denoted as the maximum global skew, the difference between the shortest and longest path. In an ideal case, the clock edge arrives simultaneously at all registers, i.e. zero skew. However, it is difficult and expensive to achieve zero clock skew. The clock skew affects the data path, and can cause race effects and limit the performance.

Transition time: The transition time is the time it takes for the clock signal to switch state. For reliability, sharp and clean edges, i.e. short transition times, are desired. The transition time should also be equal for the rising and falling edge. A short transition time also reduces the short circuit current, as described in Section 2.3.1. The drawback of low transition times is higher current peaks and noise.

Number of sinks: This parameter is equal to the total number of registers using the clock signal. It is used as a measure of the size and complexity of the clock domain.

Number of buffers: The number of buffers is the total number of buffers and inverters in the clock domain.

Number of ICGs: This parameter is the total number of clock gating cells in the clock domain.

Number of cells: The total number of cells includes all buffers, inverters, ICGs, multiplexers, and any other cells in the clock tree.

Sinks per buffer (SpB): This ratio provides an indication of both the efficiency of the buffering and the fan-out in the clock tree. The parameter can be used to compare clock trees of different size/number of sinks.

Buffer area (BA): The total buffer area is a measurement of the total area cost of the buffers inserted in the tree. Buffer area is also related to power consumption, as the buffers are a major source to power consumption in the clock tree.

Sinks per buffer area (SpBA): Similar to sinks per buffer, this parameter can be used to compare different clock trees. However, this parameter takes the buffer area into account.

Power consumption: The power consumption in the clock tree is a very important parameter. However, it can be very different in different modes. It is therefore important to consider different scenarios, and evaluate both the dynamic and static power consumption.

2.4.2 Clock Buffers and Inverters

Clock buffers are inserted into the tree structure to amplify the clock signal, reduce the propagation delay and sharpen the clock edges. Nets with high load react very slowly, which gives slow transitions. There are two primary sources to the load in the clock tree nets; the input capacitances of the gates connected to the net and the line load itself. The input capacitances become a problem when the fan-out is large. The total fan-out capacitance is equal to the sum of all the input capacitances, as shown in Equation (5) and can be of significant size when the fan-out is large. A single buffer can be inserted to drive the net, but this might require a very large buffer. Alternatively, the fan-out can be divided into several branches driven by multiple buffers. The optimal solution is dependent on the fan-out and the range of buffers available in the cell library.

$$C_{Fan-out} = \sum_{i}^{Fan-out} C_{IN,i} \tag{5}$$

The line load becomes a problem as the clock signal has to be transported a long distance. Long wires have high capacitive and resistive load. This effect is an increasing problem due to technology scaling. The dimensions and cross section of the lines are shrinking, but the line lengths are not decreasing accordingly. In order to get acceptable timing characteristics, several buffers might be required along long lines.

Inverters and buffers are also inserted to increase delay in a clock branch, exploiting the propagation delay of the buffer. This technique is used to lengthen short clock paths in order to balance the clock tree and reduce the clock skew.

Typically, a clock buffer consists of an even number of inverters. Hence, standalone inverters can be used in the clock tree similar to the buffers. The drawback of using inverters is, however, that the clock phase is inverted. The inverters must therefore consider edge dependent cells in the clock tree, e.g. registers, clock dividers and ICGs.

The selection of buffers and inverters is dependent on the technology's cell library. Typically, a range of sizes is available, from small and weak buffers to large buffers with high drive strength. In order to get good signal characteristics, the output resistance of the selected buffer should be much larger than the resistance of the driven net [7].

2.4.3 Clock Gating Cells

As previously mentioned, the switching activity in a clock tree is high as the clock signal toggles twice every cycle ($\alpha = 2$). However, depending on the application, parts of the chip may not need the clock at a given point in time. In these inactive parts, it is desired to reduce the activity to $\alpha = 0$, and thereby eliminate the dynamic consumption. This is done with ICGs similar to the one shown in Figure 2.3. In this schematic, *CLK* and *ENCLK* are the input and gated output clock signals, respectively. The enable signal, *EN*, is set to logic 1 in order to enable propagation of the clock signal through the ICG. The internal signal *ENint* controls the output clock, and is latched while *CLK* is low, as shown in Table 2.1. This is done to avoid propagation of glitches from *EN* to *ENCLK*.

Table 2.1: Internal node *ENint* state table.

CLK	EN	ENint
0	0	0
0	1	1
1	-	ENint,prev

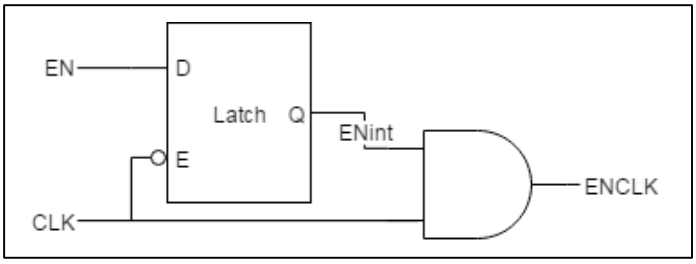

Figure 2.3: ICG schematic.

There are several different strategies of inserting the ICGs in the clock tree. This is presented in Section 2.5.

2.4.4 Registers

The registers are the sinks of the clock tree. These are sequential cells used to remember states of the system [8]. A commonly used register is the rising edge triggered D flip-flop. The input value of this flip-flop is sampled, stored and propagated to its output on the rising edge of the clock signal. For the sample to be correct, the input has to be stable for a setup time, t_{setup}, before the clock edge. In addition, the input has to remain stable for at least a hold time, t_{hold} after the clock edge. These timing requirements are illustrated in Figure 2.4. Violating the timing requirements can cause erroneous behaviour.

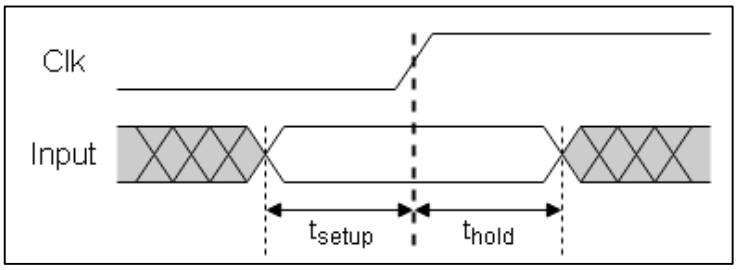

Figure 2.4: Hold and setup timing.

The registers come with various features. One of these is the scan feature, commonly used in design for test (DFT) [9]. With this feature, the registers can be organized in one or more scan chains. The chain is an alternative data path where the registers are connected as a long shift register with one input and output. The scan feature enables reading and writing to any register. This is a desirable feature in complex designs with several thousand registers.

As Figure 2.5 shows, a single multiplexer is used to implement the scan functionality [9]. The scan-mode enable signal, *SE*, selects the register input source. When *SE* is logic 1, the scan input, *SI*, is selected. Otherwise, the data input, *D*, is selected and the register is behaving as regular. However, regular output, *Q*, is also used as the scan output, *SO*. In the scan chain, the scan output is connected directly to the scan input of the next register in the chain, as illustrated in Figure 2.6.

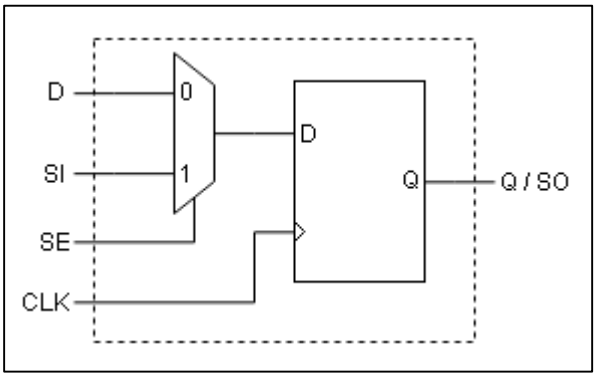

Figure 2.5: Scan register schematic.

The registers use the same clock signal for both regular and scan mode. However, an external source is typically used for the scan clock. The scan chains can contain registers from several clock domains, which gives a different clock structure than regular mode. Due to the short path in the scan chain, low skew is required in the scan clock to avoid hold time violations. Therefore, the clock system has to be optimized for both regular and scan mode.

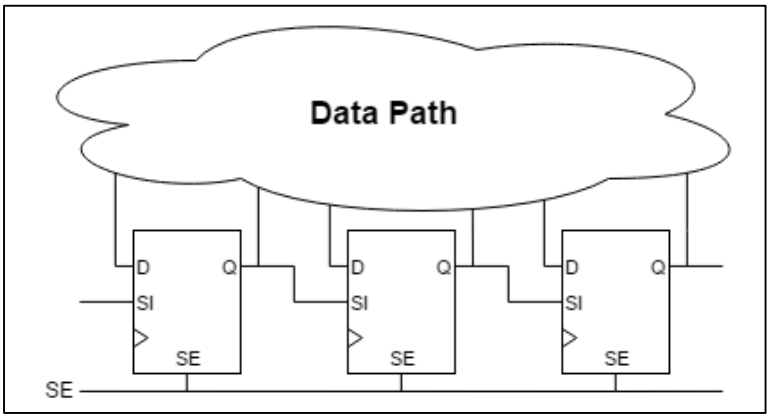

Figure 2.6: Scan chain connection.

2.5 Clock Gating Strategies

There are several strategies when it comes to inserting the ICGs in the clock tree. Design tools are able reduce the dynamic power consumption by inserting clock gates in the leaf levels as explained in Section 2.5.1 and 2.5.2. However, as the clock tree itself consumes a lot of power, ICGs can also be placed closer to the root. These clock gates are used to disable entire modules as described in Section 2.5.3.

2.5.1 Leaf-Level Clock Gating

Many registers are implemented with code similar to the Verilog HDL code in Figure 2.7, and already have an enable signal. This is often implemented with multiplexers as illustrated in Figure 2.7a. With this solution, the *n* registers are updated every cycle, but only given the new input value when the enable signal is logic 1. However, updating the register consumes power independent of the value being new or old. Unnecessary updates should therefore be avoided.

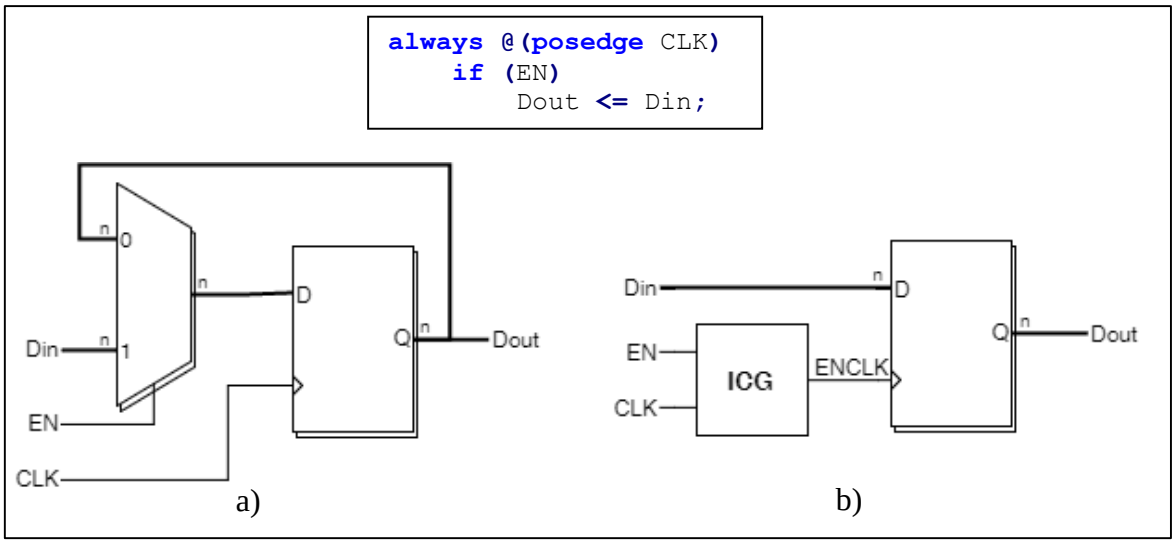

Figure 2.7: Clock gating at leaf level.

This is done with the schematic presented in Figure 2.7b. The ICG cell disables the clock signal when the enable signal is logic 0. Therefore, the register is only updated when the enable signal is logic 1, thus reducing the power consumption. Depending on the bit width, *n*, this clock gating strategy may enable area reductions, as *n* multiplexers can be replaced by a single ICG cell. The savings, of both power and area, are also dependant on the register activity and technology. Testing done in a 180 µm technology by Pokhrel, shows that gating is neither beneficial with respect to power nor area, for *n* less than three [6] [10].

Clock gating at the leaf level can be done by design tools and does not change the logic functionality of the circuit. A drawback of this method is that it increases the length of the clock path.

2.5.2 XOR Self-Gating

Many registers are implemented without the enable signal, as illustrated in Figure 2.8a. For these registers, the XOR self-gating technique can be used to gate the clock signal and reduce the dynamic power consumption [11]. This method is similar to the leaf-level strategy presented in the Section 2.5.1. However, as shown in Figure 2.8b, an enable signal is created by an XOR-gate. With this implementation, the clock signal is disabled while the data input remains unchanged, thus reducing the dynamic power consumption. The bit width for this technique is naturally limited to one register.

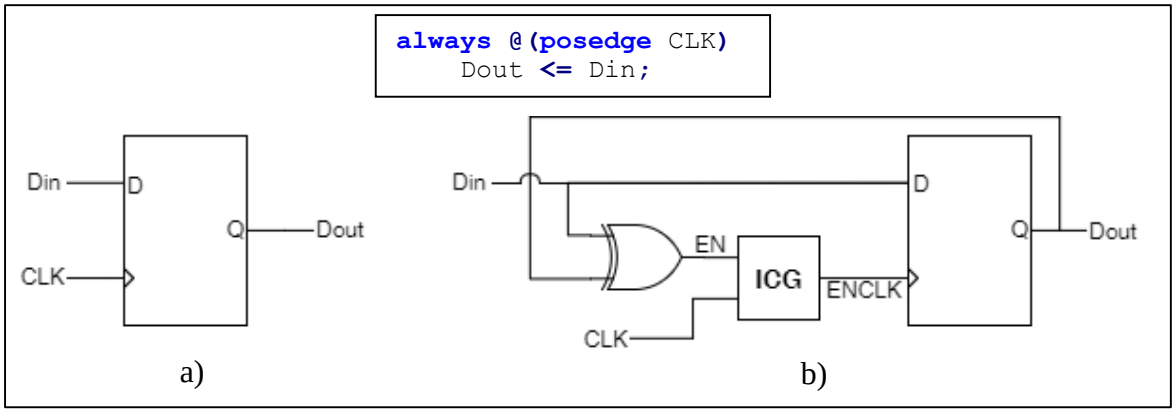

Figure 2.8: XOR self-gating.

In order to reduce the dynamic power consumption in a single register, an XOR-gate and an ICG cell has been added. This increases both the static power consumption and the area. For XOR self-gating to be beneficial, the reduction in dynamic power consumption has to be larger than the increased static consumption. It can therefore be suited for registers with high clock frequency, but low switching activity on the data input.

2.5.3 Module-Level Clock Gating

As mentioned, ICGs can be placed closer to the clock root in order to reduce the power consumption in the clock tree structure in addition to the registers. This is called module-level clock gating, as these clock gates can be used to disable the activity in large unused modules. A typical SoC for Bluetooth Smart consist of several different modules (CPU, Timers/Counters, DACs/ADCs, etc.), of which only a few are used at a given point in time. Disabling the clock signal to an unused module cancels the dynamic power consumption in the module's clock tree, registers and data path. Module-level clock gating has great power saving potential since up to 50 %, or even more, of the total dynamic power is consumed in the clock buffers [6].

The module-level clock gate can either be manually controlled by the program, or automatically controlled by the hardware. In a manual system, the clock gates are controlled by program code. Program lines are added to enable and disable the module's clock. Alternatively, an automated system can ensure that the clock is only enabled when the module is used and immediately turned off after its task is completed. This system requires some additional control logic, but simplifies the clock control for the programmer.

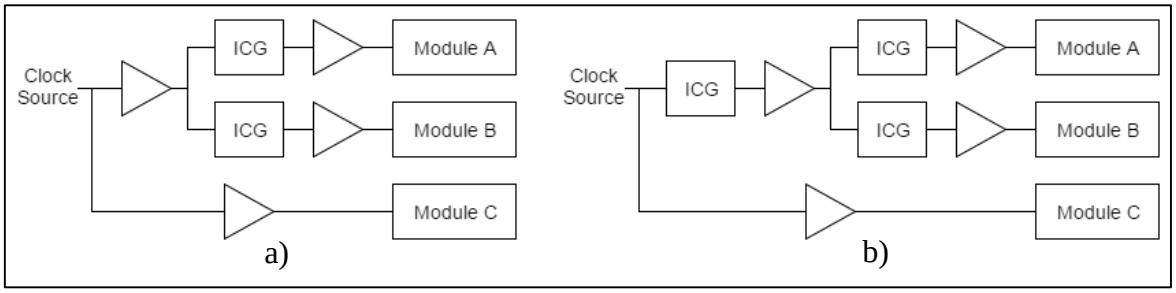

Figure 2.9: Module-level clock gating in a) one and b) two levels.

Figure 2.9 illustrates how module-level clock gating can be done. In Figure 2.9a, ICGs are inserted for modules A and B, while module C is not gated. In order to further reduce the power consumption, an additional level of clock gates has been added in Figure 2.9b. Depending on the scenario, larger parts of the clock tree to be disabled with multiple levels of clock gates. As shown in Figure 2.9b, the buffer driving the ICGs for Module A and B can be clock gated is both modules are inactive. However, adding additional levels of clock gates requires additional logic. In addition, the ICGs can cause imbalance in the clock tree.

3 Previous Work and Design Tools

This chapter presents relevant research in the topic of clock tree synthesis and clock gating. In addition, the tools used in this project are described.

3.1 Clock Tree Synthesis

Clock tree synthesis is a well-researched topic in continuous progress. The most common strategy is to minimize the clock skew. In this strategy, the clock tree is formed so that the clock edge arrives simultaneously for all sequential cells, i.e. zero skew. In order to do this, the sinks of the clock tree have to be clustered so that the load of the clock tree becomes balanced. Algorithms for clustering is addressed by [12, 13, 14, 15]. Most recent of these is the work by Shelar [15]. Shelar presents a power-aware clustering algorithm that uses a minimum-spanning tree metric to achieve low power consumption. Results on a 65 nm technology shows consistent reduction of the total clock tree capacitance by up to 21% compared to competitive approaches.

After the sinks have been clustered, the next step is to route the clock tree. Routing strategies for achieving low skew are addressed in [16, 17, 18].

As the clock complexity increases, minimizing the skew becomes difficult and expensive with respect to buffer area and power consumption. Therefore, an alternative strategy is to implement useful skew, a method of using the clock skew actively to meet timing requirements. Useful skew can be implemented to improve the performance, e.g. operating frequency [19, 20, 21]. By allowing some skew, timing violations in the critical data paths can be avoided at higher clock frequencies.

Alternatively, useful skew can be used to reduce the cost of the clock tree. Xi et al. [22] presents an algorithm that allows some negative skew (delayed clock edge arrival at the destination register), which gives a larger timing budget. Due to the large number of configurations, the algorithm uses the stochastic search method of simulated annealing to avoid being trapped in local minima. Experimental results show a power reduction of 12% to 20%, compared to previous methods, without any reduction in clock frequency.

Another approach based on useful skew is purposed by Ramachandran [23]. Instead of using a zero or bounded skew target, the arrival time requirements are derived from the sinks' timing constraints. The goal of this approach is to create a minimalistic clock tree and reduce the buffer count, thus reducing the power consumption.

3.2 Clock Gating Strategies

A recent survey by Pouiklis et al. [24] presents a spectrum of clock gating approaches. This survey presents both theoretical and practical considerations throughout the design process. Requirements, limitations and tool support have been examined for each step in the design process. In addition, an assessment of testing compatibility, such as scan clock functionality, is examined. Finally, clock gating results from other literature are presented and evaluated. The survey by Pouiklis et al. considers leaf-level clock gating only.

The literature and research on the topic of module-level clock gating is sparser than leaf-level clock gating. This lack of research can be explained by the fact that module-level clock gating is application specific. However, Bu et al. [25] present a novel module-level clock gating implementation of a CPU. First, an adaptive module-level clock-gating cell with the ability to automatically sense module activity is being created. This cell is then applied to reduce the power consumed during pipeline stalls and when modules such as FPUs and co-processors are idle. Simulations on a 65 nm technology show an average power reduction of 18% to 28%, with no impact on CPU performance and negligible area overhead [25].

In experiments by Suito et al. [26], low power techniques have been implemented on a multithreaded processor, where one technique is fine-grained module-level clock gating. In this experiment, the power consumption, chip temperature and timing overhead were examined. The total power consumption and temperature reduction of the implemented low power techniques, is 88% and 23%, respectively, while the timing overhead is up to 75 µs. However, the timing overhead caused by the module-level clock gates is only 4 µs.

3.3 Clock Tree Implementation in Synopsys® IC Compiler™

The clock tree implementation can be done in Synopsys® IC Compiler™ with the *clock_opt* command [11]. This command consists of the 12 steps listed below, where some are optional. All steps can also be done individually through separate commands.

1. Clock tree synthesis: Building a violation free clock tree, as described in Section 3.3.1.
2. Clock tree optimization: Improving the clock skew, as described in Section 3.3.3.
3. Interclock delay balancing (Optional): Balancing the skew between a group of clocks.
4. Routing: Detailed routing of the clock nets.

5. Arrival time computation: Extracting values for resistance and capacitance in order to compute accurate arrival times.
6. Adjustment of I/O timing (Optional): Adjusting the input and output delays in order to match clock arrival times.
7. Power optimization (Optional): Performing a physical optimization of the ICGs, power-aware placement and leakage optimization.
8. Congestion reduction (Optional): Reducing congestions in order to improve the routability of scan nets.
9. Scan chain optimization (Optional): Reordering of the scan chain to reduce the number of buffer crossings in the chain.
10. Placement fixing: Fixing buffer and inverter placements.
11. Placement and timing optimization: Optimizing placement of registers and combinatorial logic, yet keeping the clock tree fixed.
12. Hold time violation fixing (Optional): Fixing register hold-time violations by manipulating the data paths, inserting buffers, etc.

This project focuses on the clock tree synthesis and optimization (steps 1 and 2), described in Sections 3.3.2 and 3.3.3, respectively. In order to perform these two steps exclusively, the *clock_opt* command is given the *–only_cts* and *–no_clock_route* arguments. The most important options for the clock tree synthesis and optimization are specified in Section 3.3.1.

3.3.1 Clock Tree Options

In addition to selecting the optional steps in the clock tree synthesis, constraints and optimization targets are set prior to synthesis. These options and their default values are listed by descending priority in Table 3.1. The options can be given values both globally and per clock.

Table 3.1: Clock tree synthesis options.

Priority	Variable Name	Default value
Design Rule Constraints		
1	`max_capacitance`	0.6 pF
2a	`max_transition`	0.5 ns
2b	`leaf_max_transition`	“Unspecified”
3	`max_fanout`	2000
Clock Tree Timing Goals		
4	`target_skew`	0 ns
5	`target_early_delay`	0 ns

The design rule constraints set the maximum allowed values for the net capacitance, clock transition time and fan-out. For the transition time, the designer has the opportunity to set a separate constraint for the leaf-level nets (connected to the clock pin of the registers). If the maximum leaf transition time is not specified, the maximum transition time is used also for the leaf levels. This is the default behaviour. The skew target is used in the optimization process. When this target is met, the optimization ends. The target early delay specifies a minimum goal for the shortest clock path. This option can be used to match the delay in different clock trees.

In addition to the design rule constraints set by the user, the tool also considers constraints from the cell library and the design specification. The smallest value is chosen for max_capacitance and max_transition, while for max_fanout, the user specified value is chosen.

It is also possible to select which cells to use in the clock tree synthesis and options. Three different selections needs to be specified. The first selection is the buffers and inverters available for insertion in the synthesis stage. The second selection is the delay buffers/inverters used to balance the delay of the branches. The third selection is the resizing list used for all cells in the clock tree (buffers, inverters, ICGs, multiplexers, etc.) resizing list. With these selections, it is possible to exclude cells with adverse effects, such as high current peaks and low drive strength.

3.3.2 Clock Tree Synthesis

The first step of the clock tree synthesis is an analysis of the available buffers and inverters. In this analysis, the positive and negative edge delays of the buffers are calculated. If the delay difference is too large, the buffer is excluded for use in the synthesis and optimization [27].

If the XOR self-gating methodology is enabled, clock gates are inserted in the leaf level. When these are inserted, the tool takes timing, power and clock-tree QoR into account. In order to determine whether the inserted ICG actually saves power, accurate switching activity annotations are required. XOR self-gating is only done for registers where the reduced dynamic power is greater than the increase in static power.

The next step is to prepare the clock tree for the synthesis. The existing clock cells are upsized to cells with higher drive strength. This is done to solve constraint violations

without inserting buffers, resulting in reduced tree depth and latency. In addition, the clock cells, including ICGs, are relocated if this can increase the QoR [11]. However, cells is not moved so that the functionality of the clock tree changes.

Finally, the clock tree is built to meet the constraints, while keeping the tree balanced and the clock skew minimized. As presented in Section 3.3.1, there are three clock tree constraints; maximum capacitance, maximum transition time and maximum fan-out. These constraints can be related, as a high fan-out net often has a high capacitance, and thereby reacts slowly. Therefore, the procedures to solve the violations are similar. Maximum capacitance violations and maximum fan-out violations are fixed by dividing the fan-out /load into smaller groups and drive these with individual buffers. When the fan-out is being dividing, it is important to maintain balance in the tree in order to minimize the clock skew.

There are two ways to fix transition time violations. Similar to the case of fan-out and capacitance violations, the fan-out driven net can be divided into smaller groups driven by individual buffers. This reduces the individual net loads, which gives faster reaction time and improved transition time. Alternatively, a larger buffer with higher drive strength can be used to reduce the transition time.

3.3.3 Clock Tree Optimization

The goal of the clock tree optimization is to minimize the clock skew to meet the target skew [11]. In order to do this, the tool can do the following, where all action are optional:

- Resize buffers/inverters
- Move buffers/inverters
- Resize clock tree cells (ICGs, multiplexers, etc.)
- Move clock tree cells
- Insert delay buffers

The optimization process stops when the skew target is reached. However, if the skew target is set very tight, the optimization might be stopped before the target is reached due to high buffer costs. This is controlled by the tool’s internal cost function.

When the skew optimization process is completed, additional delay is inserted at the clock root to meet the target early delay. After this step, the shortest clock path should be greater

than the specified target early delay. Since the default of this target is 0 ns, no root delay is inserted unless the target is increased by the user.

3.3.4 Logic Level Balancing

The logic level balancing feature builds the clock tree structure so that all sinks are placed at the same level of the clock tree [11]. This is illustrated in Figure 3.1. The clock tree structure prior to the synthesis is simple, with 0 to 2 cells between the clock source and registers. In a typical synthesis, buffers are inserted, increasing the amount of cells between the clock source and the registers to 2 to 6 cells. However, with logic level balancing, additional cells are inserted in the shorter branches so that there is an equal amount of cells between the clock source and every register.

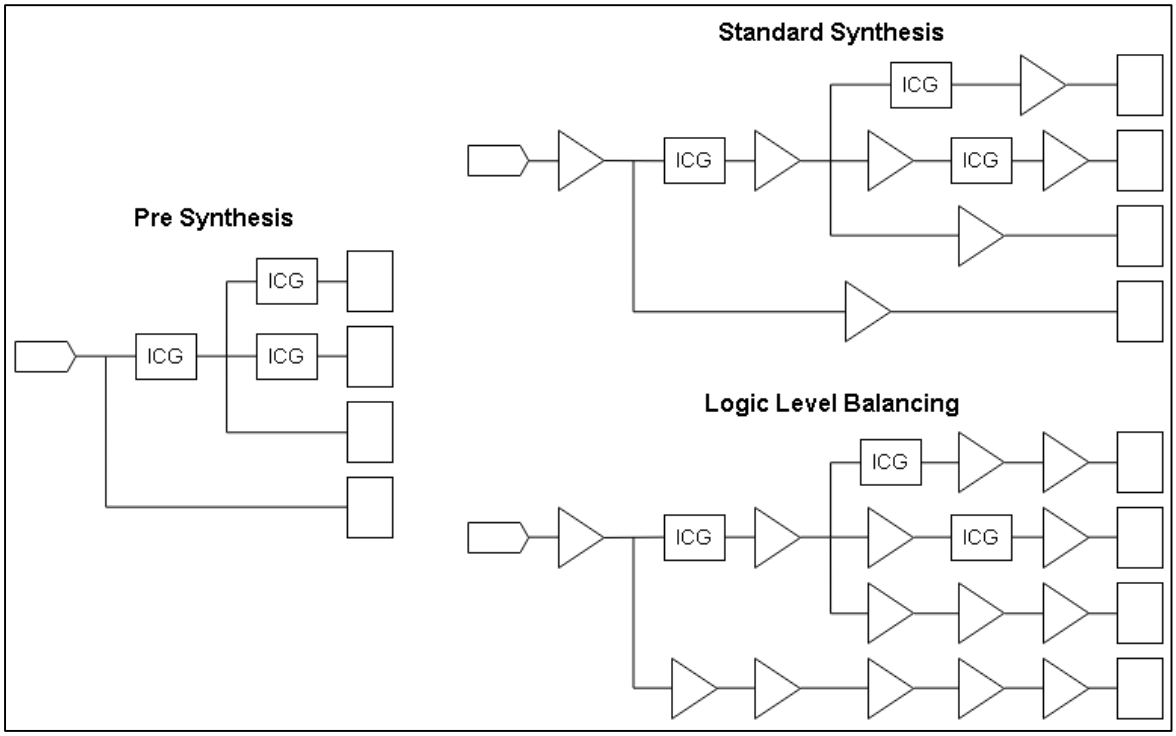

Figure 3.1: Logic Level Balancing.

The logic level balancing feature provide a balanced clock tree with low path variation and is said to give the best clock skew [28]. The structure is also robust against process variations as the depth is identical for all sinks. However, as additional buffers are inserted, logic level balancing is typically more expensive with respect to buffer area and power consumption.

3.3.5 Guidelines

Synopsys® provides several application notes with guidelines and recommendations for the clock tree synthesis in Synopsys® IC Compiler™ [27, 28, 29]. Some of the most important recommendations are listed below:

- Remove any existing buffers or inverters from the clock tree prior to the synthesis. These will be treated as balancing points and can affect the clock tree QoR [28].
- Prior to the synthesis, run the *check_clock_tree* command and solve any reported issues [29].
- Use buffers/inverters with minimal difference in their rising and falling edge delays [27, 29]. Unbalanced buffers might cause bad skew.
- Use default synthesis constraints as tightening the constraints will degrade the synthesis results considerably [27, 29]. Tight constraints can cause an excessive use of buffers and increase the clock tree power consumption.
- Use a naming convention for buffers/inverters inserted during the different steps of the synthesis [27]. This makes it easier to debug any synthesis issue.

3.4 Power Estimation in Synopsys® IC Compiler™

In this project, power estimation is done in Synopsys® IC Compiler™. This tool is calculating dynamic and static power consumption based on the static (i.e. time independent and averaged) switching activity parameters. These consist of the static probability and the toggle rate, and are denoted for each net in the design. The static probability is a floating point number between 0 and 1, and represents the percentage of the time the net is at logic 1. The toggle rate describes the number of "0-to-1" and "1-to-0" transitions the net is doing per nanosecond. E.g. a 1 MHz clock signal will have a static probability of 0.5 and toggle rate of 0.002 ns^{-1}.

With the denoted activity parameters, power is calculated using data from the cell library. The internal cell power and static power are calculated using the switching activity of the input nets and power data for the specific cell, obtained from the library. The switching power for a net, $P_{SW,net}$, is derived from Equation (1) in Section 2.3.1. The activity factor, αf_{CLK}, is equal to the net's toggle rate, TR_{net}, and C_{net} is the total net capacitance. This gives the resulting formulas presented in Equation (6).

$$\alpha f_{CLK} = TR_{net}[\mathrm{ns}^{-1}]$$
$$P_{SW,net} = \frac{TR_{net}}{2} C_{net} V_{DD}^2 \quad (6)$$

The switching activity of a net can be denoted in three different ways. Primarily, the activity can be set to a specified value, by using tool commands or loading activity results from RTL-simulations, or similar. If the activity of a net is not set, the tool will try to calculate the activity using the functional description from the driving cells. If the net does not have any drivers, i.e. an input net, the net will be given a default activity specified by the variables *power_default_toggle_rate* and *power_default_static_probability*.

In order to analyse the design in a specific power mode, activity data for this mode is typically loaded from simulation results. Alternatively, the designer sets the switching activity of the clock sources and input pins. Activity can also be set to selected nets in order to put the design in a specified mode, before the tool propagates the activity to the rest of the design.

3.5 Clock Tree Analysis Tool

In order to evaluate the clock tree performance, an analysis tool created in a previous project has been used. The development of the tool is presented in detail in the project report [1]. The tool extracts data from a set of synthesis and power reports, and stores the data in a .csv file. The .csv data file is dynamically linked to a Microsoft Excel sheet, which presents the data in an informative manner. New synthesis data can easily be added to the tool by appending new lines to the .csv file. To display the new data, the sheet has to be refreshed.

The data presented by the tool are separated into the following categories:

- Dashboard: Links to every parameter for easy navigation.
- Summary: Table of the most important clock tree parameters. Similar to the Synopsys® IC Compiler™ clock tree summary, but extended with efficiency ratios and power figures.
- Additional filters: Filters the results by data set, clock name, design name, scenario and clock tree settings.
- Design rule check: The number of synthesis constraint violations.

- Clock tree cells: The number of cells of the different of clock tree cell types and the efficiency ratio “sinks per buffer”.
- Level information: The post synthesis level information of the clock tree, including how sinks, buffers and ICGs are distributed in the clock tree levels.
- Buffer area: Clock tree buffer area statistics and the ratio of “sinks per buffer area”.
- Timing: Maximum global clock skew and longest clock path.
- Power activity 0-100%: Clock tree power figures from 0% to 100% activity as presented in Section 3.5.1.
- Power Scenario 1 and 2: Clock tree power figures for two optional power scenarios. In this project, one scenario has been used.

In this project, the most important feature of the tool is the ability to extract data from reports and make them available in Excel Pivot Tables. This simplifies the creation of data tables and plots. This also makes it easier to handle the large data collection.

3.5.1 Activity-Based Power Estimation

The activity-based power estimation scheme was created to analyse the power consumption in clock-gated trees without using results from simulations. Instead of analysing specific scenarios, in this scheme, the clock tree power consumption is estimated based on five steps of activity; 0%, 25%, 50%, 75% and 100% activity. These activity parameters determine the static probability of the clock gates’ enable signal. In other words, 0 % activity means that all ICGs are disabled, and that the clock tree operates in its lowest power mode. However, at 100 % activity, all ICGs are enabled and the worst-case power consumption is estimated.

The activity parameter is set directly on the enable pin of all ICGs in the design. The idea behind this method is that the functional description of the ICG will lead to correct propagation of switching activity during the power analysis. Alternatively, the activity can be set directly on the output of the ICGs. This method can give better accuracy, but can also become very complicated for designs with multiple layers of ICGs and clock dividers. Hence, setting the enable pin is preferred.

4 Methodology

In this project, only the clock-tree synthesis stage of the design process, highlighted in Figure 4.1, is considered. This means that no high level or RTL work have been done in this project. Instead, the two test design, presented in Section 4.1, have been provided in a post initial-placement state. From this state, the clock trees have been synthesized using Synopsys® IC Compiler™ tool, as presented in Section 3.3. The scan functionality has not been considered in this project. Therefore, the designs have not been synthesized in the scan mode.

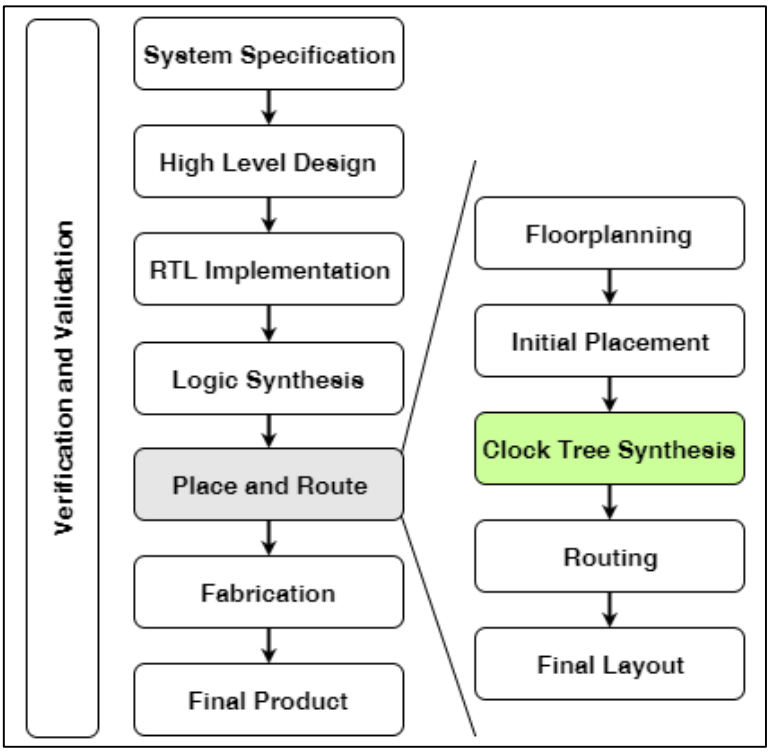

Figure 4.1: SoC design flow.

In this project, the clock skew, buffer area and power consumption have been considered the most important parameters of the clock tree. The skew and the buffer area are obtained from tool generated synthesis reports, while the power consumption is estimated according to the description in Section 4.2.

The test and experiments in this project is divided into an exploration of the clock tree synthesis functionality of Synopsys® IC Compiler™, presented in Chapter 5, and a multi-level module-level clock gating experiment, presented in Chapter 6. In both chapters, the most important results are presented and discussed.

4.1 Test Designs

In this project, two designs in the same sub-100 nm technology have been examined. In order to test how the design size and complexity affect the clock tree synthesis, a simple artificial test design has been generated as presented in Section 4.1.1. A benefit of the artificial design is reduced synthesis time. Therefore, the artificial design has been used to test the functionality of the synthesis tool. Experiences on the artificial design has then been applied on a real Bluetooth Smart design, presented in Section 4.1.2. In addition, multilevel module-level clock gating has been tested on the real design.

4.1.1 Generated Test Design

The artificial test design is generated in Synopsys® IC Compiler™ by a simple script. This script was created by supervisor Are Aarseth in order to perform experiments on the clock tree synthesis in Synopsys® IC Complier™. The idea behind Aarseth's script is to create a very simple data path, while the complexity of the clock network is defined by the user input. The scrip has been modified to suit this project.

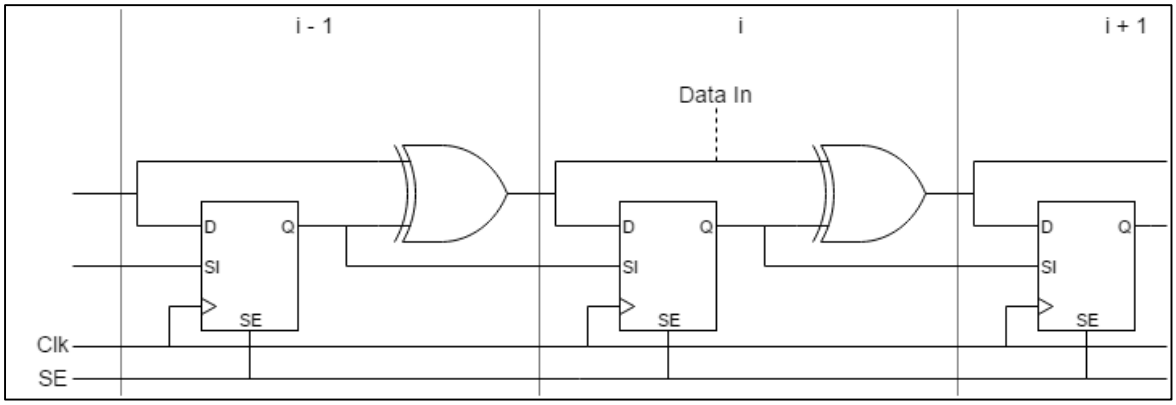

Figure 4.2: Test design data path.

The data path of Aarseth's script consists of registers and XOR-gates only, connected in a long chain as illustrated in Figure 4.2. The total number of registers in the data path, N, is specified by the user input. In the first link ($i = 1$), both the register input and the scan input are connected to an input pin. In order to avoid long paths, for every 30th link, the data input is connected to the input port instead of the XOR output of the previous link. In Figure 4.2, this is indicated with a dotted line. At the end of the chain, the output of the last register is connected to an output pin. As the schematic shows, the scan input, *SI*, is

connected directly to the output of the previous register. Scan mode is activated for all registers with the scan enable signal, *SE*.

As shown in Figure 4.3, only ICGs are used to distribute the clock signal. This is done to keep the design as simple as possible, whilst still reflecting a realistic clock tree structure. Naturally, buffers and/or inverters are inserted during clock tree synthesis and optimization. The clock network is created based on a set of input parameters provided by the user. The user input specifies the number of ICG levels. In addition, for each ICG level the percentage of registers covered by the level and maximum register bank size are specified. In Figure 4.3, this percentage is indicated on the gated branch of the level. Note that the percentage is independent of previous levels. The maximum register bank size specifies how many a registers a single ICG can cover. However, to decide the register bank size of a specific ICG, a random number generator is used. New ICGs are created until the specified percentage of gated registers is reached.

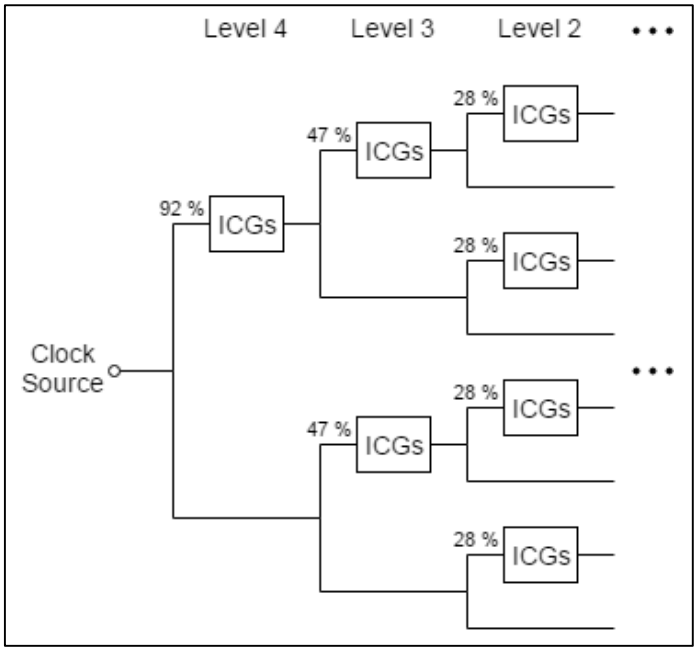

Figure 4.3: Test design clock tree structure.

Some changes to the clock generation has been implemented for this project. Two additional parameters have been added for each level of clock gates; one for the minimum register bank size and one for enabling creation of the ICG level. With the minimum register bank size parameter, the register bank size is limited within a minimum and

maximum value. For each new ICG, a random value within these limits is generated. The minimum limit is implemented to avoid branches close to the root with very few registers.

The clock tree is created from the root (clock source) down to the leaves (registers). For each ICG level, a branch covering a random amount of registers within the register bank limits of the level, is created. The ICG is then inserted at the root of this branch. This procedure continues until the leaf level is reached. When a leaf-level ICG is being created, the output of the ICG is connected directly to the clock pins of the registers.

Due to the use of a random number generator, different clock tree structures are being created each time the generation script is run, resulting in random variations in the results. This is an undesirable side effect. In order to explore the effect of the ICG levels in a clock structure, the basic clock structure should be kept constant, so that the only variation is in the ICG insertion. In other words, the script should be able to create two clock trees with identical structures, but with ICGs placed differently in this structure.

The previously mentioned level-enable parameter was added to avoid these random effects. The enable parameter is a Boolean input that enables insertion of ICGs for the current level. When a new branch is created, the ICG is only inserted if its level is enabled. If not, the clock tree generator moves to the next level, to create new branches and potentially insert ICGs. This enables the possibility of creating identical clock tree structures as long as the clock gate probability and register bank limits are kept constant. The level-enable parameters can then be changed to create clock trees with identical structure, but with different ICG levels implemented. In addition, this requires that the random number seed is reset before the creation of each clock tree.

The resulting four input parameters required per level are listed in Table 4.1.

Table 4.1: Clock inputs per level.

Input parameter	Type	Description
Gated Register Percentage	Float	Percentage of registers covered by the ICG level.
Minimum Register Bank Size	Integer	Minimum registers in the branch of the level.
Maximum Register Bank Size	Integer	Maximum registers in the branch of the level.
Enable ICG Insertion	Boolean	Enable ICG insertion in the level.

In this project, up to five levels of clock gating are implemented. The selected input, presented in Table 4.2, is based on an analysis of the real Bluetooth Smart design, presented in Section 4.1.2. The ICG Structure analysis script, described in Section 4.3, was used to analyse the clock gate structure of the design. This script provides information on the different ICG levels of the design. Using this information, a good, but simplified, representation of the structure of the real clock tree can be created. Figure 4.3 shows the first levels of the resulting clock tree structure.

Table 4.2: Clock gating statistics.

ICG level	%[1]	Min-Max[2]
Level 0 (leaf)	71 %	1 – 47
Level 1	79 %	2 – 480
Level 2	28 %	40 – 2,500
Level 3	47 %	150 – 11,500
Level 4	92 %	350 – 20,000

[1] Percentage of gated registers.
[2] Register bank size range.

For simplicity, the clock frequency in the generated design was set to 50 MHz. In Synopsys® IC Compiler™, this gives a toggle rate of 0.1. This value makes it easy to track propagation through the clock tree, as the output toggle rate on the ICG, TR_{OUT}, is given by Equation (7), where a is the activity factor and l is the number of ICGs between the current ICG and the clock root.

$$TR_{OUT} = 0.1a^{l+1} \tag{7}$$

4.1.2 Bluetooth Smart Design

In this project, a Bluetooth Smart design has been used as a test case. This design is a SoC solution with an ARM® Cortex®-M series CPU, on-board radio and several other modules and peripherals. In this project, only the main clock, *CkIn*, has been considered. This clock covers a majority of the chip, and is most important with respect to buffer area and power consumption due to its large size and high frequency. The main motivation for focusing on the clock has been done to reduce the synthesis time. The principal structure of the main clock is shown in Figure 4.4.

The frequency at the source of this domain is in the sub-100-MHz range. However, as the red markers in Figure 4.4 shows, the clock has several subtrees of different frequencies. In

total, this clock is synchronizing about 40 000 registers. The clock tree combines two strategies of clock gating. At the leaf level, the clock signal is gated by tool-inserted ICGs. In addition, ICGs are placed closer to the root in a single level of module clock gates.

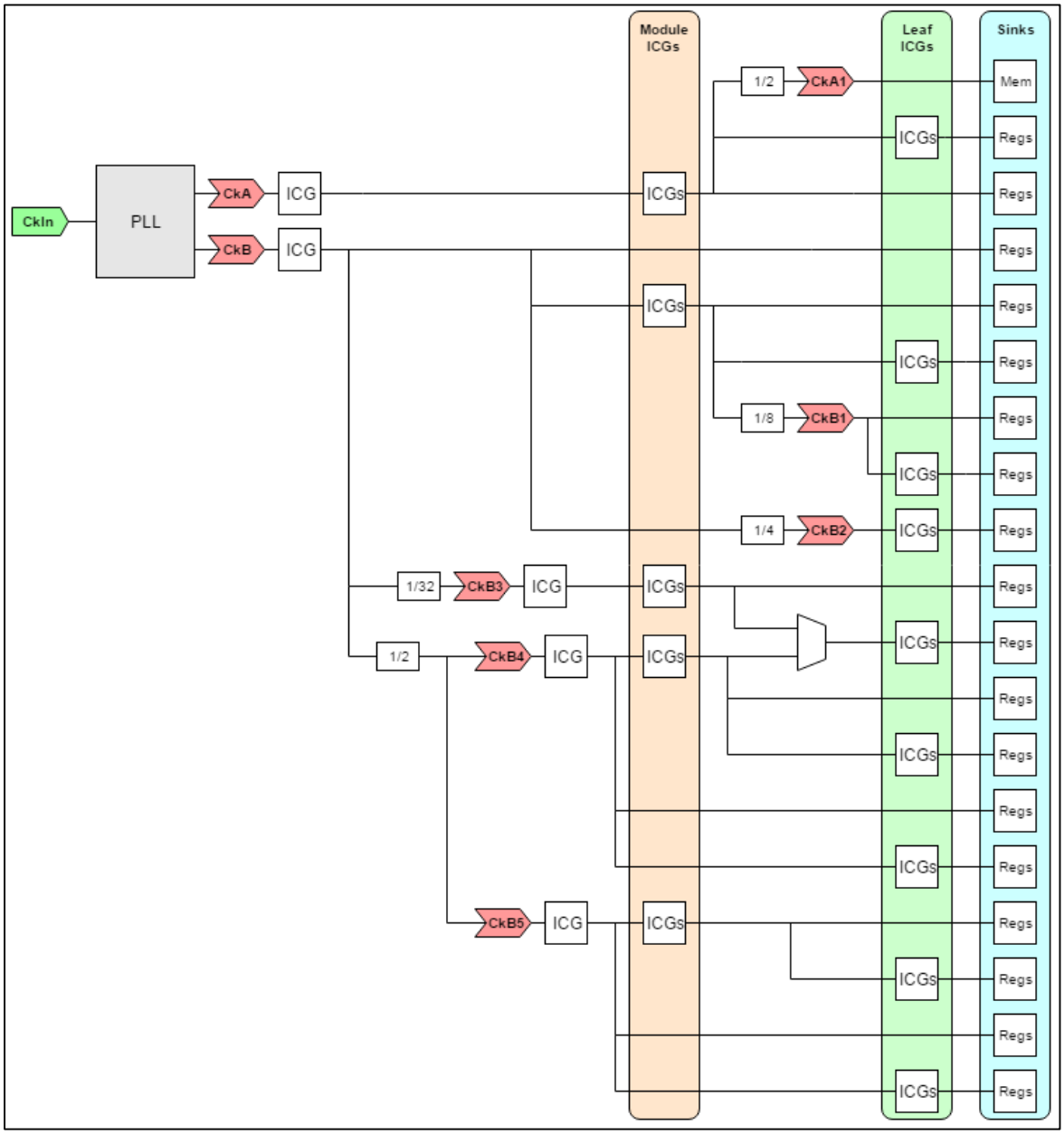

Figure 4.4: CkIn tree structure.

The module-level clock gates are automatically controlled by the hardware. Whenever a module is needed, a request is sent to the clock managing unit. The clock managing unit then enables the clock. When the module has finished its task, a notification is sent to the clock managing unit, which then disables the clock signal. This means that the clock is only enabled for active modules. ICGs are also used to turn off the clock signal at the root

of the largest subtrees (*CkA*, *CkB*, *CkB3*, *CkB4*, *CkB5*) in sleep modes. In addition, ICGs are used as pulse blockers to shape the clock signal of some of the generated clocks (*CkB1* and *CkB2*).

Even with several layers of clock gating, the high frequency clock has significant power consumption at its minimum-power mode. Therefore, in the deepest sleep modes, it is completely turned off. Instead, an energy efficient low frequency clock is used. This low frequency domain has not been considered in this project.

4.2 Power Estimation

Power estimation has been done in Synopsys® IC Complier, as presented in Section 3.4. The activity based clock tree power estimation scheme described in Section 3.5.1 has been used. In addition, a timer scenario was created for the Bluetooth Smart design, as described in section 6.1. This scenario was implemented in order to examine the effects of adding an additional level of module-level clock gating. When implementing these power estimation scenarios, it was discovered that the toggle rate of the ICGs' enable signal influenced the accuracy of the estimation. This issue and a solution is described in Section 4.2.1.

4.2.1 ICG Enable Pin Toggle Rate

When setting the ICGs' enable signal, both the static probability and the toggle rate is set with the *set_switching_activity* function, as described in Section 3.4. The static probability is the enable duty of the ICG. When this duty is 0 or 1, constantly enabled/disabled, the enable signal is not toggling. Hence, the toggle rate is set to 0. However, for any other enable duty, the toggle rate has to be set to a value greater than 0.

The enable pin static probability and toggle rate are used by the tool to propagate switching activity through the ICG. The functional behaviour in Equation (8), where SP is the static probability and TR is the toggle rate, is expected. In this equation, $ENCLK$ is the output, CLK the input and EN the enable signal, similar to the description in Section 2.4.3. This means that the output is only dependent on the static probability on the enable pin, SP_{EN}, and independent of the enable-pin toggle rate.

$$\begin{aligned} SP_{ENCLK} &= SP_{EN}SP_{CLK} \\ TR_{ENCLK} &= SP_{EN}TR_{CLK} \end{aligned} \tag{8}$$

Due to this independence, the toggle rate was initially set to a low value to reduce its impact on the power consumption. However, it was discovered that this resulted very inaccurate switching activity propagation through the ICGs. A simple test was created to examine the activity propagation with different enable-signal toggle rates. The average absolute deviation from the expected propagation in Equation (8) was used as a measurement of the accuracy. The results of this experiment is listed in Table 4.3.

Table 4.3: Enable pin toggle rate results.

Enable Toggle Rate	Deviation
0.1	194.0 %
0.01	2.5 %
0.001	3.6 %
0.0001	10.5 %
0.00001	37.1 %
0.000001	63.4 %

As Table 4.3 shows, the accuracy is dependent on the toggle rate of the enable pin. The enable-pin toggle rate of 0.01 was found to give the best accuracy, and is therefore used in power estimations in this project.

4.3 ICG Structure Analysis

A clock gate analysis script was created for Synopsys® IC Compiler™ to be able to report clock gate switching activity and characterize the ICG structure. This script is based on a Synopsys® PrimeTime PX script [30]. The script has been reworked and many new features are added in order to extract the desired clock gate statistics. This script reports the following for each ICG in the design:

ICG name: This is simply the name of the ICG cell.

ICG level: The level of the ICG. Level 0 is the leaf level, and the number increases for each level. This parameter is found by analysing the ICGs of the subtree of the current ICG.

Number of cells/buffers/registers/ICG: The total number of cells, and the number of each cell type in the ICG's subtree.

Pin switching activity: The static probability and toggle rate of all inputs and outputs of the ICG.

Error message: An error is displayed if the observed enable probability has more than ±10% deviation from static probability annotated on the enable pin.

The script also reports a summary for the total design and for each ICG level. This includes the amount of gated registers, register bank statistics (minimum, average and maximum) and average observed enable probability. An example report is provided in Appendix A.

In this project, this script has been used to analyse the clock tree structure of the Bluetooth Smart design, presented in Section 4.1.2. This data has been used to create a similar clock structure in the generated test design, as described in Section 4.1.1. This script has also been used in examination of the ICG enable pin toggle rate, presented in Section 4.2.1. Finally, the script has been used to verify that the correct behaviour is set in the power analysis.

5 Clock Tree Synthesis Exploration

In order to explore the clock tree synthesis functionality of the Synopsys® IC Compiler™, the synthesis has been performed on multiple different designs combined with different settings. The goal of this exploration is to:

- Examine how the different design parameters affect the synthesis. As described in Section 5.1, both the clock domain size and the complexity of the tree structure have been examined.
- See how constraints and targets influence the synthesis results. The skew target and the max transition time constraints have been explored as described in Section 5.2 and 5.3, respectively.
- Understand how the selection of buffers and inverters affect the synthesis, as presented in Section 5.4.
- Examine if the logic-level balancing feature can be used to achieve better synthesis results, as described in Section 5.5.
- Illustrate trade-offs and find optimal settings with respect to power consumption, buffer area and clock skew.

In order to explore the synthesis tool efficiently, a simple test design has been created as described in Section 4.1.1. This reduces the synthesis time considerably and makes it possible to test settings efficiently. Based on experiences from the generated design, similar tests has been applied to an actual Bluetooth Smart design, that is presented in Section 4.1.2.

The most important results from this exploration is presented in the following sections. The complete results from the generated design and the Bluetooth Smart design are listed in Appendix B and Appendix C, respectively.

5.1 Clock Size and Complexity

The clock complexity and size both affect the synthesis results. In a previous project, derivations of a simplified clock tree model showed a linear relationship between then number of sinks and both the buffer area and power consumption [1]. In addition, it is expected that a complex ICG structure might cause unbalance to the clock tree, resulting in higher costs than a simple clock tree.

The generated test design presented in Section 4.1.1 enables testing of how the design parameters affects the clock tree synthesis. The generator script makes it possible to create clock trees of different sizes (number of sinks) and different structures (level of ICGs). In this project, the sizes of 1, 8 and 32 thousand flip-flops have been tested with zero to five levels of clock gating. The sizes have been selected to represent typical clock tree sizes, while the clock gate structure is based on statistics described in Section 4.1.1. With zero levels of ICG, the clock tree is not gated, and for each new level, an additional layer of ICGs is inserted closer to the clock root.

5.1.1 Effects of the Clock Size

Table 5.1 shows results obtained for the tree different design sizes after clock tree synthesis with the default constraints and targets. For all the three sizes, five levels of ICGs are implemented with statistics described in Section 4.1.1.

Table 5.1: Design size synthesis results.

Design Size[1]	Clock Skew [ns]	Buffer Area [µm²]	Dynamic Power[1] [mW]	Static Power [nW]
1,000	0.108 -	120 -	0.098 -	0.128 -
8,000	0.135 (+24%)	813 (+577%)	0.774 (+687%)	0.922 (+620%)
32,000	0.150 (+38%)	3114 (+2493%)	3.12 (+3071%)	3.48 (+2619%)

[1]Number of flip-flops.
[2]At 100% activity.

With the default settings in Synopsys® IC Compiler™, the target skew is set to zero. However, as the results show, the optimization process ended before this target was reached. The resulting skew is lowest for the smallest design size. This is most likely because the optimization process has been easier, due to the small size.

As expected, the buffer area and power consumption increases with the design size. Figure 5.1 shows how the buffer area, and static and dynamic power consumption increase with the design size. The different coloured lines represent different levels of ICGs. As the plots show, the assumption of linear relation between these parameters and the number of sinks, is strengthened. However, with more levels of ICGs lines become steeper. The effects of the ICG levels are examined in the following section.

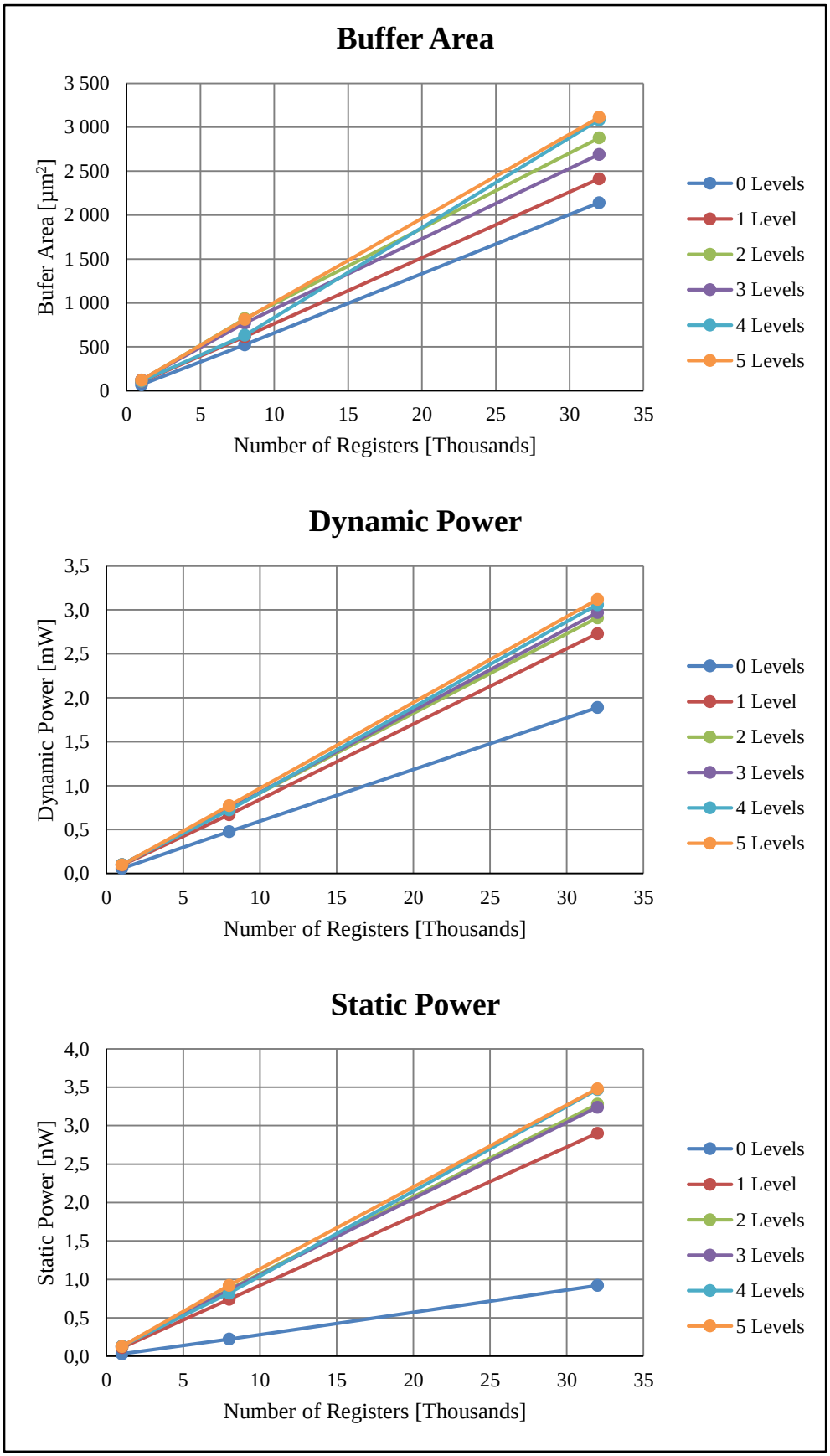

Figure 5.1: Design size and clock complexity results.

5.1.2 Clock Tree Complexity Effects

The effect of adding ICG levels for the 32 000 registers design is summarized in Table 5.2. As this table shows, the achieved clock skew is lowest with no clock gates. This is because there are no clock gates causing imbalance in the clock tree, which makes optimization easier. With the exception of the design with three levels of ICGs, the achieved clock skew is observed to increases with increasing clock tree complexity. For the three level design, the optimization process seems to have ended before the others. This is noticeable as the clock skew is highest, while the buffer area is significantly lower than for some of the other levels. The cause of this can be explained by random variations in the synthesis, as explained in Section 5.6.

Table 5.2: ICG levels synthesis results.

ICG Levels	Clock Skew [ns]	Buffer Area [µm^2]	Dynamic Power [mW] A0[1]	Dynamic Power [mW] A100[1]	Static Power [nW]
None	0.093 -	2,141 -	1.89 -	1.89 -	0.92 -
1	0.137 (+47%)	2,412 (+13%)	1.25 (-34%)	2.73 (+44%)	2.90 (+215%)
2	0.135 (+45%)	2,878 (+34%)	0.364 (-81%)	2.91 (+54%)	3.28 (+256%)
3	0.151 (+63%)	2,691 (+26%)	0.281 (-85%)	2.97 (+57%)	3.24 (+252%)
4	0.144 (+55%)	3,084 (+44%)	0.195 (-90%)	3.06 (+62%)	3.47 (+277%)
5	0.150 (+61%)	3,114 (+45%)	0.024 (-99%)	3.12 (+65%)	3.48 (+278%)

[1]At 0% and 100% activity.

The buffer area is also increasing with every additional clock level, again disregarding results obtained with three ICG levels. For each new level, ICGs are inserted, increasing the complexity of the clock tree structure. These ICGs may have enough drive strength to drive the output load and potentially avoid buffer insertions and reduce the buffer area. However, the delay of the ICGs cause imbalance in the clock tree and additional buffers are inserted to balance the different branches. The final result is therefore an increase in buffer area. Due to the increase in buffer area and insertion of ICGs, the worst case dynamic power consumption (A100) is significantly increased. The static power is also increases, as the additional buffers and ICGs increases the total leakage current. The benefit of adding a level of ICGs closer to the clock gate is reduced dynamic power consumption when clock gates are disabled (A0). With five levels of clock gating, the power consumption is reduced by 99% compared to the ungated design.

Results from the activity-based power consumption estimation, presented in Section 3.5.1, is illustrated in Figure 5.2. Without clock gating, the dynamic power consumption is

independent of the activity level. For each new level, ICGs are inserted closer to the clock root, enabling clock gating of a larger part of the clock tree. This results in reduced power consumption at low activity for each additional ICG level. However, when all of the clock tree is enabled, the additional buffers and ICGs cause higher power consumption.

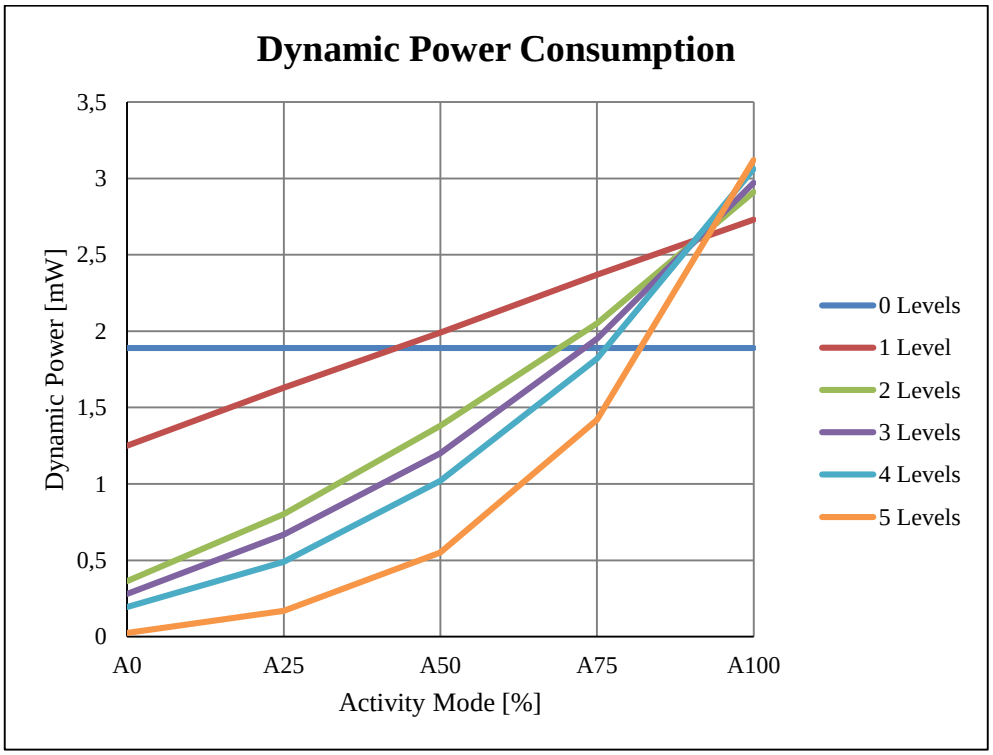

Figure 5.2: Dynamic power consumption for activity 0-100%.

For further analysis, a clock tree with 42 000 registers and 5 levels of clock gates are used. This has been selected to match the main clock domain of the Bluetooth Smart design.

5.2 Target Skew

As mentioned in Section 3.3, the target clock skew parameter determines the goal of the clock tree optimization. As the target skew is tightened, the number of branches needing balancing increases. This makes the optimization process more extensive, and expensive with respect to both buffer area usage and power consumption.

If the skew optimization becomes too expensive, the optimization process will stop without reaching the target skew. This is controlled by the tool's internal cost function, which is unknown to the user. Therefore, it has been important to examine how the tool behaves for unreachable targets.

There are several benefits of having low skew. High skew decreases the reliability of the design and can cause setup and hold time violations. Especially important is the hold time constraints in the scan chain, due to the short paths. Hold time violations are often circumvented by inserting delay buffers in the data paths. This solution comes at a cost of increased buffer area and power consumption in the data paths.

In order to examine the trade-offs between high and low skew, the tool has been examined with a range of target skew values. With the most relaxed skew targets, no or minimal optimization is needed. In the other end of the range is the zero target skew, which is the default values and requires maximum optimization. For the Bluetooth Smart design, the hold time violation cost has also been examined.

5.2.1 Generated Test Design Results

A summary of the synthesis results of the clock tree with 42 000 sinks and 5 levels of ICGs, is presented in Table 5.3. In the presented results, the target skew is varied, while the constraints are kept to their default values. The lowest skew of 0.152 ns is achieved with the zero target skew. However, as expected, this is the most expensive target with respect to buffer area and both dynamic and static power consumption. In addition, the zero-skew target was not met. Most likely, the optimization process terminated before the target was reached due to the high buffer costs.

Table 5.3: Test design target skew parameter results.

Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm^2]	Dynamic Power[1] [mW]	Static Power [nW]
0.00	0.152	3,756 -	4.09 -	4.42 -
0.25	0.282	2,210 (-41%)	3.81 (-7%)	3.78 (-14%)
0.50	0.511	1,588 (-58%)	3.68 (-10%)	3.48 (-21%)
0.75	0.745	1,526 (-59%)	3.67 (-10%)	3.44 (-22%)
1.00	1.100	1,265 (-66%)	3.65 (-11%)	3.45 (-22%)
1.50	1.386	1,229 (-67%)	3.64 (-11%)	3.44 (-22%)

[1]At 100% activity.

As the skew target is increased, the buffer area and power cost are reduced. Already at a target skew of 0.5 ns, the cost is significantly reduced. Compared to the zero target skew, the buffer area is reduced by 58%, while the dynamic and static power consumption is reduced by 10% and 21%, respectively. Further relaxing the target skew gives additional savings, especially with respect to buffer area.

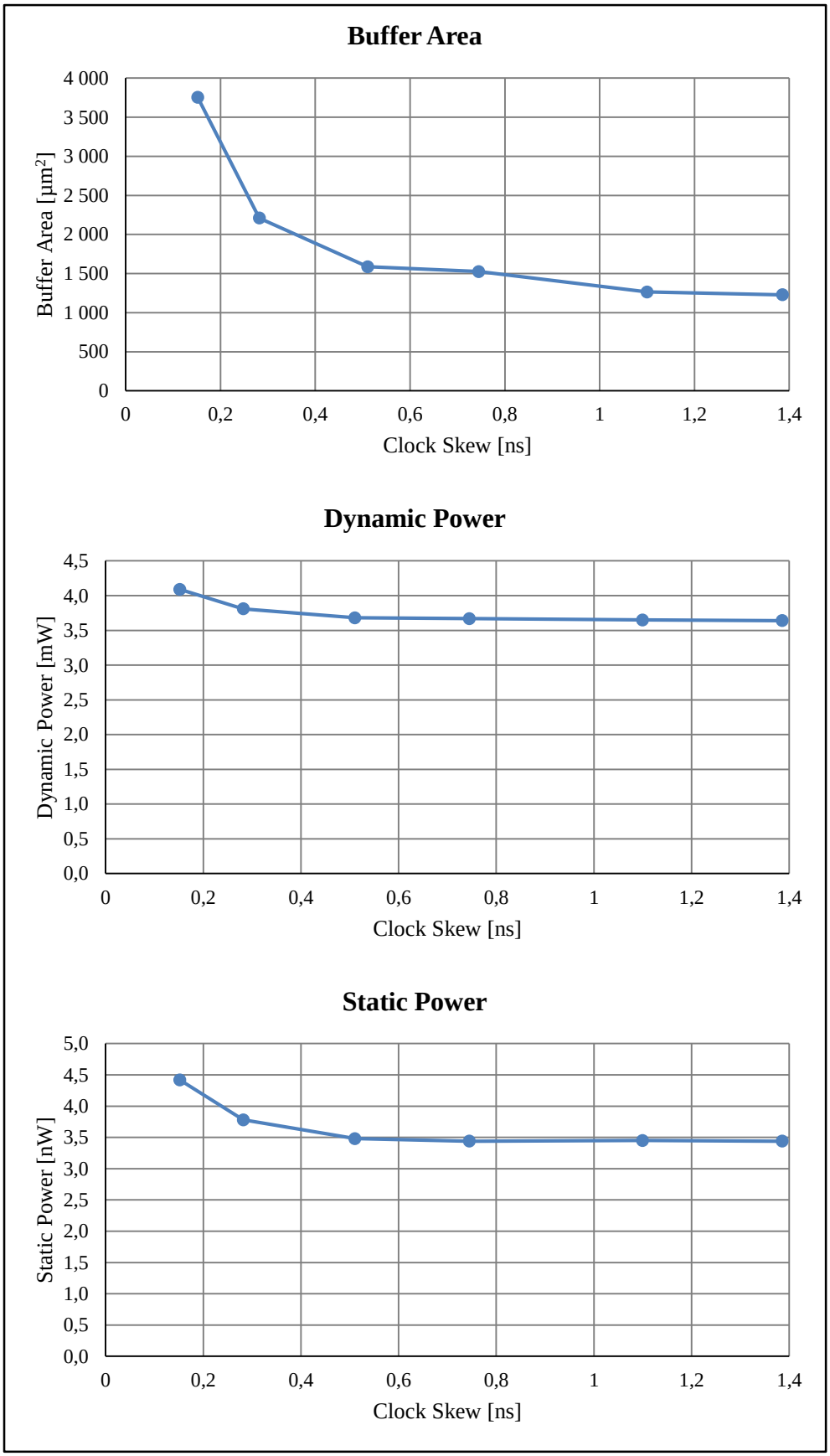

Figure 5.3: Costs vs. achieved skew.

Figure 5.3 illustrates the relationship between the achieved skew and the buffer area and power consumption costs on the vertical axes. As these plots show, at an achieved clock skew of about 0.5 ns, all cost has been reduced significantly. However, after this point the curve flattens and increased skew does not give cost reductions of similar significance.

5.2.2 Bluetooth Smart Design Results

For the Bluetooth Smart design, similar experiments have been done in order to examine how the tool reacts to different skew targets. Also here, the tested target skews range from relaxed targets with none or limited optimization, to the default zero skew target. As results for the generated test design show, at some point reducing the skew becomes significantly more expensive. It has been considered important to examine whether this is also the case for the real Bluetooth Smart design. The results of these experiments are summarized in Table 5.4.

Table 5.4: Bluetooth Smart design target skew parameter results.

Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm^2]	Dynamic Power[1] [mW]	Static Power [nW]
0.00a	0.685	6,216 -	6.00 -	129.2 -
0.00b	1.026	5,039 (-19%)	5.81 (-3.3%)	128.6 (-0.5%)
0.25	0.613	5,816 (-6.4%)	5.92 (-1.4%)	129.0 (-0.1%)
0.50	0.683	5,751 (-7.5%)	5.91 (-1.5%)	129.0 (-0.2%)
0.75	0.841	5,701 (-8.3%)	5.92 (-1.3%)	128.9 (-0.2%)
1.00	1.027	4,932 (-21%)	5.81 (-3.3%)	128.5 (-0.5%)
1.50	1.554	4,703 (-24%)	5.76 (-4.1%)	128.4 (-0.6%)
2.00	1.940	4,391 (-29%)	5.74 (-4.4%)	128.4 (-0.6%)
3.00	3.032	4,280 (-31%)	5.70 (-5.1%)	128.5 (-0.5%)
4.00	2.716	4,479 (-28%)	5.75 (-4.3%)	128.4 (-0.6%)

[1]At 100% activity.

The first thing to notice from this table is the two rows with zero target skew, *0.00a* and *0.00b*. These are results from synthesis and optimization with identical starting points and settings. It is therefore expected that the results are identical. However, as the table shows, the result for these rows are very different. The achieved skew is about 0.34 ns better in *0.00a* than in *0.00b*, but the costs are much higher. This is also illustrated in Figure 5.4, where *0.00a* is marked in yellow, and *0.00b* is red.

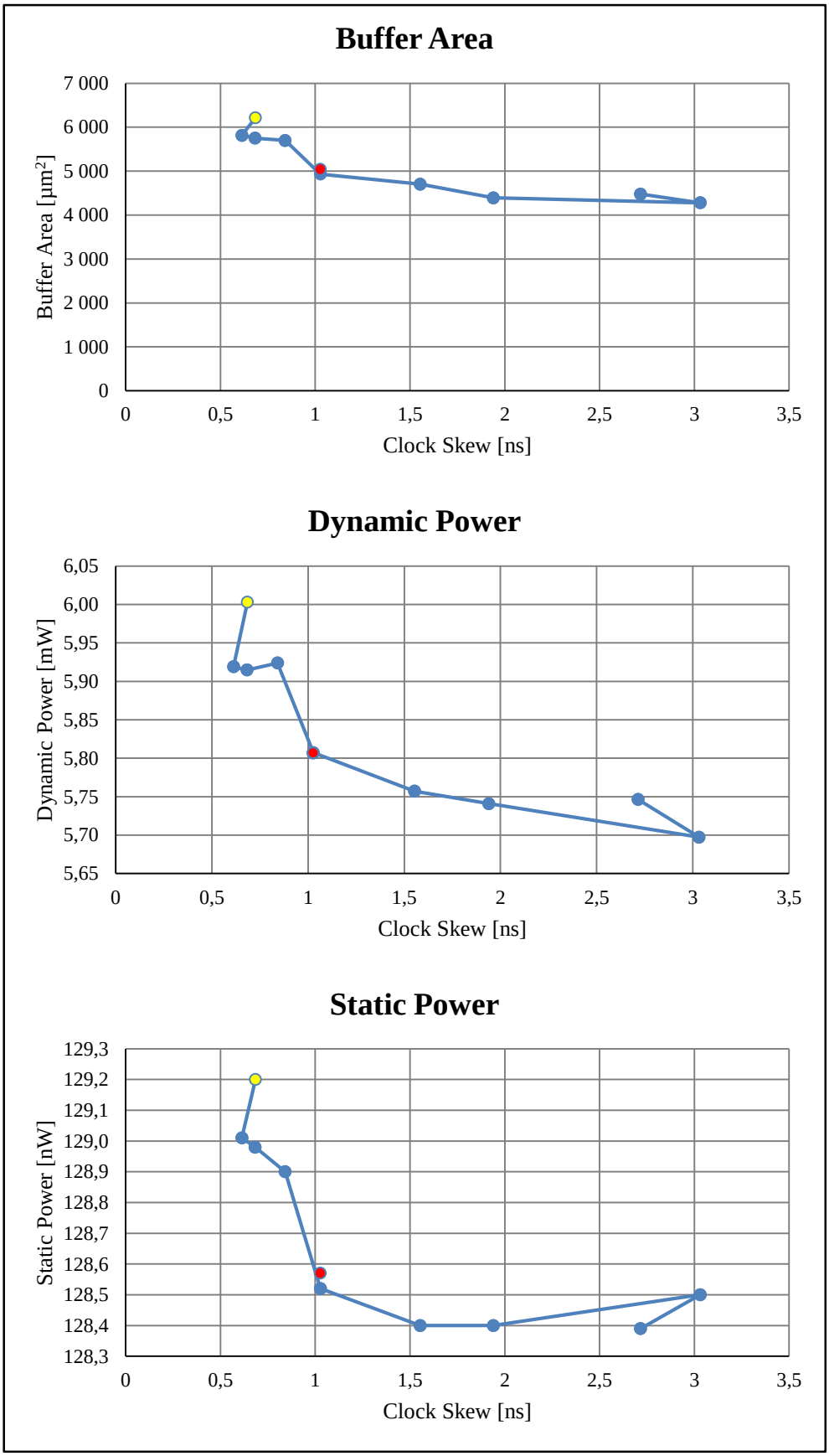

Figure 5.4: Target skew costs vs. achieved skew.

When compared to the results of the other skew targets, the achieved skew and costs for *0.00b* is similar to the target skew of 1 ns. As Figure 5.4 shows, the results of *0.00a* is not close to any other result. The achieved skew of *0.00a* is similar to the target skew of 0.5 ns. However, the costs of *0.00a* is the highest of all the tested targets. Closest in cost is the target skew of 0.25 ns, with 6.4% less buffer area, and 1.4% and 0.1% reduction in dynamic and static power, respectively. However, in addition to lower cost, this target gives 0,072 ns lower skew. This means that *0.00a* gives worse skew at higher cost, than using a target skew of 0.25 ns. This issue is discussed in Section 5.6.

Disregarding the results of *0.00b*, synthesis on the Bluetooth Smart design shows similar behaviour as for the generated test design. Tightening the skew target gives lower achieved skew, but at an increased cost. As Figure 5.4 shows, the cost is significantly higher for skew targets lower than 1 ns. At the target skew of 1 ns, the buffer area is reduced by 21% compared to the zero skew target, *0.00a*. Similarly, the dynamic and static power consumption is reduced by 3.3% and 0.5%, respectively. Compared to the generated design, the relative benefit of increasing the target skew is much lower for the Bluetooth smart design. However, the absolute values of the savings are of comparable sizes. By increasing the target skew, it is possible to save 1 000 to 2 000 μm^2 of buffer area, about 0.3 mW of dynamic power and 0.5 – 1.0 nW of static power.

As previously mentioned, these savings are less significant in the Bluetooth Smart design. The overall synthesis results are for this design significantly worse for all parameters. For the Bluetooth Smart design, the tool is not even close to achieving as low skew as in the generated design. In addition, the costs are significantly higher, especially the static power consumption. This can be explained by the difference in size and complexity of the clock tree of the two different designs. The generated design is a simple design with only ICGs in the clock tree, while the clock tree of the Bluetooth Smart is much more complex with a PLL, and several clock dividers and multiplexers in the clock tree. These cells cause imbalance and add cost to the clock tree synthesis.

The benefit of lower skew is improved timing performance of the clock tree. With low skew, the difference in the clock-edge arrival time is low. This is illustrated in the clock edge arrival time histograms in Figure 5.5. For a tight skew target, the achieved clock skew is low, resulting in a narrow peak in the histogram. For a relaxed skew target, the histogram spread is much higher. Another important thing to notice is that the centre of the

histogram is located at a higher value for thigh skew targets. This means that the average delay from the clock source to the sinks (latency) increases with tighter skew targets.

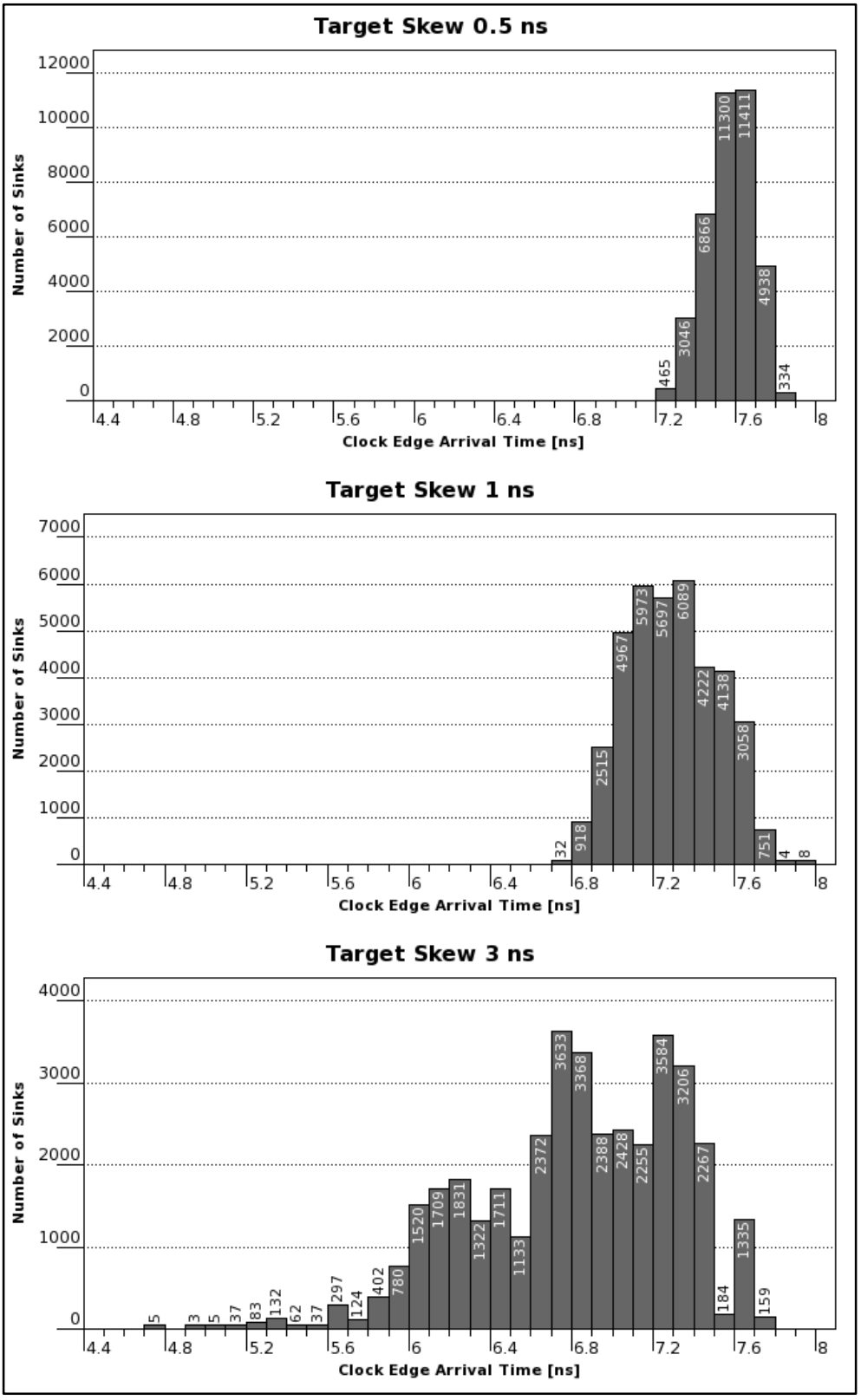

Figure 5.5: Clock edge arrival histogram.

The estimated hold time cost is listed with the achieved skew and the longest path in Table 5.5, where the hold time cost is expressed in terms of the total hold time violation for the clock domain in nanoseconds. As mentioned, it was expected that low skew gives reduced hold time violations. However, as the results show, this is not the case. Instead, the hold time cost is increasing with the tree depth. The hold time cost is clearly highest for target skew of *0.00a* and 0.75. These are also the only results with paths longer than 10 ns. The longest paths of the other results are closer to 8 ns, and the hold time cost is reduced by up to 83%.

Table 5.5: Bluetooth Smart target skew hold time cost.

Target Skew [ns]	Clock Skew [ns]	Longest Path [ns]	Hold Time Cost [ns]
0.00 a)	0.685	10.235 -	1,270.6 -
0.00 b)	1.026	7.818 (-24%)	334.1 (-74%)
0.25	0.613	7.847 (-23%)	425.5 (-67%)
0.50	0.683	7.893 (-23%)	413.2 (-67%)
0.75	0.841	10.223 (-0.1%)	1,190.6 (-6.3%)
1.00	1.027	7.798 (-24%)	296.7 (-77%)
1.50	1.554	7.798 (-24%)	231.0 (-82%)
2.00	1.940	7.7 (-25%)	216.1 (-83%)
3.00	3.032	7.747 (-24%)	216.7 (-83%)
4.00	2.716	8.762 (-14%)	333.1 (-74%)

The correlation between long clock tree paths and high hold time cost is caused by unbalanced sinks in the clock domain. These sinks have been intentionally excluded from the clock tree synthesis, and are therefore not balanced. A sink can be excluded from synthesis for several reasons, e.g. if the register is not part of normal functionality. Due to the lack of insight to the Bluetooth Smart design, the reason for excluding these sinks are not known. It is therefore not known if the hold time violations are critical to the design behaviour and require fixing. Further examinations of the hold time violations are therefore needed. However, the results still indicate that the tree depth is more important than the clock skew for normal operation mode.

As mentioned in Section 2.4.4, hold time constraints are typically more challenging in the scan mode, due to the short data paths. The scan clock has not been considered in this project, and has not been synthesized. It has therefore not been possible to examine the hold time cost in the scan mode.

5.3 Maximum Transition Time Constraint

As presented in Section 3.3.1, the maximum transition time constraints set the maximum allowed transition time in the clock nets. To meet these constraints, buffers are inserted in the clock tree. Therefore, it is expected that tight transition constraints result in higher buffer area cost and power consumption.

An important benefit of low transition time is decreased propagation delay through the cells in the clock tree. In addition, it affects the register delay. Figure 5.6 shows the propagation delay from the clock pin, *Clk*, to the data output, *Q*, as a function of the clock-signal transition time. The values are obtained from the timing description of the register in the cell library and are listed in Appendix D. The maximum allowed clock transition time is 10 ns, which gives more than 3 ns delay. For a clock tree in the 10 to 100 MHz range, this is too slow. Therefore, much shorter transition times are examined in this project. At the default transition constraint of 0.5 ns, the delay is about 0.6 ns.

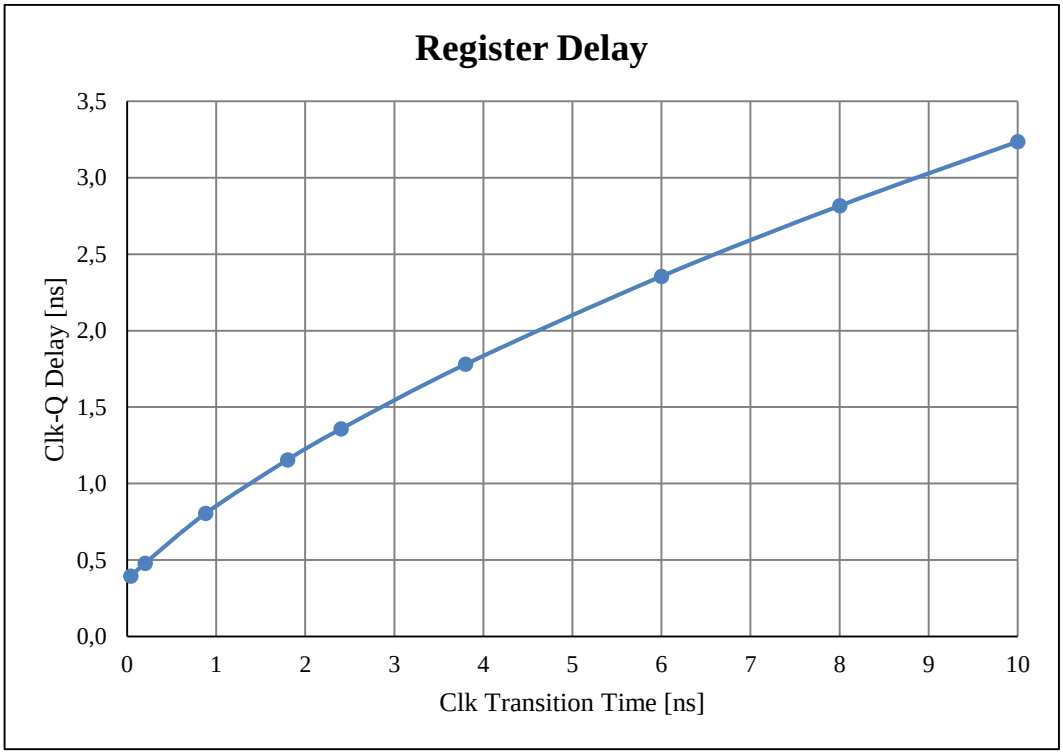

Figure 5.6: *Clk* to *Q* propagation delay.

In order to examine the impact of the max transition constraint, synthesis has been tested with values both tighter and more relaxed than the default 0.5 ns. In addition, the

leaf_max_transistion option has been tested in order to examine if it can be beneficial to set a tighter transition time for the leaf nets only.

5.3.1 Generated Test Design Results

The synthesis results from the maximum transition-time constraint tests on the generated design are presented in Table 5.6. These results show a small benefit in buffer area and power consumption when increasing the constraint to 1 ns. However, increasing the constraint further to 1.5 ns does not have any effect. For both constraints, the achieved skew is slightly worse than the default setting.

Table 5.6: Max transition parameter results for the generated design.

Max Transition [ns]	Clock Skew [ns]	Buffer Area [µm²]	Dynamic Power[1] [mW]	Static Power [nW]
0.25	0.162 (+6.6%)	7,872 (110%)	4.86 (+19%)	6.35 (+44%)
0.50	0.152 -	3,756 -	4.09 -	4.42 -
1.00	0.190 (+25%)	3,294 (-12%)	3.91 (-4.4%)	3.81 (-14%)
1.50	0.193 (+27%)	3,294 (-12%)	3.92 (-4.2%)	3.81 (-14%)

[1]At 100% activity.

Tightening the max transition to 0.25 ns gives significant increase in costs, compared to default settings. The buffer area is more than doubled, and the increase in dynamic and static power is 19% and 44%, respectively. In addition, the tool is not able to achieve as low skew as in the default setting. The benefit with the tight constraint is about 0.1 ns reduction of the delay through the registers.

Table 5.7: Leaf max transition parameter results on the generated design.

Leaf Max Transition [ns]	Clock Skew [ns]	Buffer Area [µm²]	Dynamic Power[1] [mW]	Static Power [nW]
0.10	0.519 (+242%)	20,438 (+444%)	6.56 (+60%)	10.9 (+147%)
0.25	0.164 (+8.0%)	6,884 (+83%)	4,70 (+15%)	5.84 (+32%)
0.50	0.152 -	3,756 -	4.09 -	4.42 -

[1]At 100% activity.

This propagation benefit can also be obtained by only setting the leaf node transition time constraint. Results for this is shown in Table 5.7, where the leaf node transition time constraint is tightened to 0.25 and 0.1 ns, while the transition time constrain is kept at 0.5 ns. With a leaf transition constraint of 0.25 ns, the same 0.1 ns delay reduction is achieved. Compared to the case where the transition time constraint is set for the whole tree, the cost has been reduced.

Reducing the leaf transition time to 0.1 ns gives about 0.16 ns shorter propagation delay in the registers, compared to default settings. However, as the results shows, this is very expensive with respect to achieved skew, buffer area and power consumption. This increased cost cannot be justified by the reduced register delay.

Due to the high cost of tightening the transition time constraint for the whole tree, this has not been tested for the Bluetooth Smart design. However, tightening the transition time constraint for the leaf node has been examined. In addition, benefits of relaxing the transition time, both in the leaf level and the whole tree, has been examined.

5.3.2 Bluetooth Smart Design Results

The results obtained by relaxing the max transition constraint in the clock synthesis of the Bluetooth Smart design are shown in Table 5.8. As this table shows, the cost is increasing for all parameters. This is the opposite behaviour of the generated design, where the cost decreased with increasing transition constraint. The cost increase has been observed for a range of target skew and can therefore not be explained by random variations in the synthesis. A possible explanation is that the increased transition time increases the delay through the gates in the clock tree. This can increase the unbalance in the clock tree, which adds additional costs. However, the exact reason is unknown.

Table 5.8: Max transition parameter results on the Bluetooth Smart design.

Leaf Max Transition [ns]	Clock Skew [ns]		Buffer Area [µm^2]		Dynamic Power[1] [mW]		Static Power [nW]	
0.5	0.683	-	5,751	-	5.91	-	129.0	-
1.0	0.685	0.3%	6,073	5.6%	6.02	1.8%	129.1	0.1%

[1]At 100% activity.

When tightening the leaf transition constraint, the behaviour is similar to the generated test design. As shown in Table 5.9, decreasing the transition time in the leaf nodes, gives worse achieved skew, higher buffer area and increased power consumption. The cost increase is very high and cannot be justified by improved propagation delay in the registers.

Table 5.9: Leaf max transition parameter results on the Bluetooth Smart design.

Leaf Max Transition [ns]	Clock Skew [ns]		Buffer Area [µm^2]		Dynamic Power[1] [mW]		Static Power [nW]	
0.25	0.788	16%	8,200	43%	6.40	8.3%	132.0	2.3%
0.50	0.683	-	5,751	-	5.91	-	129.0	-

[1]At 100% activity.

In total, the examination of the transition time constraints on the Bluetooth Smart design has shown that the best results are achieved with the default settings of 0.5 ns.

5.4 Buffer Selection

The sub-100 nm technology library used in this project has several buffers and inverters that can be used in the clock tree synthesis. These have different characteristics with respect to size, drive strength, etc. As mentioned in Section 2.4.2, in order to achieve good clock performance, the delay through a clock buffer should be equal on both edges. If it is not, the duty of the clock becomes unbalanced. Therefore, the selected buffers and inverters are analysed prior to the synthesis, and unbalanced buffers are removed. The technology library contains a set of special clock tree buffers and inverters that should fulfil these requirements. It is therefore expected that none of these will removed by the tool. The synthesis log has been examined to verify this.

As mentioned is Section 2.4.2, a buffer typically consist of an even number of inverters. Therefore, it could be beneficial to use inverters in the synthesis to divide and distribute the drive strength in the tree. The effects of using inverters is tested by using two different buffer selections. The first selection consists of buffers only. This is the standard selection. The second selection contains both buffers and inverters. Both selections have been tested for default constraints and a range of target skew values.

5.4.1 Generated Test Design Results

In the buffer analysis prior to synthesis, no buffers or inverters are removed due to unbalanced edge delay. This is as expected, since only dedicated clock tree buffers and inverters are used. However, the two smallest buffers and the two smallest inverters are removed due to their low drive strength. These cells are not able to meet the default timing constraints and should not be used in the synthesis. Removing these cells from the buffer selection can improve the synthesis results, but also the synthesis runtime.

Table 5.10 shows the synthesis results with zero target skew, for the selections with buffers only and both buffers and inverters. When using inverters in the synthesis, the tool is not able to achieve the same skew as when using buffers only. However, while the buffer area is 15% higher, there is no significant change dynamic power consumption and the static power is reduced by 4.6%. The number of buffers, which also counts inverters, is significantly increased. This is because the inverters have to been inserted in pairs to keep

the correct clock phase. However, this increase does not give an equal increase for the other cost parameters, because the inverters are smaller and consumes less power.

Table 5.10: Buffer selection results.

Parameter	Buffers only	Buffers and inverters
Clock Skew [ns]	0.150	0.225 (+50%)
Number of Buffers	1,029	1,707 (+66%)
Buffer Area [µm^2]	3,633	4,192 (+15%)
Dynamic Power A100 [mW]	3.95	3.96 (+0.3%)
Static Power [nW]	4.37	4.17 (-4.6%)

In Figure 5.7, the cost parameters are plotted as functions of the achieved clock skew. As these plots show, the difference in power consumption is very small. The dynamic power is more or less identical for skew larger than 0.5 ns. However, for skews smaller than 0.5 ns, the trade-off between dynamic power consumption and clock skew is worse when using inverters in the synthesis. The trade-off is similar for the static power consumption. At high skew, using inverters is beneficial, but at low skew using only buffers gives the best results.

The motivation for using inverters is to divide and distribute the drive strength in the clock tree. This means that some buffers would be replaced by two inverters each, if the synthesis tool finds it beneficial. Naturally, this increases the total number of buffers as observed in Figure 5.7. In addition, dividing the buffers into inverters increases the overhead area. This effect can explain the increase in buffer area.

Another important factor is the edge dependant cells in the clock tree. ICGs and clock dividers are edge dependant, and the phase can therefore not be modified. This makes the use of inverters more complicated. Therefore, using inverters can be more beneficial in simple clock tree without edge dependencies.

Due to the better skew, smaller buffer area and lower dynamic power consumption, synthesis with buffers only is preferred.

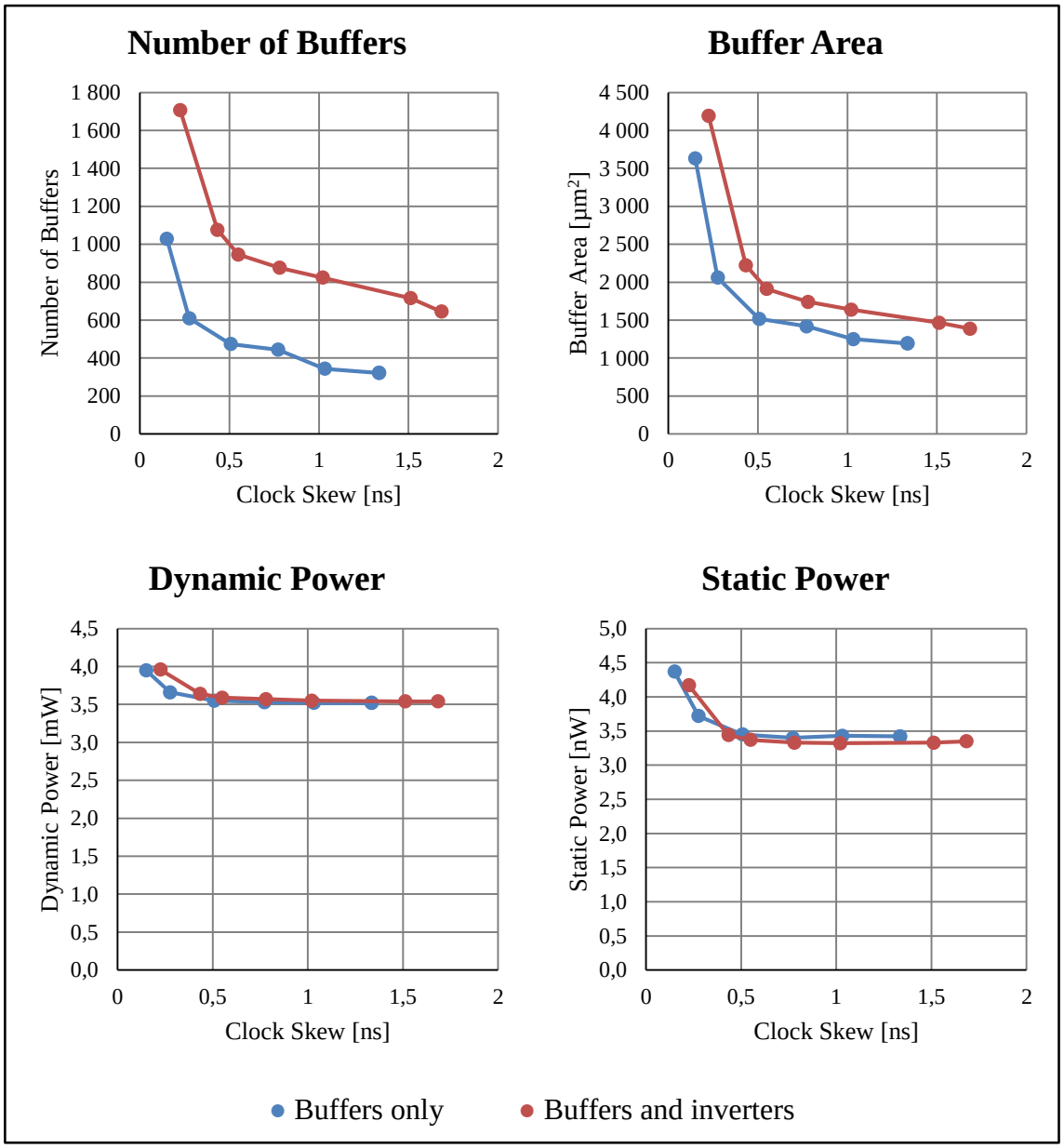

Figure 5.7: Buffer selection costs vs. achieved skew.

5.5 Logic Level Balancing

As mentioned in Section 3.3.4, it is stated by Hsaio [28] that the best skew is achieved using the logic level balancing option. With this feature enabled, the tree is synthesized so that all sinks are placed on the same logic level in the tree. In order to achieve this balance, additional buffers have to be added in short branches. Therefore, it is expected that the improved skew comes at high buffer area and power consumption costs.

In this exploration, it is desirable to examine the benefits and drawbacks of using logic level balancing. Therefore, the synthesis result obtained using logic level balancing has

been compared to results from regular synthesis. For both modes, synthesis has been performed with the default constraints and a range of target skew.

5.5.1 Generated Test Design Results

The logic level balancing option increases the synthesis time considerably. With the option enabled, the synthesis runtime increased from about 10 to 43 minutes on the generated design with 40 000 registers and 5 levels of ICGs. This increase was observed on the test server with no other active tasks.

A comparison between the standard mode and the mode using logic level balancing is shown for the zero skew target in Table 5.11. As the results show, the best skew is actually achieved in the standard mode, but the cost is much higher. This observation is the opposite of what was expected. However, this can be explained by examining Figure 5.8, where the cost parameters are plotted with the achieved skew on the horizontal axes.

Table 5.11: Logic level balancing results.

Parameter	Standard	Logic Level Balancing
Clock Skew [ns]	0.150	0.407 (+172%)
Number of Buffers	1,029	618 (-40%)
Buffer Area [µm^2]	3,633	2,117 (-42%)
Dynamic Power A100 [mW]	3.95	3.88 (-1.8%)
Static Power [nW]	4.37	3.69 (-16%)

As Figure 5.8 shows, the achieved skew when using logic level balancing is never higher than about 0.5 ns, even with relaxed skew target. This is because the clock skew prior to the optimization process, marked with a black outline in the figure, is much lower than the skew target. In standard mode, the achieved skew prior to optimization is much higher at about 1.4 ns. The cost is however significantly lower.

In standard mode, buffers are inserted in the optimization process to balance the tree. This reduces the clock skew, but increases the costs. With logic level balancing, however, the clock structure is fixed. Inserting buffers in the tree will break the logic level balance, and can therefore not be done. Instead, the only possible optimization step is to resize and move the existing buffers. This gives only a small improvement in the skew, accompanied by a small increase in cost.

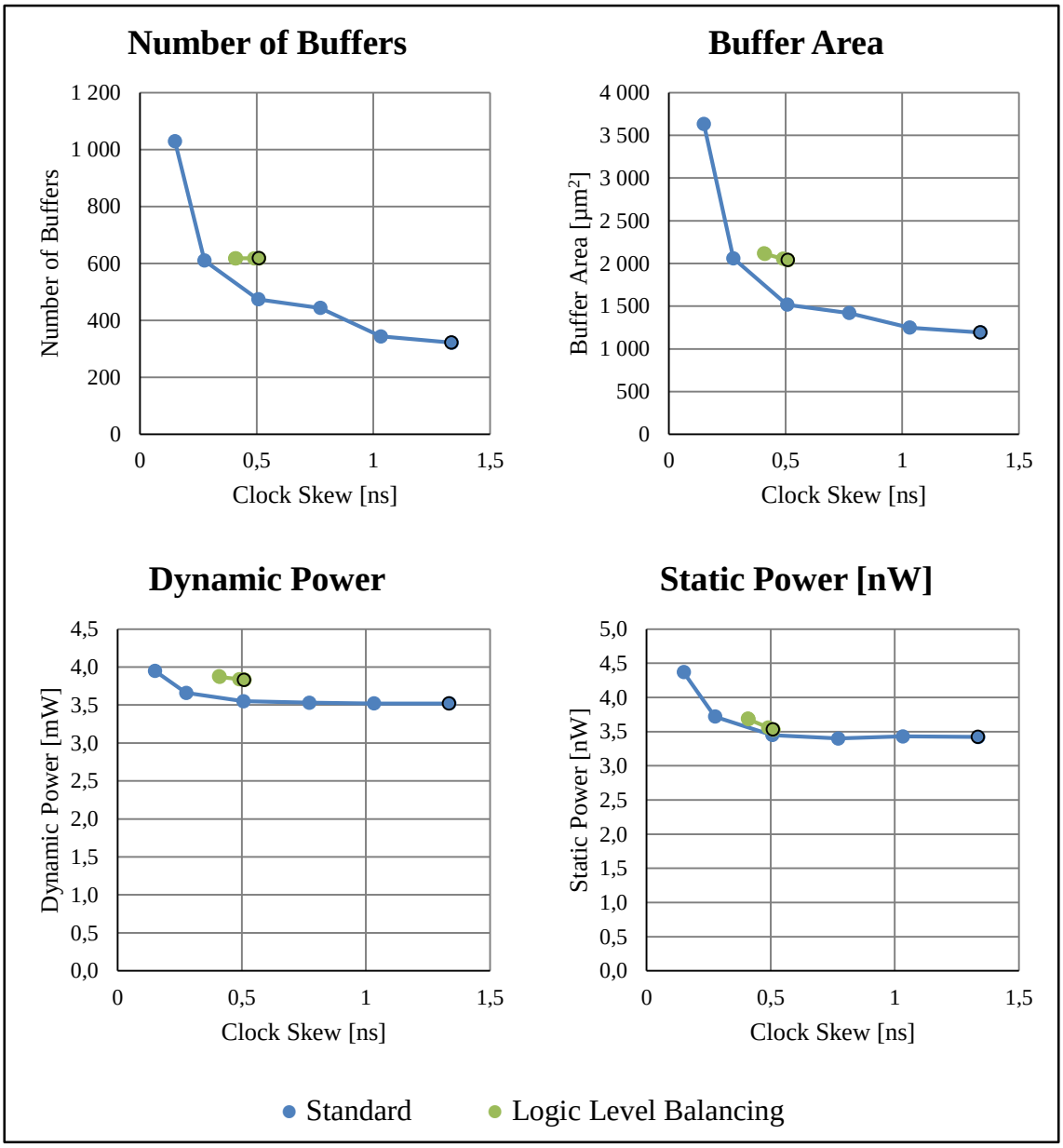

Figure 5.8: Logic level balancing cost vs. achieved skew.

Figure 5.8 shows that the logic level balancing feature is not able to provide as low skew as the standard mode, and the costs are higher. Logic level balancing is therefore not beneficial for a design with this complexity. However, logic level balancing can be a valid option for simple clock structures.

5.6 Synthesis Inconsistency

As discovered in Section 5.2.2, two clock tree synthesises with identical starting point and setting can give different results. Inconsistencies have also been observed for other settings

and designs. The inconsistencies can be separated into two groups: minor differences at relaxed skew targets, and major differences at tight targets.

The minor differences at relaxed skew targets are most likely caused by a random factor in the synthesis and optimization algorithms. As the differences are small, the randomness does not have any major impact on the results. However, the randomness makes it difficult to reproduce results and adds some uncertainty to the quality of the results.

For tight skew targets, the differences are much larger and have a significant impact on the results. In the two tests with zero target skew presented in Section 5.2.2, the difference was very high. In the first test, *0.00a*, the optimization was very expensive in terms of buffer area and power consumption, without much reduction in the clock skew. Many buffers had been inserted, increasing the buffer area, power consumption and longest path. In the second test, *0.00b*, the optimization stopped much earlier. The results of this test are comparable to results obtained using a target skew of 1 ns, with respect to both in target skew and costs, in terms of buffer area and power consumption.

These major differences are only observed for tight skew targets, indicating a weakness in the optimization process. Sometimes, the optimization process seems to get stuck in a state where it continues to insert buffers without achieving any improvement in the skew. This results in very high costs, and should be avoided. This weakness is especially important as the default skew target is zero. It is with this target the largest differences have been observed.

The randomness in the optimization affects the final result of the clock tree synthesis. In order to avoid this randomness, multiple synthesises, with identical skew target, should be performed. The best result can then be selected.

5.7 Summary

The experiments in the exploration of the clock tree synthesis in Synopsys® IC Complier have been summarized in Table 5.12.

Table 5.12: Clock Tree Synthesis Exploration Summary

Name	Purpose	Method	Results
Clock Size and Complexity	Examine the effects of the clock size in terms of number of registers and the complexity in number of ICG levels.	Performing synthesis on artificial clock trees with 1, 8 and 32 thousand registers and 0-5 levels of clock gating.	Linear relationship between clock size and costs in buffer area and power consumption. However, increasing the number of ICGs levels increase the synthesis costs, but reduce the dynamic power consumption in low power modes.
Target Skew	Explore how the optimization process behaves for different target skews.	Testing a range of target skew values, from no to full optimization, on both an artificial and a real design	Significant increase in buffer area and power consumption with tight target skew. Hold time violations was not found to be an issue in the normal mode when relaxing the skew target.
Max Transition	Study the effect of tightening or relaxing the max transition constraints.	Performing synthesis on an artificial and a real design with various max transition and leaf transition constraints.	Tightening the transition time constraint adds significant costs, especially in the leaf nodes. Relaxing the constraint shows some benefits in the artificial design, but no improvements in the real design.
Buffer Selection	Check the influence of the buffer selection on the synthesis.	Performing synthesis with two different selections; buffers only and both buffers and inverters.	Weak buffers/inverters are removed prior to synthesis. Using inverters degrades the synthesis results, except for the static power consumption.
Logic Level Balancing	Examine if the logic level balancing option gives better skew, and at what cost.	Comparing synthesis results with and without this option enabled	Low flexibility in the optimization with logic level balancing enabled. Better results are obtained with this option disabled. Increases synthesis runtime.

6 Clock Gating Experiment

As mentioned in Section 4.1.2, the Bluetooth Smart design has one level of module-level clock gating in the high frequency domain. This level consists of several hundred ICGs, located close to the modules and the leaf levels of the clock tree, and far away from the root. This means that several buffers are inserted between the clock root and the module level ICGs. These buffers are never clock gated, and consume power whenever the high frequency clock is enabled. Therefore, the potential of adding another ICG level closer to the clock root, thus reducing the power consumed in these buffers, has been examined.

The purpose of this experiment has not been to implement a new functional ICG level, but to examine how an additional ICG level affects the clock tree performance. The logic for controlling the new ICGs has therefore not been implemented. However, if the result of this experiment is positive, a full implementation can be justified.

In the implementation of the new ICG level, described in Section 6.2, the goal has been to reduce the power consumption in a common low power scenario. This scenario is described in Section 6.1. The results of this examination are listed in Appendix C. However, the most important results are presented and discussed in Section 6.3.

6.1 Low Power Scenario

The selected low power scenario is a common sleep scenario where only a high frequency timer is active. This mode is used for short idle periods or when accurate timing is critical. Alternatively, a low frequency counter on the low frequency domain can be used while the high frequency timer is completely turned off. The low frequency counter is preferred for long sleep periods due to improved power performance.

In order to estimate the power consumption in this timer scenario, a power scenario was created. In this scenario, only the ICGs covering the timer module, from the clock source to the registers, are enabled. All other ICGs are disabled. This estimate is not an exact representation of the real timer scenario. In the real case, the leaf-level ICGs are controlled by the hardware, and therefore dependant on the module settings and data input. The leaf-level enable duty of 100%, as used in this power scenario, is not an accurate estimate. However, as the purpose of these experiments is to examine the potential power reduction close to the root of the clock tree, the accuracy is not an issue. When comparing the two different ICG implementations, the fidelity is more important.

6.2 Implementation

The high-frequency timer module is situated in a subtree of the main high frequency clock. As shown in Figure 6.1 a), the existing single-level module-level clock gating implementation of this subtree consists of 167 ICGs, of which two are controlling the timer module in the selected scenario. The fan-out in the in the input net of this ICG level is large. The single clock pin of the ICG at the root of the subtree is driving 167 ICGs, in addition to some ungated registers. In order to be able to drive this net, buffers are inserted. However, these will not be covered by the existing module-level ICGs.

As illustrated in Figure 6.1 b), the new ICG level is inserted between the exiting level and the root of the subtree. The new level consists of only three ICGs, which makes the fan-out at the root much lower. One of the new ICGs is covering the two existing timer-module ICGs. In the timer scenario, only this branch is enabled. The other 165 existing ICGs are divided into two groups of 16 and 149 ICGs, respectively, according to their location in the design hierarchy and layout. An ICG is inserted before each group. Both of these ICGs are disabled in the timer scenario.

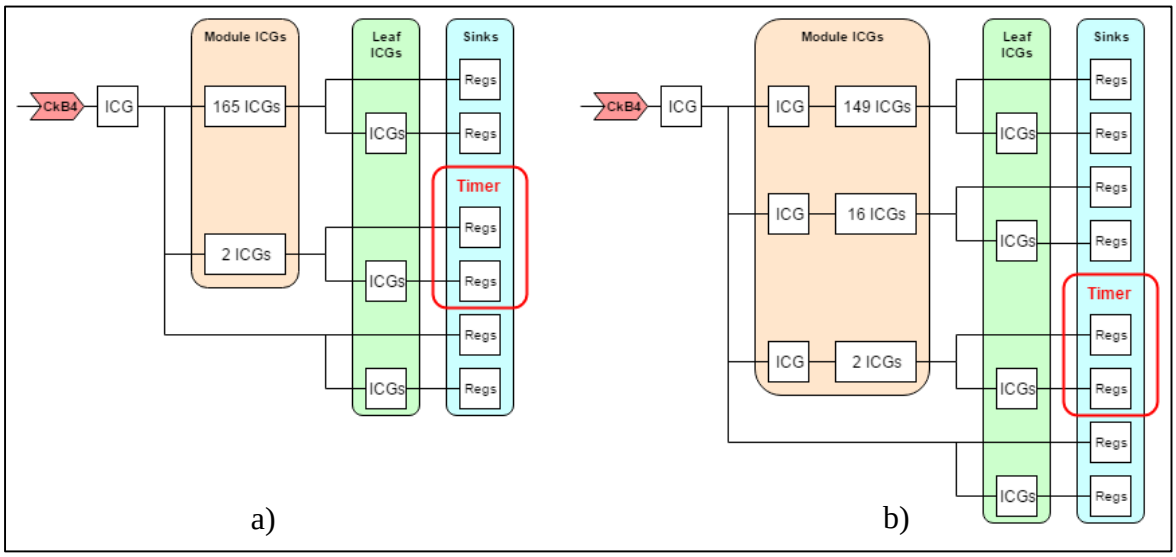

Figure 6.1: Subtree clock structure with a) one and b) two ICG levels.

The new level of clock gates has been implemented using ECO, which allows changes to the synthesized design without any changes in the RTL code. RTL changes are avoided for two reasons. Changes to the RTL code requires detailed knowledge of the design. Due to the limited project time, familiarization of the design has not been possible. In addition, RTL changes would require the whole design to be resynthesized. This is time consuming,

and could also result in a different starting point for the clock tree synthesis. By using ECO, the main features of the existing clock structure is kept, and only modified to include the new ICG level.

Since RTL changes have been avoided, the control circuitry of the new level has not been implemented. This simplification does not affect the clock tree structure, which is the main focus of this study. However, the control circuitry would introduce additional cost in terms of area and power. Instead of implementing the control circuitry, the enable pins of the ICGs are connected to tie-cells. There are two different tie cells, connecting the pin to either logic 0 or logic 1. In addition, the tie cells offer basic ESD protection. In this project, the enable pin is tied to a logic 1 for the new ICGs. However, for power analysis, this is overridden to set the correct scenarios.

The new clock tree structure has been synthesized with default constraints for a range of target skew values, to examine how the new ICG level affect the synthesis. The new ICG level makes the clock structure more complex. Therefore, it is expected to have a negative impact on the clock skew, buffer area and maximum power consumption. The increased cost might, however, be justified by reduced power consumption in the timer scenario.

6.3 Results and Discussion

A summary of the synthesis results with zero target skew is presented in Table 6.1. The modified clock structure with two levels of module-level ICGs is compared with the unmodified structure. With the additional ICG level, the achieved skew is increased by about 0.1 ns. However, the buffer area, and the dynamic and static power consumption are reduced. This is an unexpected result as the additional ICG level increases the complexity and imbalance of the clock tree. However, as presented in Section 5.2.2, the zero skew target results of the unmodified Bluetooth Smart design was abnormally expensive. In order to get a better comparison, the cost has been plotted for the entire range of target skew in Figure 6.2.

Table 6.1: Additional ICG level results.

Parameter	1 Level	2 Levels
Clock Skew [ns]	0.685	0.792 (+16%)
Buffer Area [µm^2]	6,216	5,410 (-13%)
Dynamic Power A100 [mW]	6.00	5.90 (-1.7%)
Static Power [nW]	129.2	128.8 (-0.3%)
Scenario Power [mW]	0.287	0.192 (-33%)

As this figure shows, difference is not as large as the zero skew target results indicated. There is no significant difference in buffer area usage and static power consumption. However, the maximum dynamic power consumption is slightly higher with the additional ICG level.

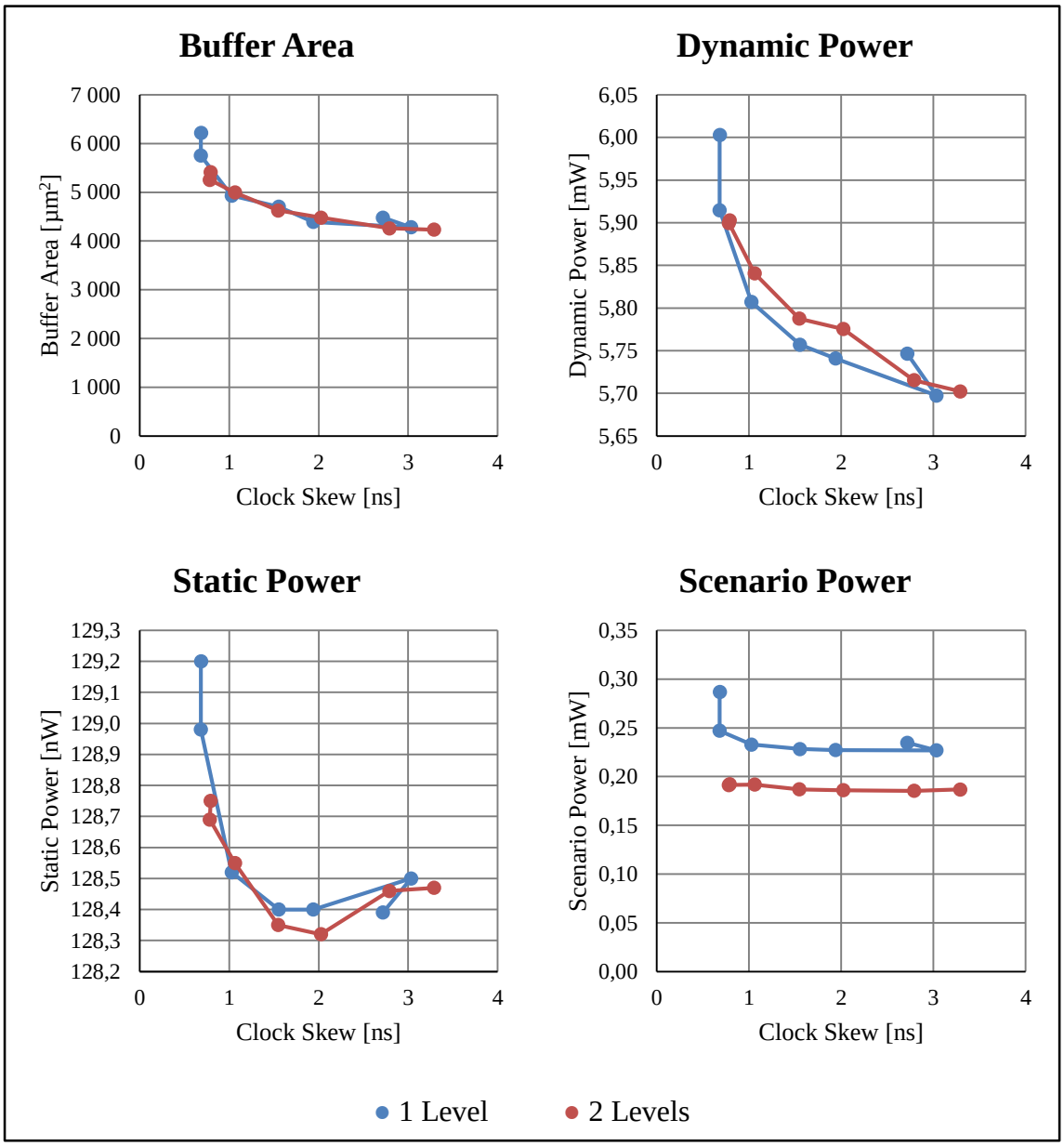

Figure 6.2: Multi level module ICG results.

The most important parameter is the power consumption in the timer scenario. At zero target skew, the reduction is 33% compared to the unmodified design. However, as Figure 6.2 shows, a major part of this reduction is caused by the expensive optimization in the

unmodified design. Ignoring the zero skew target, the power reduction is about 20%. This is still a significant reduction.

7 Clock Tree Synthesis Recommendations

This chapter presents recommendations for clock tree synthesis in Synopsys® IC Compiler™. These recommendations are based on the Synopsys® guidelines presented in Section 3.3.5 and results from the synthesis exploration in Chapter 5. Recommendations for the starting point of the synthesis in given in Section 7.1. In Section 7.2, recommended settings are presented, while a proposed synthesis procedure is explained in Section 7.3.

7.1 Prerequisites

The synthesis starting point is important for the synthesis result. As the experiments in Section 5.1 show, the costs of the clock synthesis increase with the clock complexity. The designer should therefore strive to maintain a simple and balanced clock structure. In addition, all existing buffers in the clock structure should be removed prior to the synthesis, as these can affect the synthesis results as explained in Section 3.3.5. Before the synthesis, it is also recommended to execute the *check_clock_tree* command and solve any reported issues.

7.2 Recommended Settings

7.2.1 Constraints and Targets

The default synthesis constraints are recommended. As presented in Section 3.3.5, Synopsys'® guidelines advice against tightening the constraints. This is supported by experiments on the maximum transition time constraints, described in Section 5.3, where tightening the constraints gave significant increase in costs. In addition, relaxing the transition constraint gave increased costs for the Bluetooth Smart design, as well.

Results presented in Section 5.2 indicate that it can be beneficial to relax the skew target. At the default zero skew target, the optimization can become very expensive, without any major benefits in reduced skew. The results also show that low skew is not necessarily that important with respect to hold time violations in the normal clock mode. A sweep in target skew, where buffer area, power consumption and timing violation are analysed, is therefore recommended.

7.2.2 Buffer Selection

Buffer selection studies, presented in Section 5.4, show that using inverters in the synthesis degrades the synthesis results. This result is most likely caused by the edge dependent cells in the clock tree, such as ICGs. In order to maintain the correct clock phase for these cells,

buffers has to be inserted in pairs, thus losing the main advantage of dividing the drive strength. However, use of inverters can be beneficial in simple clock structure without edge dependant cells. If not, inverters should not be used in the synthesis.

7.2.3 Logic Level Balancing

According to Synopsys® Guideline, the best skew result is obtained using the logic level balancing feature [28]. However, this does not correspond with the experiments presented in Section 5.5. Results indicate that logic level balancing is not beneficial for the tested clock structure, as the buffer area and power consumption cost are higher, while the achieved clock skew is worse. In addition, a significant increase in the synthesis time was observed when this feature is enabled. Logic level balancing can make the clock structure more robust against process variations, but this benefit is traded for worse skew and increased costs. The logic level balancing feature is therefore not recommended.

7.2.4 XOR Self-Gating

XOR self-gating can be used to reduce the power consumption in registers with low activity on the data input, but high activity on the clock. The clock tree synthesis in Synopsys® IC Compiler™ supports automated implementation of this strategy. However, the tool needs accurate simulation data of switching activity in order to determine where XOR self-gating is beneficial. This simulation cannot be a specific corner case, but should rather reflect the average activity of the chip. This is because the tool justifies self-gating a register if the reduction in dynamic power is lower that the extra leakage caused by the additional logic. This leakage contributes to the total power consumption at all times, regardless of the activity level, unless power gating is applied.

Since accurate switching activity simulation data are not available in this project, XOR self-gating has not been examined.

7.3 CTS Procedure

The order of the clock tree synthesis can affect the synthesis results, as the clock buffers and paths occupy chip area. Therefore, a reasonable sequence should be specified in designs with more than one clock domain or mode. It is recommended to start with the clock domains with the highest importance and toughest requirements. The most important clock is typically the main high-frequency clock. The main high frequency clock controls the normal behaviour of the system, and is therefore very important for the reliability.

However, the clock with the toughest requirements might be the scan clock. The scan clock typically covers every register in the design, crossing clock domains, making it the clock tree with the largest number of sinks. In addition, the short data path in the scan chain means that low skew is required in order to avoid timing violations. Therefore, it might be beneficial to start with the scan clock.

As discussed in Section 5.6, the clock tree synthesis is inconsistent, and random variations in the result is expected. Therefore, the design process should never be based on a single clock tree synthesis. Multiple synthesises are needed in order to avoid any random bad results. Observations indicate that this issue is most important for tight skew targets. As mentioned in Section 7.2.1, sweeping the skew target is recommended. By performing a skew sweep, and examining the power consumption and timing violation cost, the designer becomes able to consider trade-offs and make an informed decision.

After the synthesis, any constraint violations should be examined. Depending on the severity, the violations might require manual fixing. In order to simplify debugging of the clock tree synthesis, a naming convention should be used for cells inserted in the different steps of the synthesis. This makes it easy to determine why a specific clock tree buffer was inserted by the tool.

8 Conclusions

The clock tree synthesis functionality of Synopsys® IC Compiler™ has been explored by testing different settings on a generated test design and a real Bluetooth Smart design. Experimental work show that the best synthesis result is obtained with the default transition time constraints of 0.5 ns, combined with using only buffers in the synthesis (avoid using inverters). The logic-level balancing feature is not recommended due to high synthesis cost.

The synthesis results show benefits of increasing the skew target from the default value of zero, both for the generated design and the Bluetooth Smart design. For the generated design, the buffer area was reduced by 2 168 μm^2 (58%) by increasing the target skew to 0.5 ns. Similarly, the dynamic and static power were reduced by 0.41 mW (10%) and 0.94 nW (21%), respectively. Due to the complexity of the clock tree structure in the Bluetooth Smart design, the resulting skew, buffer area and power consumption were significantly higher than the generated design. However, increasing the skew target to 1 ns gave a reduction of 1 284 μm^2 (21%) in buffer area, 0.19 mW (3.3%) in dynamic power consumption and 0.7 nW (0.5%) reduction in static power consumption. If the target skew is further increased from the mentioned values, only small additional benefits are obtained.

Examinations of the hold-time violation cost on the Bluetooth Smart design show that increasing the skew target does not increase the hold-time violation cost. Instead, the results show a reduction in the hold-time cost due to lower latency in the tree. With high clock latency, the holdtime cost associated with intentionally unbalanced sinks of the clock tree increases. Due to the lack of insight of the design, the reason for excluding these sinks are not known. It is therefore uncertain whether these violations are critical to the functionality of the design. Further work is needed in order to examine this.

In this report, a multi-level module-level clock gating strategy has been proposed for the Bluetooth Smart design. In this strategy, another level of module-level clock gates are inserted between the existing single level and the clock source. Three clock gates have been inserted with the purpose of minimizing the power consumed in a common high-frequency timer scenario. Experiments shows power savings in this scenario of about 0.04 mW (~18%), at the cost of a small increase in the worst-case dynamic power consumption in the clock tree, compared to the unmodified design. The buffer area and static power are virtually unaffected. This is a very promising result. However, since RTL

work has been avoided in this project, the necessary control circuitry has not been implemented. Further work of implementing this logic is required to determine whether the multi-level clock gating strategy is beneficial.

8.1 Further Work

As mentioned, the hold-time violation cost associated with the unbalanced sinks needs to be examined. In addition, the hold time cost of the scan mode should also be evaluated. This requires synthesis of the scan clock, which has been avoided in this project. It is expected that the skew requirements are tougher in scan mode, due to the short data paths. Therefore, it would be sensible to examine the scan domain similarly to how the main clock was examined in this project, i.e. evaluating the trade-offs between clock skew, buffer area, power consumption and hold time violations.

As mentioned in Section 3.1, Ramachandran [23] proposes a minimalistic useful-skew approach. Instead of using a zero or bounded global skew target, the arrival time requirements can be derived from the timing constraints of the sinks. The purpose of this approach is to reduce setup and hold-time violations, whilst minimizing the clock tree. This and similar strategies could reduce the buffer area and power consumption, and should therefore be examined.

9 Bibliography

[1] K. Austbø, “Clock Tree Analysis,” Norwegian University of Science and Technology, Trondheim, Norway, 2015.

[2] Juniper Research Ltd, “IoT - Internet of Transformation,” Hampshire, 2015.

[3] ABI Research, “The Internet of Things Will Drive Wireless Connected Devices to 40.9 Billion in 2020,” 20 August 2014. [Online]. Available: https://www.abiresearch.com/press/the-internet-of-things-will-drive-wireless-connect/. [Accessed 12 June 2016].

[4] Bluetooth SIG, Inc., “Volume 6 - Core System Pakage [Low Energy Controller volume],” in *Bluetooth Specification Vesion 4.0*, Bluetooth SIG, Inc., 2010.

[5] J. Decuir, “Introducing Bluetooth Smart, Part 1: A Look at Both Classic and New Technologies,” *IEEE Consumer Electronics Magazine*, pp. 12-18, January 2014.

[6] M. Keating, D. Flynn, R. Aitken, A. Gibbons and K. Shi, Low Power Methodology Manual for System-on-Chip Design, New York: Springer, 2008.

[7] E. G. Friedman, "Clock Distribution Networks in Synchronous Digital Integrated Circuits," *Proceedings of the IEEE,* vol. 89, no. 5, pp. 665-692, 2001.

[8] D. M. Harris and S. L. Harris, Digital Design and Computer Architecture, Second Edition, Burlington: Elsevier Science, 2012.

[9] A. L. Crouch, “Chapter 3 Scan Architectures and Techniques,” in *Design-for-Test for Digital IC’s and Embedded Core Systems*, Austin, Texas, Prentice Hall, 1999.

[10] K. C. Pokhrel, "Physical and Silicon Measures of Low Power Clock Gating Success: An Apple to Apple Case Study," Synopsys Users Group (SNUG), 2007.

[11] Synopsys Inc, IC Compiler Implementation User Guide Version K-2015.06-SP2, Mountain Veiw, CA, 2015.

[12] M. Edahiro, “A Clustering-Based Optimization Algorithm in Zero-Skew Routings,” *Proceedings of the ACM/IEEE Design Automation Conference*, p. 612–616, June 1993.

[13] A. Mehta, Y.-P. Chen, N. Menezes, D. Wong and L. Pileggi, “Clustering and Load Balancing for Buffered Clock Tree Aynthesis,” *Proceedings of the IEEE International Conference on Computer Design,* pp. 217-223, October 1997.

[14] A. Vittal and M. Marek-Sadowska, “Power Optimal Buffered Clock Tree Design,” *Proceedings of the ACM/IEEE Design Automation Conference*, p. 497–502, June 1995.

[15] R. S. Shelar, “An Efficient Clustering Algorithm for Low Power Clock Tree Synthesis,” Intel Corporation, Austin, Texas, 2007.

[16] K. Boese and A. Kahng, “Zero-Skew Clock Routing Trees with Minimum Wirelength,” *Proceedings of the IEEE International ASIC Conference,* pp. 17-21, September 1992.

[17] R.-S. Tsay, “An Exact Zero-Skew Clock Routing Algorithm,” *IEEE Transactions on Computer-Aided Design of Integrated,* pp. 242-249, February 1993.

[18] M. Edahiro, “An Efficient Zero-Skew Routing Algorithm,” *Proceedings of the ACM/IEEE Design Automation Conference*, p. 375–380, June 1994.

[19] V. Nawale and S. Clara, “Optimal Useful Clock Skew Scheduling In the Presence of Variations Using Robust ILP Formulations,” in *IEEE/ACM International Conference on Computer-Aided Design*, San Jose, California, 2006.

[20] H.-M. Chou and S.-C. Chang, “Useful-Skew Clock Optimization for Multi-Power Mode Designs,” in *IEEE/ACM International Conference on Computer-Aided Design*, San Jose, California, 2011.

[21] J. P. Fishburn, “Clock Skew Optimization,” *IEEE Transactions on Computers,* vol. 39, no. 7, pp. 945-951, 1990.

[22] J. G. Xi and W. W.-M. Dai, “Useful-Skew Clock Routing with Gate Sizing for Low Power Design,” *Journal of VLSI Signal Processing,* no. 10, p. 163–179, 1997.

[23] V. Ramachandran, “Construction of Minimal Functional Skew Clock Trees,” *Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design,* pp. 119-120, March 2012.

[24] G. Pouiklis and G. C. Sirakoulis, “Clock Gating Methodologies and Tools: a Survey,” *International Journal of Circuit Theory and Applications,* no. 44, p. 798–816, 2016.

[25] A. Bu, P. W. J. Yu and W. Shan, “Power Optimization and VLSI Design of CPU Based on Adaptive Clock-Gating,” *Journal of Southeast University,* vol. 45, no. 2, pp. 219-223, 2015.

[26] K. Suito, M. Takasu, R. Ueda, K. Fujii, H. Matsutani and N. Yamasaki, “Experimental Evaluation of Low Power Techniques on Dependable Responsive Multithreaded Processor,” in *28th International Conference on Computers and Their Applications 2013,* Honolulu, Hawaii, 2013.

[27] Synopsys Inc, “Understanding Clock Tree Synthesis Log Messages,” 15 May 2012. [Online]. Available: https://solvnet.synopsys.com/retrieve/035726.html. [Accessed 11 June 2016].

[28] C. Hsiao, “IC Compiler Clock Tree Synthesis - Make it Work for You!,” 13 June 2007. [Online]. Available: https://solvnet.synopsys.com/retrieve/020919.html. [Accessed 22 May 2016].

[29] Synopsys Inc, “Recommendations for Improving CTS QoR and Debugging CTS QoR Issues,” 9 July 2013. [Online]. Available: https://solvnet.synopsys.com/retrieve/034505.html. [Accessed 11 June 2016].

[30] G. Sumit, “report_clock_gating for PT-PX - SolveNet,” 18 October 2007. [Online]. Available: https://solvnet.synopsys.com/retrieve/021849.html. [Accessed 12 May 2016].

Appendix A Example ICG Structure Report

```
************************************
Report  : Clock Gating
Design  : simple_test
Version : J-2014.09-SP3
Date    : Wed Jun 15 14:33:31 2016
************************************

Information: Updating switching activity

Information: No clock-gating cell pin names are specified, default names will be used...

-------------------------+-----+------------------------+--------------+--------------+--------------+--------------+----
                         |     |       Subtree          |      EN      |      TM      |    IN CLK    |   OUT CLK    |
Clock gate               | Lvl | Cells Sinks Buffer ICG |  SP     TR   |  SP     TR   |  SP     TR   |  SP     TR   |
-------------------------+-----+------------------------+--------------+--------------+--------------+--------------+----
ck_ExampleClock_Icg_0_1  |   0 |     0    42      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_2  |   0 |     0    87      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_3  |   0 |     0    17      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_4  |   0 |     0    34      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_5  |   0 |     0    54      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_6  |   0 |     0    54      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_7  |   0 |     0    13      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_8  |   0 |     0    79      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_9  |   0 |     0    25      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_10 |   0 |     0    27      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_11 |   0 |     0    29      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_20 |   0 |     0    84      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_21 |   0 |     0    10      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_22 |   0 |     0    15      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_23 |   0 |     0    68      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_24 |   0 |     0    26      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_25 |   0 |     0    44      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_0_26 |   0 |     0    12      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.128 0.0255 | 0.064 0.0128 | OK
ck_ExampleClock_Icg_1_1  |   1 |     7   376      0   7 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.128 0.0255 | OK
ck_ExampleClock_Icg_1_2  |   1 |     4   200      0   4 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.128 0.0255 | OK
ck_ExampleClock_Icg_0_12 |   0 |     0    85      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_13 |   0 |     0    49      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_14 |   0 |     0    10      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_15 |   0 |     0   118      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_16 |   0 |     0    47      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_17 |   0 |     0    16      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_18 |   0 |     0    59      0   0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
```

```
ck_ExampleClock_Icg_0_19 |   0 |    0    77      0    0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_1_3  |   1 |    2   118      0    2 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.128 0.0255 | OK
ck_ExampleClock_Icg_1_4  |   1 |    3   136      0    3 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.128 0.0255 | OK
ck_ExampleClock_Icg_1_5  |   1 |    2    70      0    2 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.128 0.0255 | OK
ck_ExampleClock_Icg_0_27 |   0 |    0    89      0    0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_28 |   0 |    0    84      0    0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_29 |   0 |    0    26      0    0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_30 |   0 |    0    60      0    0 | 0.500 0.0100 | 0.000 0.0000 | 0.253 0.0505 | 0.127 0.0254 | OK
ck_ExampleClock_Icg_0_31 |   0 |    0    80      0    0 | 0.500 0.0100 | 0.000 0.0000 | 0.252 0.0504 | 0.128 0.0256 | OK
ck_ExampleClock_Icg_2_1  |   2 |   21  1152      0   21 | 0.500 0.0100 | 0.000 0.0000 | 0.500 0.1000 | 0.253 0.0505 | OK
ck_ExampleClock_Icg_2_2  |   2 |   14   648      0   14 | 0.500 0.0100 | 0.000 0.0000 | 0.500 0.1000 | 0.253 0.0505 | OK
ck_ExampleClock_Icg_1_6  |   1 |    1   100      0    1 | 0.500 0.0100 | 0.000 0.0000 | 0.500 0.1000 | 0.252 0.0504 | OK
ck_ExampleClock_Icg_0_32 |   0 |    0    34      0    0 | 0.500 0.0100 | 0.000 0.0000 | 0.500 0.1000 | 0.253 0.0506 | OK
ck_ExampleClock_Icg_0_33 |   0 |    0    46      0    0 | 0.500 0.0100 | 0.000 0.0000 | 0.500 0.1000 | 0.253 0.0506 | OK


************************************************************
Clock-Gating Summary:
====================
        Design total:
        Number of clock-gating cells     : 41
        Number of registers              : 2000
        Number of buffers                : 0

        Level 0:
        Number of clock-gating cells     : 33
        Number of clock-gated buffers    : 0
        Number of clock-gated registers  : 1600 (80.00%)
        Largest register bank size       : 118
        Average register bank size       : 48.5
        Smallest register bank size      : 10
        Observed enable probability SP   : 0.50
        Observed enable probability TR   : 0.50

        Level 1:
        Number of clock-gating cells     : 6
        Number of clock-gated buffers    : 0
        Number of clock-gated registers  : 1000 (50.00%)
        Largest register bank size       : 376
        Average register bank size       : 166.7
        Smallest register bank size      : 70
        Observed enable probability SP   : 0.50
        Observed enable probability TR   : 0.50
```

```
        Level 2:
        Number of clock-gating cells     : 2
        Number of clock-gated buffers    : 0
        Number of clock-gated registers  : 1800 (90.00%)
        Largest register bank size       : 1152
        Average register bank size       : 900.0
        Smallest register bank size      : 648
        Observed enable probability SP   : 0.51
        Observed enable probability TR   : 0.51

************************************************************
```

Appendix B Generated Design Results

Synthesis results for the generated test design are listed in Table B.2 to B.6. The settings presented in Table B.1 are used, unless other settings are specified in the tables.

Table B.1: Default synthesis settings.

Setting	Value
Max transition	0.5 ns
Leaf max transition	0.5 ns
Max capacitance	0.6 pF
Max fan-out	2000
Target early delay [ns]	0
Logic level balancing	Disabled
Buffer selection	Buffers only

Table B.2: Synthesis results for 1 000 register design size.

ICG Levels	Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm^2]	Dynamic Power [mW] A0[1]	Dynamic Power [mW] A100[1]	Static Power [nW]
None	0.00	0.033	68.6	0.0583	0.0583	0.0294
	0.25	0.032	68.6	0.0573	0.0573	0.0294
	0.50	0.032	68.6	0.0573	0.0573	0.0294
1	0.00	0.086	103.9	0.0441	0.0940	0.1120
	0.25	0.250	40.6	0.0365	0.0831	0.0846
	0.50	0.250	40.6	0.0364	0.0830	0.0846
2	0.00	0.097	119.3	0.0166	0.1010	0.1270
	0.25	0.248	42.8	0.0138	0.0879	0.0968
	0.50	0.299	36.4	0.0138	0.0864	0.0927
	0.75	0.534	34.7	0.0138	0.0864	0.0933
3	0.00	0.095	123.8	0.0129	0.1000	0.1310
	0.25	0.237	54.0	0.0118	0.0897	0.1030
	0.50	0.464	35.0	0.0094	0.0850	0.0950
	0.75	0.739	27.7	0.0095	0.0843	0.0940
4	0.00	0.118	108.9	0.0101	0.1020	0.1310
	0.25	0.239	55.2	0.0086	0.0921	0.1060
	0.50	0.500	47.0	0.0087	0.0907	0.1040
	0.75	0.759	38.9	0.0084	0.0890	0.0953
	1.00	0.881	31.6	0.0084	0.0884	0.0948
	1.50	1.267	30.0	0.0085	0.0886	0.0959
5	0.00	0.108	120.1	0.0066	0.0984	0.1280
	0.25	0.249	90.2	0.0073	0.0936	0.1180
	0.50	0.461	52.9	0.0048	0.0869	0.1000
	0.75	0.742	38.4	0.0043	0.0853	0.0975
	1.00	0.936	37.5	0.0041	0.0869	0.1020
	1.50	1.393	30.2	0.0032	0.0860	0.1000
	2.00	1.710	26.9	0.0027	0.0855	0.0991

[1]At 0% and 100% activity.

Table B.3: Synthesis results for 8 000 register design size.

ICG Levels	Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm²]	Dynamic Power [mW] A0[1]	Dynamic Power [mW] A100[1]	Static Power [nW]
None	0.00	0.082	523.3	0.4770	0.4770	0.2240
	0.25	0.085	509.0	0.4700	0.4700	0.2170
	0.50	0.086	509.0	0.4690	0.4690	0.2170
1	0.00	0.117	614.6	0.3090	0.6700	0.7410
	0.25	0.250	262.9	0.2540	0.6080	0.5950
	0.50	0.440	245.6	0.2510	0.6040	0.5850
2	0.00	0.112	821.2	0.0934	0.7340	0.8710
	0.25	0.255	365.7	0.0825	0.6520	0.6810
	0.50	0.484	235.5	0.0724	0.6290	0.6260
	0.75	0.754	186.8	0.0719	0.6260	0.6250
	1.00	0.866	175.6	0.0719	0.6250	0.6260
3	0.00	0.141	771.1	0.0869	0.7390	0.8610
	0.25	0.259	461.7	0.0748	0.6820	0.7330
	0.50	0.499	308.3	0.0646	0.6510	0.6620
	0.75	0.764	254.2	0.0580	0.6450	0.6530
	1.00	1.022	194.9	0.0586	0.6420	0.6540
	1.50	1.022	191.2	0.0580	0.6420	0.6530
4	0.00	0.164	632.2	0.0423	0.7230	0.8210
	0.25	0.259	411.0	0.0375	0.6820	0.7350
	0.50	0.496	278.0	0.0336	0.6550	0.6700
	0.75	0.747	213.9	0.0336	0.6500	0.6680
	1.00	1.009	208.3	0.0330	0.6510	0.6710
	1.50	1.326	191.2	0.0330	0.6510	0.6710
5	0.00	0.135	813.4	0.0242	0.7740	0.9220
	0.25	0.256	446.3	0.0229	0.7100	0.7690
	0.50	0.510	315.6	0.0196	0.6820	0.6980
	0.75	0.734	254.5	0.0188	0.6720	0.6800
	1.00	0.957	231.8	0.0189	0.6710	0.6820
	1.50	1.271	221.8	0.0181	0.6700	0.6800

[1]At 0% and 100% activity.

Table B.4: Synthesis results for 32 000 register design size.

ICG Levels	Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm²]	Dynamic Power [mW]					Static Power [nW]
				A0[1]	A25[1]	A50[1]	A75[1]	A100[1]	
None	0.00	0.093	2140.6	1.890	1.890	1.890	1.890	1.890	0.921
	0.25	0.116	2031.7	1.870	1.870	1.870	1.870	1.870	0.872
	0.50	0.117	2031.7	1.860	1.860	1.860	1.860	1.860	0.872
1	0.00	0.137	2412.5	1.250	1.630	1.990	2.370	2.730	2.900
	0.25	0.266	1190.0	1.050	1.430	1.780	2.150	2.520	2.400
	0.50	0.400	975.8	1.010	1.390	1.740	2.110	2.480	2.280
	0.75	0.595	931.6	1.010	1.390	1.740	2.110	2.470	2.290
	1.00	0.589	926.0	1.010	1.380	1.740	2.110	2.470	2.290
2	0.00	0.135	2877.8	0.364	0.803	1.380	2.050	2.910	3.280
	0.25	0.266	1513.1	0.321	0.715	1.240	1.870	2.680	2.720
	0.50	0.517	919.5	0.286	0.662	1.170	1.780	2.570	2.480
	0.75	0.782	802.5	0.283	0.657	1.170	1.780	2.560	2.470
	1.00	0.973	695.0	0.283	0.656	1.160	1.770	2.550	2.470
3	0.00	0.151	2690.5	0.281	0.668	1.200	1.950	2.970	3.240
	0.25	0.276	1594.3	0.254	0.609	1.110	1.810	2.780	2.790
	0.50	0.519	1044.1	0.230	0.568	1.040	1.730	2.670	2.550
	0.75	0.785	956.5	0.224	0.561	1.030	1.720	2.660	2.530
	1.00	1.038	824.9	0.226	0.562	1.030	1.720	2.650	2.550
	1.50	1.139	814.2	0.226	0.561	1.030	1.720	2.650	2.550
4	0.00	0.144	3083.9	0.195	0.489	1.020	1.820	3.060	3.470
	0.25	0.288	1637.2	0.178	0.438	0.921	1.660	2.810	2.860
	0.50	0.509	1179.1	0.165	0.415	0.878	1.590	2.710	2.600
	0.75	0.789	1029.0	0.165	0.413	0.872	1.580	2.700	2.600
	1.00	0.993	946.7	0.167	0.414	0.873	1.580	2.700	2.610
	1.50	1.286	886.8	0.164	0.408	0.859	1.570	2.680	2.560
	2.00	1.286	890.4	0.164	0.408	0.859	1.570	2.680	2.570
5	0.00	0.150	3114.1	0.024	0.169	0.551	1.420	3.120	3.480
	0.25	0.282	1690.9	0.022	0.153	0.503	1.300	2.860	2.880
	0.50	0.532	1188.9	0.018	0.143	0.478	1.250	2.770	2.650
	0.75	0.754	1062.3	0.019	0.143	0.467	1.240	2.740	2.600
	1.00	1.033	910.8	0.019	0.143	0.467	1.240	2.740	2.610
	1.50	1.369	877.8	0.019	0.143	0.466	1.240	2.740	2.610

[1]At 0%, 25%, 50%, 75% and 100% activity.

Table B.5: Synthesis results for design with 40 000 sinks and 5 ICG levels.

Max Transition [ns]	Leaf Max Transition [ns]	Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm²]	Dynamic Power[1] [mW]	Static Power [nW]
0.25	0.10	0.00	0.169	22639.4	7.05	12.20
		0.25	0.300	21389.4	6.83	11.70
		0.50	0.528	20821.9	6.72	11.50
		0.75	0.753	20474.7	6.65	11.40
		1.00	1.015	20313.4	6.62	11.40
		1.50	1.450	20273.9	6.62	11.40
		2.00	1.485	20266.4	6.62	11.40
	0.25	0.00	0.162	7872.0	4.86	6.35
		0.25	0.286	6872.9	4.69	5.94
		0.50	0.534	6208.5	4.58	5.74
		0.75	0.798	5614.3	4.51	5.68
		1.00	1.029	5528.3	4.50	5.66
		1.50	1.288	5497.0	4.49	5.65
0.50	0.10	0.00	0.519	20438.0	6.56	10.90
		0.25	0.326	21250.0	6.71	11.30
		0.50	0.558	20155.5	6.51	10.80
		0.75	0.798	19975.2	6.47	10.80
		1.00	1.033	19885.0	6.45	10.80
		1.50	1.434	19842.1	6.46	10.80
		2.00	1.715	19837.9	6.46	10.80
	0.25	0.00	0.164	6883.6	4.70	5.84
		0.25	0.284	5685.4	4.49	5.38
		0.50	0.530	5073.3	4.39	5.16
		0.75	0.792	4817.4	4.36	5.16
		1.00	1.050	4462.6	4.34	5.20
		1.50	1.478	4452.3	4.34	5.20
		2.00	1.482	4448.6	4.34	5.20
	0.50	0.00	0.152	3756.2	4.09	4.42
		0.25	0.282	2210.0	3.81	3.78
		0.50	0.511	1587.6	3.68	3.48
		0.75	0.745	1525.7	3.67	3.44
		1.00	1.100	1265.0	3.65	3.45
		1.50	1.386	1229.2	3.64	3.44
1.00	0.10	0.00	0.165	23666.1	7.15	12.20
		0.25	0.299	21926.7	6.84	11.50
		0.50	0.550	20825.2	6.61	11.00
		0.75	0.784	20630.6	6.57	11.00
		1.00	0.969	20545.3	6.54	10.90
		1.50	1.524	20411.1	6.53	10.90
		2.00	2.007	20371.7	6.53	10.90
	0.25	0.00	0.149	7368.0	4.80	5.97
		0.25	0.284	5697.2	4.50	5.32

[1]At 100% activity.

Max Transition [ns]	Leaf Max Transition [ns]	Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm^2]	Dynamic Power[1] [mW]	Static Power [nW]
1.00	0.25	0.50	0.527	4915.4	4.36	5.07
		0.75	0.768	4626.7	4.32	5.04
		1.00	1.025	4557.5	4.31	5.07
		1.50	1.524	4256.5	4.28	5.02
		2.00	1.990	4192.1	4.27	5.02
	0.50	0.00	0.154	3707.2	4.08	4.34
		0.25	0.290	2059.7	3.77	3.67
		0.50	0.522	1381.0	3.64	3.36
		0.75	0.754	1278.8	3.61	3.29
		1.00	1.019	1127.6	3.60	3.32
		1.50	1.330	1057.6	3.60	3.31
		2.00	1.934	1055.9	3.60	3.31
	1.00	0.00	0.190	3293.6	3.91	3.81
		0.25	0.273	1862.8	3.65	3.22
		0.50	0.530	1154.4	3.50	2.88
		0.75	0.767	972.7	3.46	2.82
		1.00	1.035	906.4	3.45	2.81
		1.50	1.480	754.6	3.46	2.89
		2.00	2.017	742.8	3.46	2.89
1.50	0.10	0.00	0.505	21384.7	6.68	11.30
		0.25	0.353	21641.2	6.76	11.40
		0.50	0.544	20959.9	6.61	11.10
		0.75	0.771	20704.3	6.58	11.00
		1.00	0.980	20658.4	6.56	11.00
		1.50	1.554	20575.2	6.56	11.00
		2.00	2.042	20535.2	6.55	11.00
	0.25	0.00	0.149	7368.0	4.80	5.97
		0.25	0.287	5697.2	4.50	5.32
		0.50	0.527	4915.4	4.36	5.07
		0.75	0.765	4626.7	4.32	5.04
		1.00	1.027	4557.5	4.31	5.07
		1.50	1.523	4256.5	4.28	5.02
		2.00	1.989	4192.1	4.27	5.02
	0.50	0.00	0.151	3897.0	4.11	4.42
		0.25	0.287	2130.2	3.78	3.67
		0.50	0.541	1405.9	3.64	3.36
		0.75	0.755	1316.6	3.61	3.30
		1.00	1.022	1201.2	3.61	3.32
		1.50	1.423	1083.6	3.60	3.34
		2.00	1.787	1075.5	3.60	3.34
	1.00	0.00	0.173	3003.3	3.89	3.80
		0.25	0.287	1699.3	3.64	3.25
		0.50	0.545	1125.3	3.50	2.95
		0.75	0.747	1061.5	3.49	2.91

[1]At 100% activity.

Max Transition [ns]	Leaf Max Transition [ns]	Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm^2]	Dynamic Power[1] [mW]	Static Power [nW]
1.50	1.00	1.00	1.012	953.7	3.48	2.93
		1.50	1.504	795.5	3.46	2.90
		2.00	1.763	790.4	3.46	2.90
	1.50	0.00	0.193	3293.6	3.92	3.81
		0.25	0.292	1841.8	3.74	3.29
		0.50	0.536	1283.0	3.60	2.97
		0.75	0.801	1163.4	3.58	2.95
		1.00	1.018	1065.1	3.57	2.98
		1.50	1.523	930.7	3.56	3.00
		2.00	1.575	923.7	3.56	2.99

[1]At 100% activity.

Table B.6: Synthesis results for design with 40 000 sinks and 5 ICG levels.

Mode	Target Skew [ns]	Clock Skew [ns]	Number of Buffers	Buffer Area [µm^2]	Dynamic Power[1] [mW]	Static Power [nW]
Standard	0.00	0.150	1029	3632.7	3.95	4.37
	0.25	0.276	610	2060.8	3.66	3.72
	0.50	0.506	474	1517.6	3.55	3.45
	0.75	0.772	444	1419.0	3.53	3.40
	1.00	1.032	344	1249.6	3.52	3.43
	1.50	1.335	322	1193.6	3.52	3.42
	0.00	0.150	1029	3632.7	3.95	4.37
Buffers and inverters	0.00	0.225	1707	4192.4	3.96	4.17
	0.25	0.433	1076	2222.6	3.64	3.44
	0.50	0.549	946	1910.7	3.59	3.37
	0.75	0.780	876	1740.5	3.57	3.33
	1.00	1.020	824	1639.1	3.55	3.32
	1.50	1.512	716	1465.5	3.54	3.33
	2.00	1.684	645	1387.4	3.54	3.35
Logic level balancing	0.00	0.407	618	2116.8	3.88	3.69
	0.25	0.411	618	2111.5	3.87	3.68
	0.50	0.489	618	2054.6	3.84	3.56
	0.75	0.510	618	2039.8	3.83	3.53

[1]At 100% activity.

Appendix C Bluetooth Smart Results

Synthesis results are listed in Table C.2 to C.4. Settings presented in Table C.1 are used unless other settings are specified.

Table C.1: Default synthesis settings.

Setting	Value
Max transition	0.5 ns
Leaf max transition	0.5 ns
Max capacitance	0.6 pF
Max fan-out	2000
Target early delay [ns]	0
Logic level balancing	Disabled
Buffer selection	Buffers only

Table C.2: Synthesis results for the Bluetooth Smart design.

Max Transition [ns]	Leaf Max Transition [ns]	Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm^2]	Dynamic Power[1] [mW]	Static Power [nW]
0.5	0.25	0.00	0.884	8025.1	6.392	131.91
		0.50	0.788	8200.2	6.403	131.95
		1.00	1.121	7884.3	6.366	131.92
		1.50	1.531	7852.6	6.369	131.95
		2.00	2.039	7906.6	6.371	132.50
		3.00	3.043	7787.3	6.343	132.52
		4.00	2.410	7860.7	6.344	132.35
	0.50	0.00a	0.685	6216.4	6.003	129.20
		0.00b	1.026	5038.6	5.807	128.57
		0.25	0.613	5815.9	5.919	129.01
		0.50	0.683	5751.3	5.915	128.98
		0.75	0.841	5700.6	5.924	128.90
		1.00	1.027	4932.2	5.807	128.52
		1.50	1.554	4703.2	5.757	128.40
		2.00	1.940	4391.2	5.741	128.40
		3.00	3.032	4280.1	5.697	128.50
		4.00	2.716	4479.2	5.746	128.39
1.0	0.25	0.00	0.884	8025.1	6.392	131.91
		0.50	0.788	8200.2	6.403	131.95
		1.00	1.121	7884.3	6.366	131.92
		1.50	1.531	7852.6	6.369	131.95
		2.00	2.039	7906.6	6.371	132.50
		3.00	3.043	7787.3	6.343	132.52
		4.00	3.867	7770.5	6.334	132.52
	0.50	0.00	0.726	6095.9	6.005	129.11
		0.50	0.685	6072.7	6.020	129.11

[1]At 100% activity.

Max Transition [ns]	Leaf Max Transition [ns]	Target Skew [ns]	Clock Skew [ns]	Buffer Area [µm^2]	Dynamic Power[1] [mW]	Static Power [nW]
1.0	0.50	1.00	1.054	5316.1	5.903	128.69
		1.50	1.560	4958.0	5.836	128.53
		2.00	1.855	4793.0	5.814	128.44
		3.00	2.765	4571.0	5.798	128.36
		4.00	3.082	4298.3	5.764	128.41
	0.75	0.00	0.726	6095.9	6.005	129.11
		0.50	0.684	6192.6	5.995	129.18
		1.00	1.054	5316.1	5.903	128.69
		1.50	1.493	4989.1	5.812	128.58
		2.00	1.855	4793.0	5.814	128.44
		3.00	2.765	4571.0	5.798	128.36
		4.00	2.716	4479.2	5.746	128.39
	0.10	0.00	0.685	6216.4	6.003	129.20
		0.50	0.685	6072.7	6.020	129.11
		1.00	1.054	5316.1	5.903	128.69
		1.50	1.560	4958.0	5.836	128.53
		2.00	1.855	4793.0	5.814	128.44
		3.00	2.765	4571.0	5.798	128.36
		4.00	2.716	4479.2	5.746	128.39

[1]At 100% activity.

Table C.3: Longest path and hold time cost for the Bluetooth Smart design.

Max Transition [ns]	Leaf Max Transition [ns]	Target Skew[ns]	Clock Skew [ns]	Longest Path [ns]	Hold Time Cost [ns]
0.5	0.5	0.00a	0.685	10.235	1270.58
		0.00b	1.026	7.818	334.13
		0.25	0.613	7.847	425.50
		0.50	0.683	7.893	413.24
		0.75	0.841	10.223	1190.56
		1.00	1.027	7.798	296.66
		1.50	1.554	7.798	230.95
		2.00	1.940	7.700	216.13
		3.00	3.032	7.747	216.73
		4.00	2.716	8.762	333.06

Table C.4: Multi-level module-level clock gating comparison.

Module ICGs	Target Skew[ns]	Clock Skew [ns]	Buffer Area [µm^2]	Dynamic Power[1] [mW]	Static Power [nW]	Scenario Power [mW]
1 Level	0.0a	0.685	6216.4	6.003	129.20	0.2867
	0.5	0.683	5751.3	5.915	128.98	0.2471
	1.0	1.027	4932.2	5.807	128.52	0.2329
	1.5	1.554	4703.2	5.757	128.40	0.2282
	2.0	1.940	4391.2	5.741	128.40	0.2272
	3.0	3.032	4280.1	5.697	128.50	0.2269
	4.0	2.716	4479.2	5.746	128.39	0.2347
2 Levels	0.0	0.792	5409.6	5.903	128.47	0.2540
	0.5	0.782	5254.2	5.900	128.46	0.2430
	1.0	1.063	4994.7	5.841	128.32	0.2352
	1.5	1.547	4625.3	5.788	128.35	0.2272
	2.0	2.024	4478.6	5.775	128.55	0.2264
	3.0	2.790	4261.3	5.715	128.69	0.2256
	4.0	3.289	4233.0	5.702	128.75	0.2269

[1]At 100% activity.

Appendix D Register Delay Calculation

The technology cell library shows the delays from *Clk* to *Q* as listed in Table D.1.

Table D.1: Register delay.

Clk Transition Time [ns]	Clk-Q Delay [ns]
0.04	0.395098
0.20	0.479772
0.88	0.805251
1.80	1.156050
2.40	1.357950
3.80	1.780950
6.00	2.356220
8.00	2.817090
10.00	3.236580

The formula given in Equation (D1) gives a good approximation of the propagation delay. In this formula, T_{Clk-Q} is the propagation delay from *Clk* to *Q*, and $T_{Tran,Clk}$ is the clock signal transition time. This formula has been used to calculate the delays for transition times used in this project, as listed in Table D.2.

$$T_{Clk-Q} = \frac{1}{0.5506\, T_{Tran,Clk}^2 + 1.5218\, T_{Tran,Clk} - 0.6702} \tag{D1}$$

Table D.2: Register delay.

Clk Transition Time [ns]	Clk-Q Delay [ns]
0.10	0.4370
0.25	0.5104
0.50	0.6268
0.75	0.7368
1.00	0.8414
1.50	1.0370