Analysis of Clock Gating Impact on FPGA Behavior

by

Jafar HONARMAND

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF THE MASTER'S DEGREE
WITH THESIS IN ELECTRICAL ENGINEERING
M.A.Sc.

MONTREAL, AUGUST 13, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Claude Thibeault, Thesis Supervisor
Department of Electrical Engineering, École de technologie supérieure

Mr. Dominic Deslandes, President of the Board of Examiners
Department of Electrical Engineering, École de technologie supérieure

Mr. Bora Ung, Member of the jury
Department of Electrical Engineering, École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND PUBLIC

ON AUGUST 3,2021

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGMENT

Firstly, I would like to express my gratitude to my patient and supportive supervisor, professor. Claude Thibeault, who has supported me throughout this research project. I am extremely grateful for his guidance during the running of this project.

Furthermore, I would also like to appreciate my wife which her pure love and support make my path more convenient.

Also, I cannot forget to thank my family for all the unconditional support. They always encourage me throughout my whole life to reach my goals.

To conclude, I want to thank all my friends at the LACIME laboratory, especially Mathieu Gratuze, for his help and support.

Analyse de l'impact du conditionnement du signal d'horloge sur le comportement des circuits intégrés programmables FPGA

Jafar HONARMAND

RÉSUMÉ

Le projet de maîtrise se concentre sur la recherche de l'effet du bruit sur les délais et sur la définition du pire des cas en mode fonctionnel pendant que la méthode du conditionnement du signal d »horloge (clock gating) s'applique au circuit de mesure. Dans la revue de la littérature, des efforts sont faits pour vérifier que cette méthode est l'une des techniques de réduction de la consommation d'énergie les plus pratiques dans la conception de circuits sur FPGA. Cependant, le point qui est pris en compte dans la mesure est que cette méthode peut induire du bruit sur l'alimentation lorsque les autres parties du circuit restent actives. De plus, une seule horloge est utilisée pour éviter les produits d'intermodulation et les fluctuations indésirables. La conception proposée comprend deux lignes à retard différentes. Les deux lignes à retard sont utilisées pour calculer l'effet de bruit avec cette différence, l'une crée un codage unaire et l'autre crée à la fois le codage unaire et des impulsions. Les impulsions sont représentatives de différents retards dans la conception.

Les résultats confirment la présence de deux phases transitoires: quand l'injection du bruit débute et quand elle est arrêtée. Des fluctuations de délais apparaissent pendant ces deux phases transitoires, le début de l'injection ralentissant en moyenne la logique combinatoire et la fin de l'injection l'accélérant.

Mots-clés: FPGA, mode fonctionnel, retard, bruit

Analysis of clock gating impact on FPGA behavior

Jafar HONARMAND

ABSTRACT

This master's project is concentrated on finding the noise effect on delays and defining the worst-case scenario in the functional mode while clock gating is applied to the measurement circuit. In the literature review, efforts are made to verify that clock gating is one of the most practical power consumption reduction technique in FPGA-based design. However, the point that is considered in the measurement is that clock gating could induce noise on VDD with other parts of the design that remain active. To avoid unwanted fluctuations such as intermodulation due to the use of multiple clocks, only one clock is used. The proposed design includes two different delay lines. Both delay lines are used to calculate the noise effect in the design with this difference, one creates a thermometer coded output, and the other creates both thermometer coded output and pulses. The pulses are representative of different delays in the design.

The results confirm the presence of two transient phases: one when the noise is started and the other one when the noise is stopped. Fluctuations in delay appear in both phases. Starting the noise overall slows down the combination logic while stopping the noise makes it faster.

Keywords: FPGA, functional mode, delay, noise

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABREVIATIONS

ADC	Xilinx Analog to Digital Convertor
CE	Clock Enable
CLB	Configurable Logic Block
CMT	Clock Management Tile
CSV	Comma-Separated Values
CUT	Circuit Under Test
DVFS	Dynamic Voltage and Frequency Scaling
FIFO	First In First Out
FPGA	Field Programmable Gate Arrays
FSM	Finite State Machine
IC	Integrated Circuit
ILA	Internal Logic Analyzer
IOB	Input/Output Blocks
IP	Intellectual Property
LED	Light Emitting Diode
LUT	Look-Up Table
MMCM	Mixed-Mode Clock Manager
MSB	Most Significant Bit
PL	Programmable Logic
PLL	Phase-Locked Loop
PS	Processing System

PSN	Power Supply Noise
RTL	Register Transfer Level
TDC	Time to Digital Converter
VCD	Value Change Dump
VDD	Voltage Drain-Drain

INTRODUCTION

Power consumption is one of the significant concerns in the design of microchips. Researchers are aimed to find new solutions and methods to decrease power consumption. Clock gating is a power consumption reduction method that is used in FPGA and ASIC devices. The major power-consuming in electronics products is the systems clock signal, and it is responsible for the transition state of the elements, which typically leads to the switching power consumption (Shanmugasundaram, 2018). Clock gating could be applied to decrease the switching activity on circuit signals (Yeap, 2012). This technique is used to turn off the clock for the blocks that are not used for a period of time in the design. In FPGA, to apply the clock gating, the clock control elements need to be provided by the manufacturer. Different manufacturers such as Xilinx and Altera implement some specific primitives for clock gating. The seven-series Xilinx FPGA provides BUFGCE to produce clock gating. BUFGCE could be used to shut down multiple regions during non-operation. Toggling the enable of the BUFGCE is meaning of stopping entirely dynamic power consumption (Xilinx, 2018). Clock gating is a simple and adequate power consumption reduction method, but the drawback of using clock gating , which includes noise on power (VDD) when enabled or disabled, needs to be considered and measured.

Indeed, noise impacts the delays, and defining this effect will help find the worst-case scenario, which is when the chip is running, and the noise on the VDD may cause an increase on the combinational logic delay. Therefore, two parts needed to be created to find the worst-case scenario: a circuit to measure delays and a noise source to emulate clock gating. Time to Digital Converters (TDCs) are potential delay measurement circuits. Some, like (Narasimman, Prabhakar, & Chandrachoodan, 2015), (Chan & Roberts, 2004) and (Soni, Patel, Panda, & Sarbadhikari, 2017) presents different types of TDCs. Since the TDC structure is an entirely digital, simple, and compact structure that provides flexibility and low-cost design in the FPGA, it could be used to implement a delay line. However, the structure needs to be adapted to allow delay measurements during the transient noise phases caused by enabling or disabling clock gating. Examples of the noise sources allowing clock gating emulation are already

available in the literature (Larche, 2013) and (Gnad, Oboril, Kiamehr, & Tahoori, 2018). They need to be adapted to allow different noise modes.

Thesis objectives

The principal objective of this research project is to study the impact of power noise on delays in the functional mode, induced by clock gating. A particular emphasis is put on the transient noise phases appearing when clock gating is enabled and disabled, under different noise modes and at different clock frequencies. To do that, a new design is proposed and implemented to obtain and compare the results in terms of delays.

Thesis organization

This thesis is organized into three different Chapters as follows:

- Chapter 1 presents the primary material required for this work, such as the Xilinx platform used in the design, description of essential elements and primitives, and design implementation details in Vivado software. Furthermore, the overview of the related work is discussed in this chapter.
- Chapter 2 presents the proposed design, which allows examining the impact of noise on delays at different frequencies. The proposed design aims to use different delay elements for comparison purposes at a wide range of frequencies.
- Chapter 3 presents the results and analysis in the design in terms of delay while the noise affects the system.

A brief conclusion is the last part of this research.

Thesis contribution

To the best of our knowledge, this is the first time measuring delays, for each clock period, during transient phases induced by clock gating for a wide range of clock frequencies, is considered. The results for different noise modes clearly show the impact of clock gating.

CHAPTER 1

LITERATURE REVIEW AND FPGA ARCHITECTURE

1.1 Literature review

This research aims to demonstrate the behavior of chips at different frequencies when clock gating affects this behavior. Moreover, this work used the Xilinx Zynq-7000 SoC FPGA, based on a 28 nm technology. Notably, computing the noise impact on the delays and defining the worst-case scenario in the functional mode, namely when the chip is running, are the topics of interest in this research. The worst-case scenario will be when the noise on the VDD causes an increase on the combinational logic delay. The literature review section is divided into three parts that discuss the clock gating as a power consumption reduction technique, the comparison of different delay lines, the noise source, and existing delay measurement platforms.

1.1.1 Clock gating

Clock gating is a well-known power optimization approach used in FPGA devices (as well as in ASICs) to reduce additional switching activity. Clock gating is an easy and efficient method to decrease the dynamic power consumption by turning off the non-used parts in the design and go to the standby state (Pedram & Rabaey, 2002). By applying the clock gating technique, we could decrease the power by reducing the switching activity in the FPGA elements such as flip-flops, gates, and clock trees (Y. Zhang, Roivainen, & Mammela, 2006). It is required to use the appropriate clock control components prepared by the manufacturer to apply the clock gating inside an FPGA. In Altera devices, it is possible to generate a global clock with enabling port applying the mega function ALTCLKCRTL (Altera 2018). The other manufacturers like Xilinx have also proposed solutions for clock gating in their products, such as Intelligent clock gating (Xilinx 2012). Intelligent clock gating is used to decrease the dynamic power

consumption on parts of the design. As this technique is based on controlling the CE of the slices, deactivating many slices would be impossible and will increase the power consumption. The idea of using Xilinx primitives comes from (Oliver, Curto, Bouvier, Ramos, & Boemo, 2012), where it uses only the available clock control signal due to additional skew. This clock gating method uses the BUFGCE primitive described before. One of the main disadvantages of clock gating is that it can induce noise on VDD with other parts of the design that remain active. This noise can in turn affect delays in the related combinational logic. This is the impact we aim to measure in this project.

1.1.2 Comparison of different delay lines

Several methods for measuring the delay have been developed. Measurements of the interest path delays inside the specific FPGAs using two ring oscillators and analyzing their operating frequencies are presented in (Ruffoni & Bogliolo, 2002). An on-chip path delay measurement is introduced in (Pei, Li, & Li, 2009) to measure the delay in the circuit for efficiently detecting and debugging delay faults in the fabricated integrated circuits. Some other techniques are suggested in (Katoh, Tanabe, Zahidul, Namba, & Ito, 2009), (Raychowdhury, Ghosh, & Roy, 2005), (Matsumoto, 2005), and (Wong, Sedcole, & Cheung, 2008), either measure the speed of combinational circuit paths or even complete sequential circuits (Wong & Cheung, 2011). A Time to Digital Converter (TDC) changes a time interval among different pulses into digital numbers. A TDC structure is an entirely digital, simple, and compact structure that provides flexibility and low-cost design in the FPGA, and TDC is a common way to design a delay line (Soni et al., 2017). Figure1.1 and 1.2 illustrate the principle of the TDC delay line (also known as tapped delay line) and its timing diagram. Buffers are the delay elements in this delay line (Yao, 2011). The start signal is going through the design by delay elements and is captured with the arrival of the rising edge of the stop signal. The sampling procedure generates by flip-flops shows the position of the delay line elements when the stop signal happens. The output of the flip-flops is a thermometer code that shows one if the start signal reaches the stop signal rising edge and zero if the start signal does not reach the stop signal rising edge.

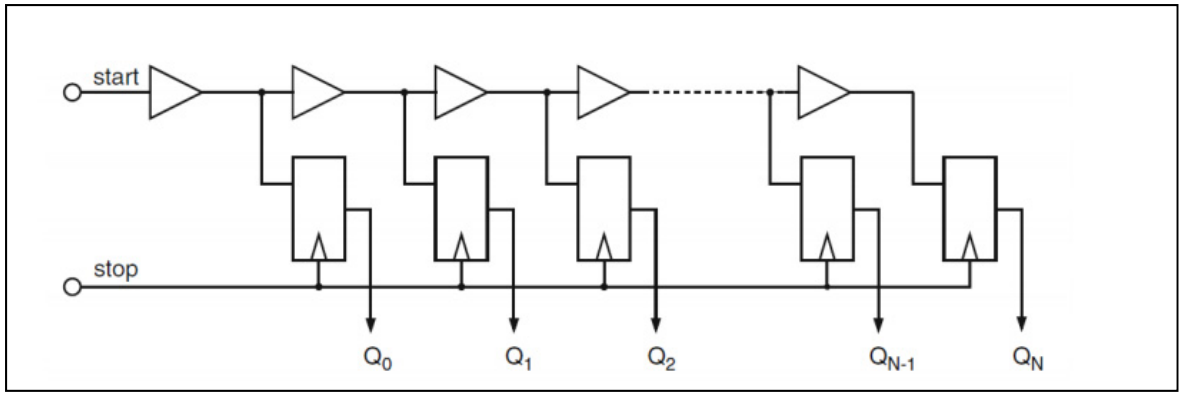

Figure 1.1 Principle of the TDC delay line
Taken from Yao (2011)

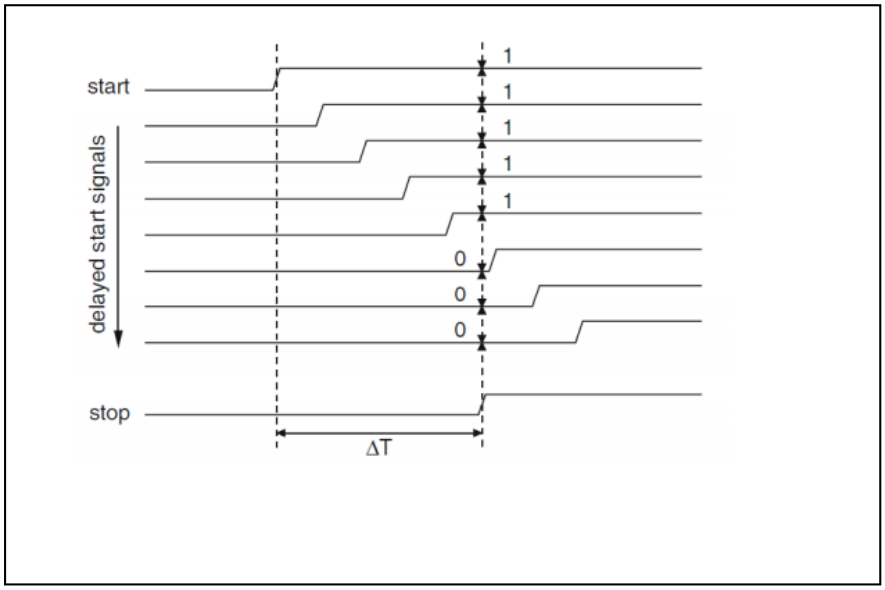

Figure 1.2 TDC delay line timing diagram
Taken from Yao (2011)

The TDC delay line also could be implemented with CARRY4 as delay units (M. Zhang, Wang, & Liu, 2020). The advantages of using a carry chain are high accuracy and resolution. Each CARRY4 block consists of four MUXCY elements and could be used as a fast delay element in the design without adding the net delay for each element. The other feature in the

design of the CARRY4 is when the first element is placed, the position of another will automatically determine by the tool. The CARRY4 delay line sampling model is the same as buffer delay line TDC, and the data are captured in the stop signal rising edge. The delay of each CARRY4 is between 60 to 70 ps based on the device and its speed (Torres et al., 2013). Figure 1.3 presents the Xilinx CARRY4 block which the MUXCYs are used as a delay element to improve the resolution. Note that using MUXCY as a delay element will increase the number of components in the delay line and might cause the metastability problem in the results, and we need to post-processed them. Therefore, as a first try in this research, we use CARRY4 logics as delay elements. To the best of our knowledge, the novelties of this project come from the capabilities of our experimental setup to measure delays during transient phases induced by clock gating for a rather wide range of clocks frequencies. We use two different delay lines. Both delay lines are based on the general structure shown in Figure 1.2. One uses buffers as delay elements, and the other one uses the CARRY4 block. The number of delay elements need to be determined in this design. The flexibility with respect to the clock frequency is provided by the adjustable buffer chain of delay elements prior to the delay lines. This part allows having more accurate results compared with the clock rising edge at each experiment.

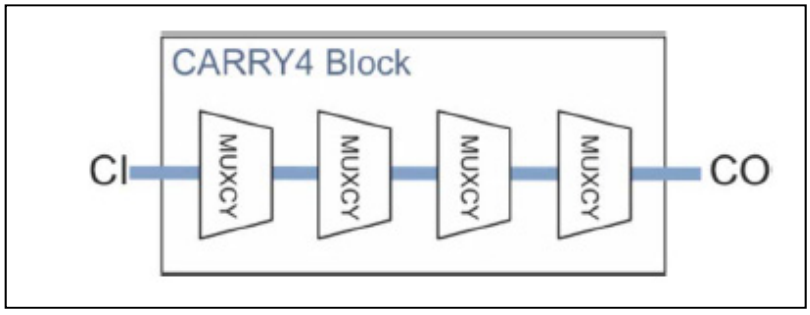

Figure 1.3 Xilinx CARRY4 block
Taken from Torres et al (2013)

1.1.3 The noise source model

As mentioned before, using clock gating as a power consumption technique will cause transient phases in the circuit power distribution. Any stop and starts of the clock gating will induce switching activity and need to be considered.

Using an appropriate noise source in the design is the essential part to get reliable results. The noise source presented in (Larche, 2013) contains three shift registers. Each one has 1200 flip-flops to which the different clock is applied. The first flip-flop is a toggle flip-flop, and its output going into the 1199 remaining flip-flops. There are four possibilities to generate the noise, which 0, 1200, 2400, and 3600 flip-flops could be activated each time to create the required noise in the design. As shown in Figure 1.4, the noise source block diagram is working with different frequencies, and it would add the intermodulation product effects to the design. Intermodulation distortion occurs in nonlinear devices and produces unwanted additional signals, resulting from the interaction of two or more signals (Thibeault & Gagnon, 2018). Appling the BUFGCE clock gating technique will use only one clock as a clock reference to avoid the intermodulation product effects in the design. Another issue that is not considered in this design is that 1200 clock periods are needed to feed all the flip-flops in each chain. If the clocks do not feed all the flip-flops, then the maximum noise would not be added to the design.

Also, applying the noise source by using a line of toggling flip-flop is used in (Gnad et al., 2018). In this case switching activity could be control simultaneously from the control part. Note that this work is used a fixed 100 MHz frequency and the changes to the clock frequency did not take into consideration which is the one of the design's goal. Based on that, this design must be modified to get the maximum noise more instantly by finding the required number of toggling flip-flops.

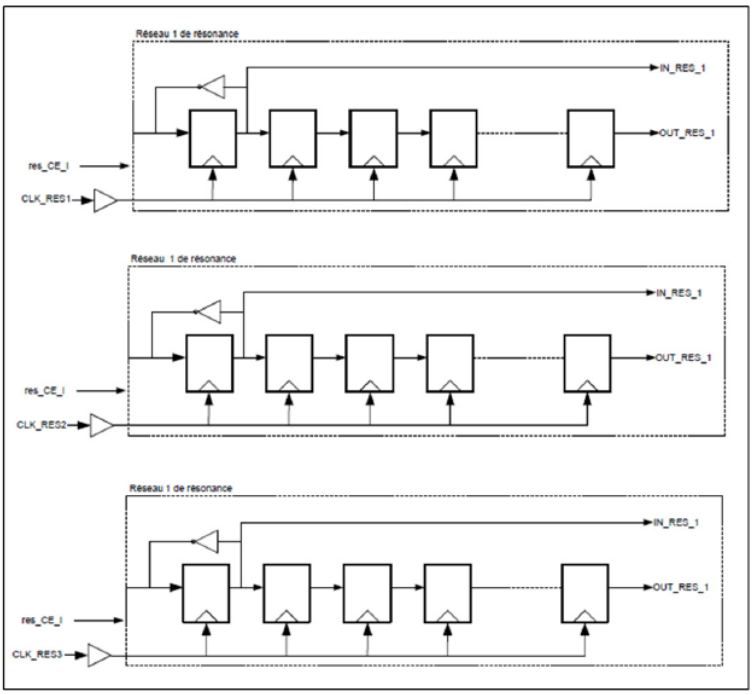

Figure 1.4 Proposed noise generator block
Taken from Larche (2013)

1.1.4 Delay measurement platforms

In addition to the specific items discussed before (delay lines, noise sources), one also must pay attention to the overall functionalities of exiting delay platforms. There exists different platforms using different techniques to measure the delay, such as ring oscillator delay measurement, time to voltage delay measurement (Wong, 2011), and pulse width based measurement (Larche, 2013). In this thesis, we are mostly interested by the last two techniques, when used to measure the impact of noise on delays in an FPGA. From that perspective, the closest two existing platforms are the ones presented in (Larche, 2013) and (Gnad et al., 2018).

The platform developed by (Larche, 2013) was designed to perform delay measurement in steady state conditions to reveal to impact of intermodulation noise caused by two different clock domains. From (Larche, 2013) we reused and adapted the proposed pulse width based

measurement technique to measure the delay margin. Figure 1.5 shows the delay measurement part of their design, including two flip-flops, one AND gate, one NOT gate, and one XOR gate. Combined with the inverter, the first (leftmost) flip-flop is of toggling type and is launching a transition when enabled by the CE input signal. The launched transition travels along the delay line, which is made of a simple AND type logic gate, and is captured by the rightmost flip-flop. The XOR gate allows measuring the delay margin pulse whose width is a good estimate of the difference between the delay of the end of the line and the arrival of the next rising edge of the clock. Note that this signal is measured on a pin of the FPGA, which requires the pulse width to be large enough not to be filtered out. In addition to what was mentioned earlier about the noise source, the main differences with our platform is the fact that: 1) we are interested measuring the impact of noise on delays in the transient phases, and 2) we measure pulse widths inside the FPGA.

The platform developed by (Gnad et al., 2018) was also designed to perform delay measurement in steady state conditions In addition to what was mentioned earlier about the noise source, the main differences with our platform is the fact that: 1) we measure the impact of noise on delays in the transient phases, and 2) we support an addition noise mode, where transitions are sent along the clock distribution network but no flip-flop is toggling.

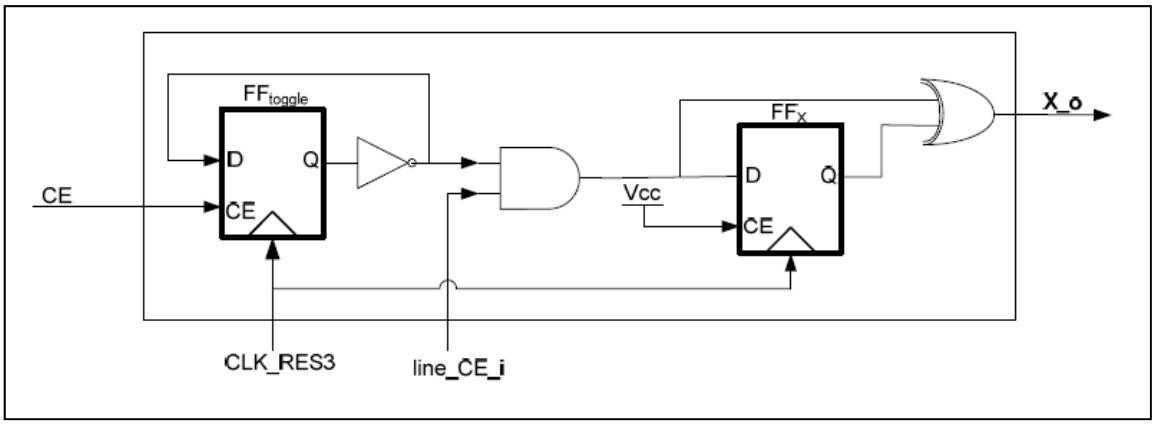

Figure 1.5 Delay measurement block taken from Larche (2013)

1.2 FPGA introduction

This chapter introduces the primary material required for a better understanding of this work. Initially, the platform and the Xilinx Field Programmable Gate Array (FPGA) logic circuit used in this design are discussed in detail. Furthermore, some essential elements will be described as follows: Clock Management Tiles (CMTs), BUFGCE primitive, Xilinx Analog to Digital Converter (XADC) block, Intellectual Property (IP) core design components like Internal Logic Analyzer (ILA), Virtual Input/Output (VIO) and First-In-First-Out (FIFO) generator. Afterward, the design implementation in Vivado software will be addressed. Finally, an overview of the related work will be discussed.

1.3 Platform and the FPGA used in the design

The Xilinx ZC702 evaluation board is used in our design. The board's physical appearance and block diagram are shown in Figure 1.6 and Figure 1.7.

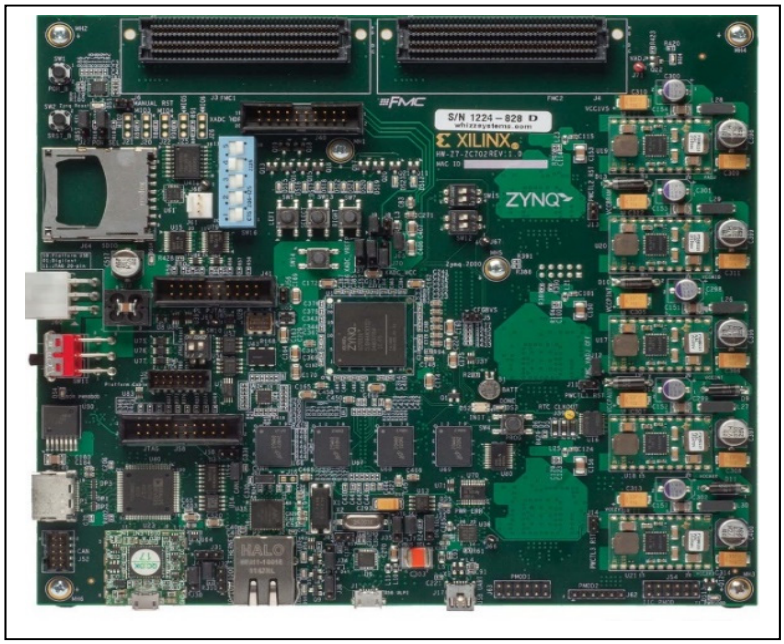

Figure 1.6 ZC702 physical appearance
Taken from Xilinx (2019)

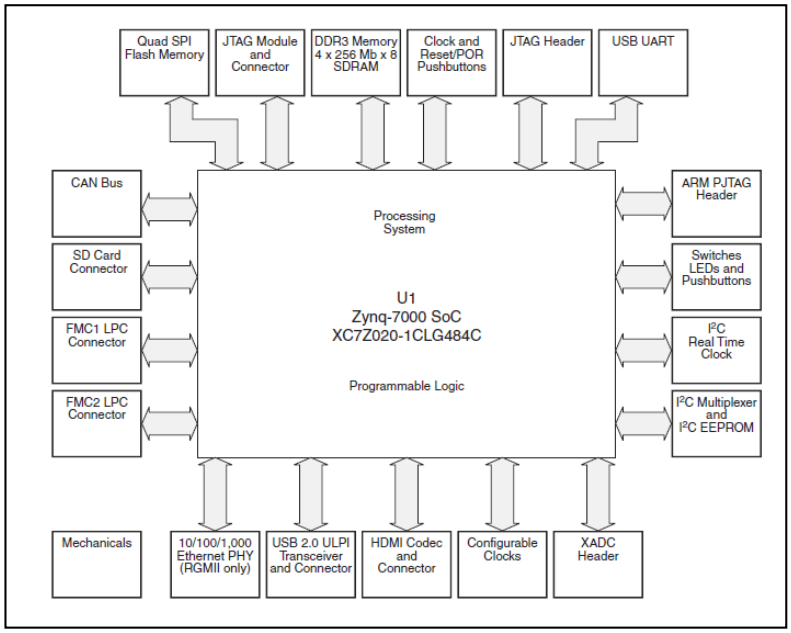

Figure 1.7 ZC702 block diagram
Taken from Xilinx (2019)

The ZC702 board features are listed as follows:

- Compatible with all versions of Vivado;
- 1 GB DDR3 component memory;
- 128 Mb flash memory;
- USB JTAG interface;
- Clock source:
    - Fixed differential 200 MHz LVDS oscillator;
- User I/Os:
    - Two programmable user pushbuttons;
    - Eight user LEDs;
    - Dual row Pmod;
    - Single row Pmod;
- Power on/off switch;
- Xilinx Zynq XC7Z020-1CLG484C device;

- Internal XADC;
- 53200 slice LUTs;
- 106400 slice registers;
- 13300 slice;
- 53200 LUT as logic;
- 17400 LUT as memory;
- 53200 LUT flip-flop pairs;
- 140 blocks RAM;
- 200 bounded IOB;
- 192 IBUFDS;
- 32 BUFGCTRL;
- Up to 4 clock generator.

The Xilinx Zynq-7000 XC7Z020-1CLG484C SoC FPGA is used in the ZC702 board includes both a Processing System (PS) and 28 nm Programmable Logic (PL) on a single device. The Zynq-7000 SoC block diagram is shown in Figure 1.8.

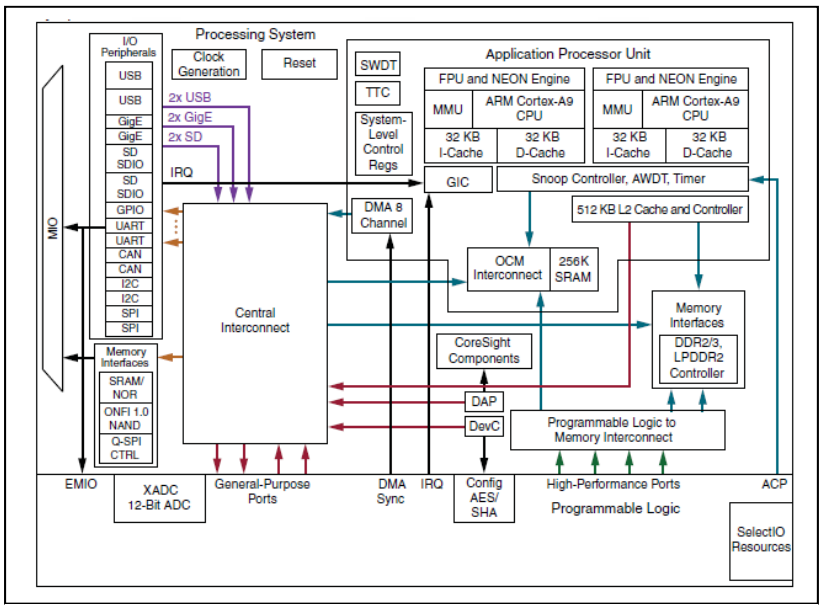

Figure 1.8 Zynq-7000 SoC block diagram
Taken from Xilinx (2018)

1.4 FPGA logic circuit

An FPGA is a semiconductor Integrated Circuit (IC) device produced to be configured by a user, and its main components are the Configurable Logic Blocks (CLBs), the programmable interconnects, the Input/Output Blocks (IOBs), and the CMTs. The number and the position of the elements are unique for each device.

1.4.1 CLBs

The CLBs are the essential part of any circuits' design, including two slices, one SLICEM, one SLICEL, or two SLICEL. Figure 1.9 presents the CLB details in Zynq-7000 with SLICEM (Slice0) and SLICEL (Slice1). About 67% of the slices are SLICEL, and the balance is SLICEM. The difference between SLICEL and SLICEM is the latter supports two additional functions: storing data using distributed RAM and shifting data with 32-bit registers .The slices are not connected directly together. Both slices connect to the switch matrix to reach the general routing.

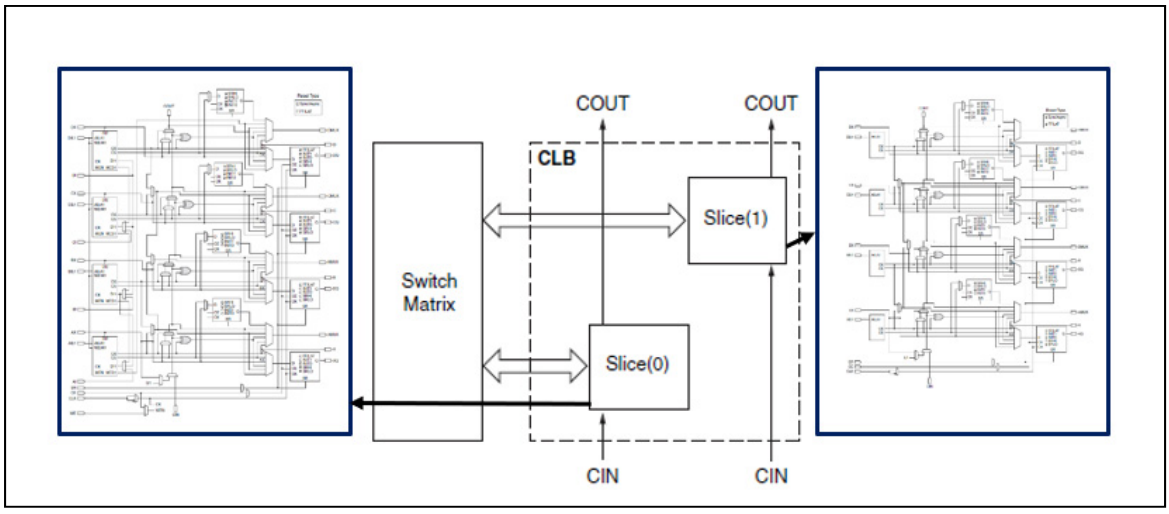

Figure 1.9 Zynq-7000 CLB details with SLICEM and SLICEL
Taken from Xilinx (2016)

Each slice includes the following parts:

- **Look-Up Table (LUT):** Four 6-input LUTs in each slice are used as function generators. Each LUT has six inputs (A1 to A6) and two outputs (O5 and O6).
- **Storage components:** There are eight storage components through a slice. Four can be used as a D flip-flop or latch, and the others only can be used as a D flip-flop. All the essential signals like a clock, CE, set, and reset are connected to all the storage components.
- **Multiplexers:** Are used to combine LUTs into different functions.
- **Carry logic:** There are four carry multiplexers (MUXCY) and four XOR gates in each slice. Carry logic provides fast operation in each slice. In this design, carry logic is used as a fast delay element because the connection between them follows a specific routing that does not add the net delay to each carry logic's design. As shown in Figure 1.10, the signal is entering from the CI input of each carry logic and going out from the CO3 output.

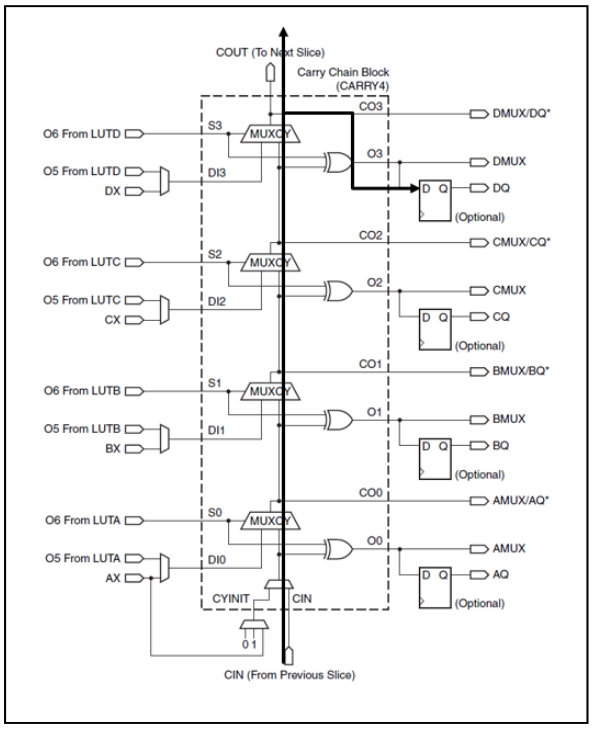

Figure 1.10 Carry logic connection in a slice
Taken from Xilinx (2016)

1.4.2 Programmable interconnects

The Programmable interconnects provide the routing path between different FPGA parts, such as CLBs and IOBs.These vertical and horizontal connections are connected to several switch matrices to reach the other resources. Figure 1.11 shows the example of FPGA programmable interconnects.

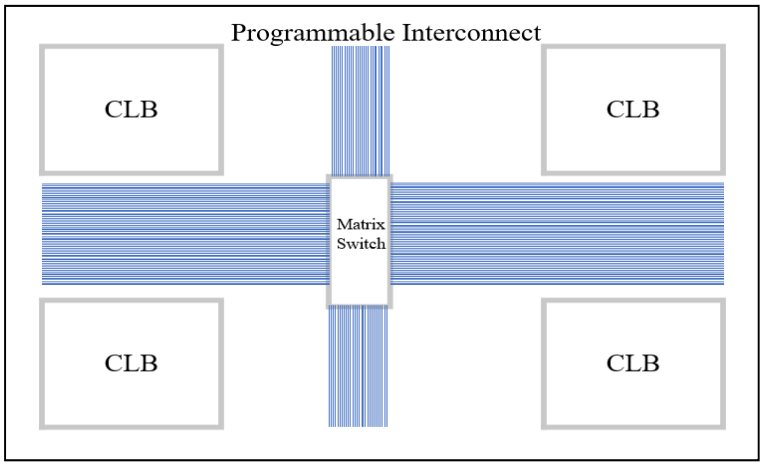

Figure 1.11 Example of FPGA programmable interconnects taken from Xilinx (2016)

1.4.3 IOBs

An IOB connects the inside FPGA architecture to the exterior design through interfacing pins. In Xilinx FPGAs, each I/O group is called a bank, and as shown in Figure 1.12, Zynq-7000 XC7Z020 has four I/O banks consisting of 50 IOBs. The HR I/O banks are produced to work with voltages higher than 3.3V.

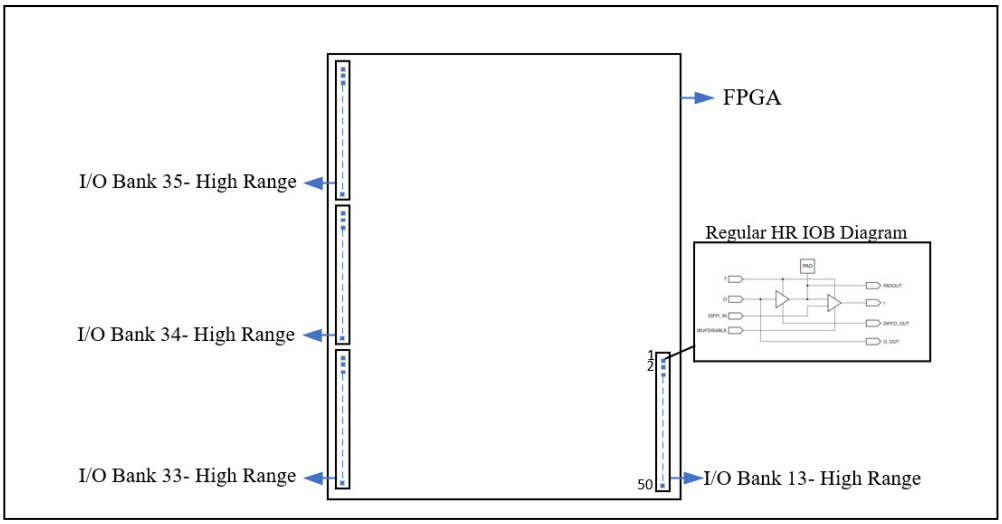

Figure 1.12 Zynq-7000 XC7Z020 I/O banks taken from Xilinx (2018)

1.4.4 Clock management tiles

The CMTs are used to create the required clocks in the design. Each CMT includes one Mixed-Mode Clock Manager (MMCM) and one Phase-Locked Loop (PLL). In our design, we need to generate eight different phases of 450 MHz, the fast clock for the pulse width measurement. To do that, two PLLs are used, the first one generating 0,90,180,270 degrees, and the second one generating 45,135,225,315 degrees of the 450 MHz clock. Furthermore, the following three clocks need to be generated: 1) the design reference clock to create the synchronous pulses, 2) a copy of that clock to control the noise block and 3) a clock twice the reference clock frequency for the measurement part. Two MMCMs are used to create the specified three clocks. Figure 1.13 presents the MMCM and PLL primitives.

The system clock, a 200 MHz differential clock, is used as an input clock in our work. This clock is connected to package pins D18 and C19, respectively, for the positive and negative parts in the FPGA. Since each clock needs to connect to the buffer after going inside the FPGA, an IBUFDS is applied to change the input clock to the single-ended clock.

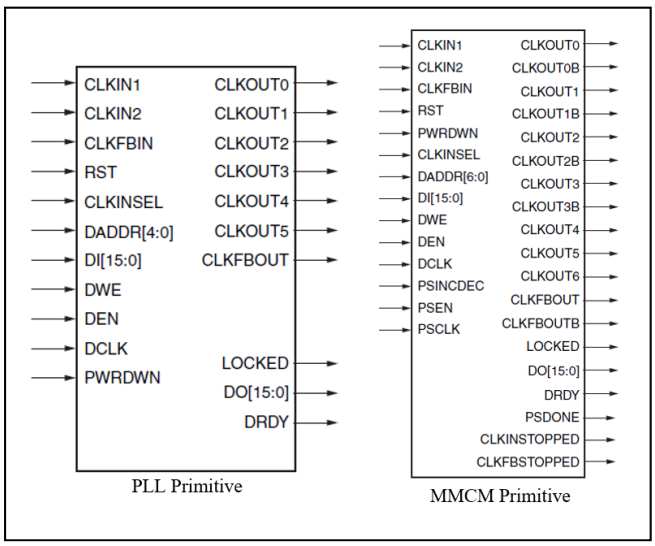

Figure 1.13 MMCM and PLL primitives
Taken from Xilinx (2018)

The BUFGCTRL is the primitive from which all different buffer primitives are created. The first one derived from it for our design is BUFG, which is a simple one-input one-output global clock buffer. Connecting the unrequired pins of BUFGCTRL to VDD or ground will create a BUFG. Figure 1.14 shows the relation between BUFG and BUFGCTRL.

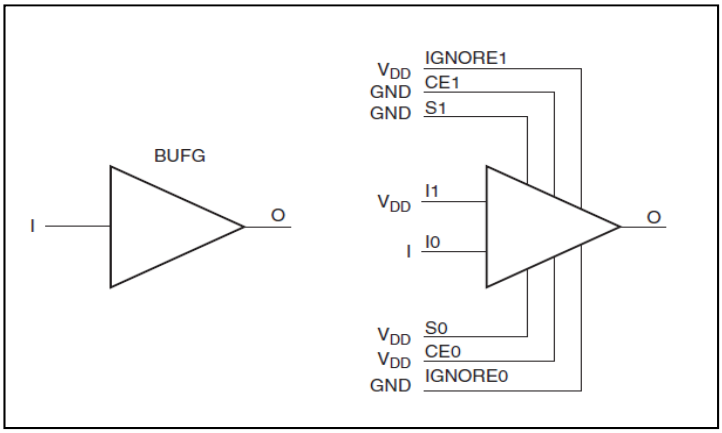

Figure 1.14 Relation between BUFG and BUFGCTRL
Taken from Xilinx (2018)

1.5 BUFGCE primitive

A second clock buffer primitive derived from BUFGCTRL for our design is BUFGCE, which is used to perform clock gating. Clock gating is the way to shut down the design's non-operating parts, while there is no need to use them. As shown in Figure 1.15, BUFGCE has three pins: input clock, output clock, and Clock Enable (CE). In case CE is zero, the BUFGCE output is zero, meaning that the clock would be deactivated. When CE changes to one, the clock would be activated. We use two BUFGCE to control the noise generator in the design. Controlling these BUFGCEs will let us active the noise when we need that, and in other circumstances, even while the rest of the design is working, this part does not consume dynamic power.

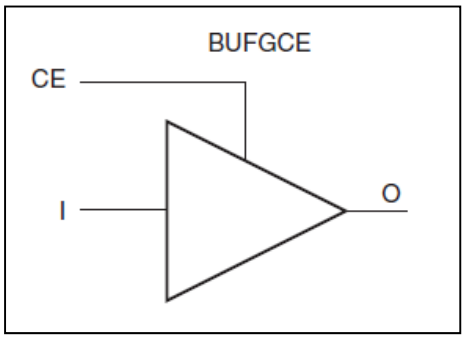

Figure 1.15 BUFGCE primitive
Taken from Xilinx (2018)

1.6 XADC block

The XADC block provides temperature sampling in Zynq-7000 SoC devices. The sampling operations are performed at the speed of one Mega sample per second by internal sensors. Figure 1.16 shows the XADC primitive and its window in Vivado software. Using XADC in our design allows measuring the temperature while doing the experiments to monitor if there is a significant impact on the results.

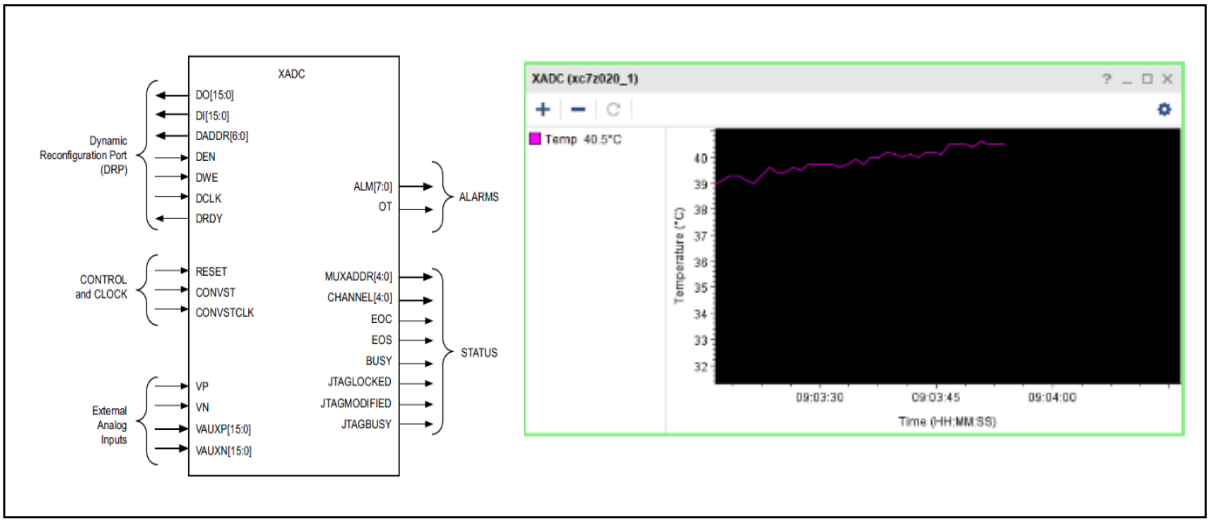

Figure 1.16 XADC primitive and its window in Vivado
Taken from Xilinx (2018)

1.7 IP core design components

The Intellectual Property (IP) core design is a technique in which the FPGA designer could use pre-designed blocks with specific capabilities. Xilinx has produced a wide range of these IPs, which are used in various types of FPGAs. In our work, several IP core components are used, which are discussed in the following subsections

.

1.7.1 ILA

The Xilinx logic core ILA is an FPGA logic analyzer that monitors any required internal signal in a design. Trigger options, data storage, and monitoring the signals' waveform are characteristics of this logic core. Figure 1.17 shows the ILA logic core symbol. The ILA input clock is synchronous to the design and provides sampling at each clock reference period. To monitor the required signals, we need to connect these signals to the ILA probes. The number of probes could be selected up to 1024, and for each probe, the width can be selected up to 4096 bits. The user could arrange the trigger place for any probes, identifying rising or falling edges.

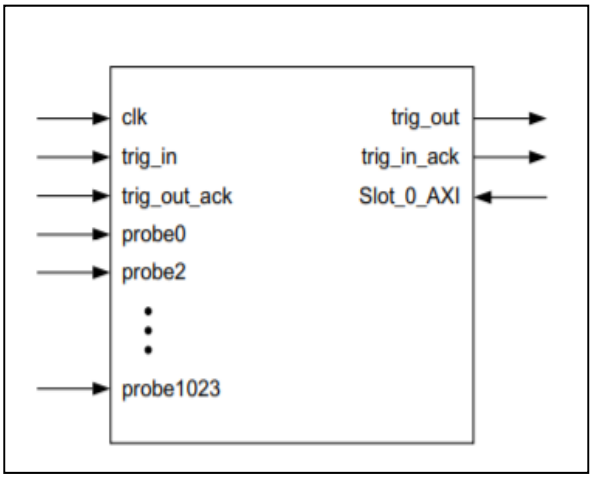

Figure 1.17 ILA logic core symbol
Taken from Xilinx (2016)

In this project, as shown in Figure 1.18 the ILA includes eight probes as below:

- **Probe0[0:0]:** A 1-bit probe to monitor the output of one counter, allowing to indirectly measure the design reference clock frequency, which is too fast to be read directly by the ILA.
- **Probe1[0:0]:** A 1-bit probe to monitor the *Ce_toggling1* signal, which is the CE signal enabling applying noise (namely, enabling toggle flip-flops and their related clock signals). This probe is used as the first trigger in the design. The ILA is set to detect the rising edge for this probe.
- **Probe2[0:0]:** A 1-bit probe to monitor the *Ce_toggling2* signal, which is the CE signal enabling the measurement part. This probe is used as the second trigger in the design. The ILA is set to detect the rising edge for this probe.
- **Probe3[7:0]:** A 8-bit probe used to monitor the first generated delay in the design, which is Sum_start_b4. This delay is designed to be less than one clock reference period.
- **Probe4[7:0]:** A 8-bit probe used to monitor the second generated delay in the design, which is Sum_delay_margin. This delay allows to estimate the delay margin between a given point in a delay line (namely the b4 signal) and the next rising edge of the clock reference.

- **Probe5[7:0]:** A 8-bit probe used to monitor the third generated delay in the design, which is Sum_delay-line. This delay is designed to be greater than one clock reference period.
- **Probe6[7:0]:** A 8-bit probe used to monitor values (called q0 to q7) captured by a register , along the first delay line in the design, namely the buffer delay line. These values, which are thermometer coded, show the progression of a transition along the delay line.
- **Probe7[47:0]:** A 48-bit probe used to monitor another set of values (called Q_carry4), captured by another register along the second delay line in the design, namely the carry4 delay line. These values, which are also thermometer coded, show the progression of a transition along the second delay line.

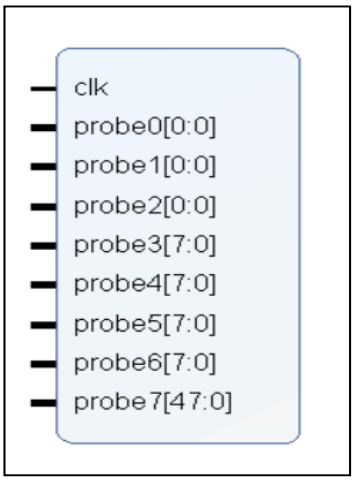

Figure 1.18 Design's ILA details

The probes' signals are captured at the clock reference speed, according to the probes and triggers settings. Also, in this step, the samples are saved in Block Random Access Memories (BRAMs). BRAMs are used to store data on a large scale inside the FPGA. In our design, the number of BRAMs used by the ILA to sort the data is 2.5. As shown in Figure 1.19, an auto-instantiated debug core hub is used as a link to transfer the ILA captured data to the PC. This section connects the JTAG chain within the BSCANE2 component to the ILA within the design.

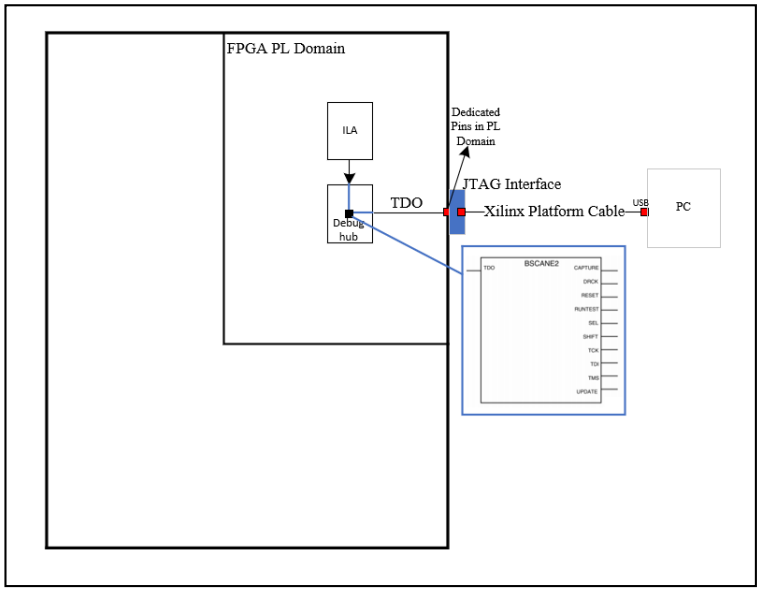

Figure 1.19 ILA to PC connections
Taken from Xilinx (2016,2021)

When the design implementation is finished, and the device is programmed, the data appears in Vivado ILA software's waveform window. Another ILA capability is to export the data in three formats, namely Native (as an ASCII), Comma-Separated Values (CSV), or Value Change Dump (VCD) files. This work's preferred format is CSV to export the waveform window data because post-processing is required. Export the data to a CSV gives the data in binary formatted, and because the delay values in the design are decimal, the radix needs to be changed.

Figure 1.20 presents the complete ILA waveform window in the design. As we can observe, there are two transient phases in the design: at the beginning and end of the *Ce_toggling1*. The *Ce_toggling1* is set to be one microsecond to reach the steady-state condition in the design. Capturing the q0 to q7 and Q_carry4 values are started right after the first transient and continue after we stop the noise to monitor the second transient effect. The *Ce_toggling2* is shown the entire measurement time (1.25 microseconds) in the design, and when it is finished, the delay values are ready to be read.

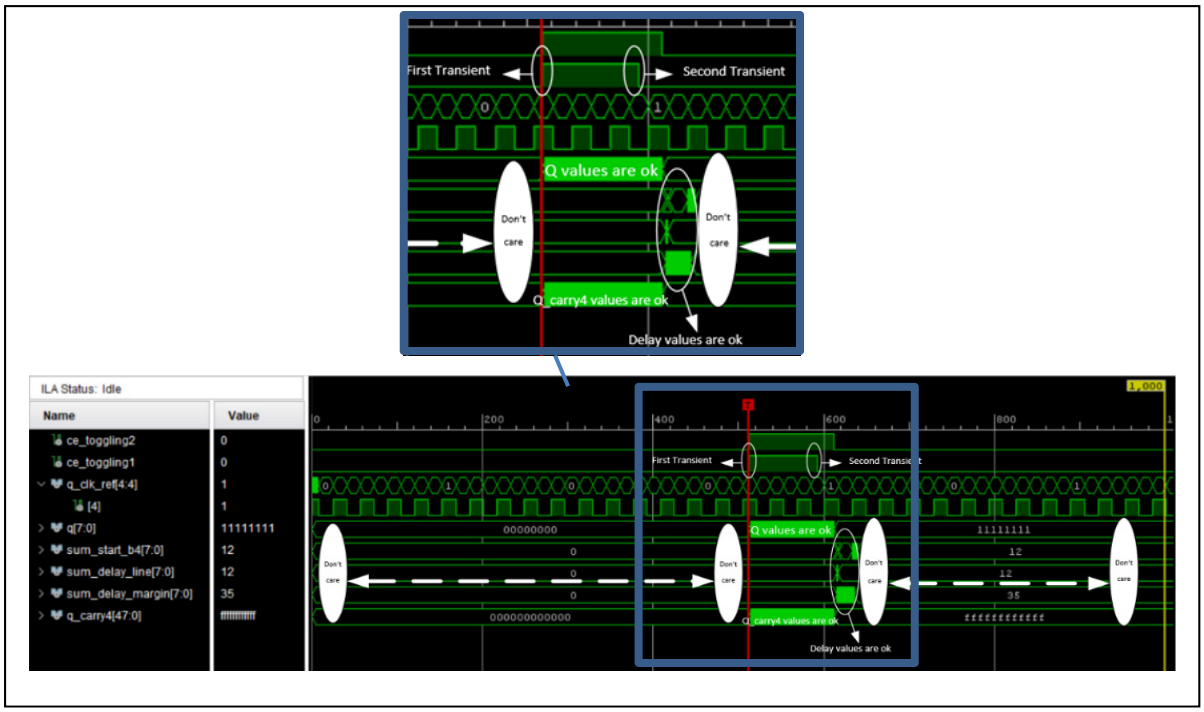

Figure 1.20 Complete ILA waveform window

1.7.2 VIO

The Xilinx logic core VIO is used to monitor and control the inner FPGA signals. Up to 256 input and output probes could be selected. Also, each probe width could be set to up to 256 bits. The VIO input clock is synchronous to the design. Also, it is possible to connect the VIO probes to the LEDs. Figure 1.21 shows the VIO block diagram, which contains input and output registers and the connections to the JTAG interface, which follows the same pattern as the ILA through the debug hub.

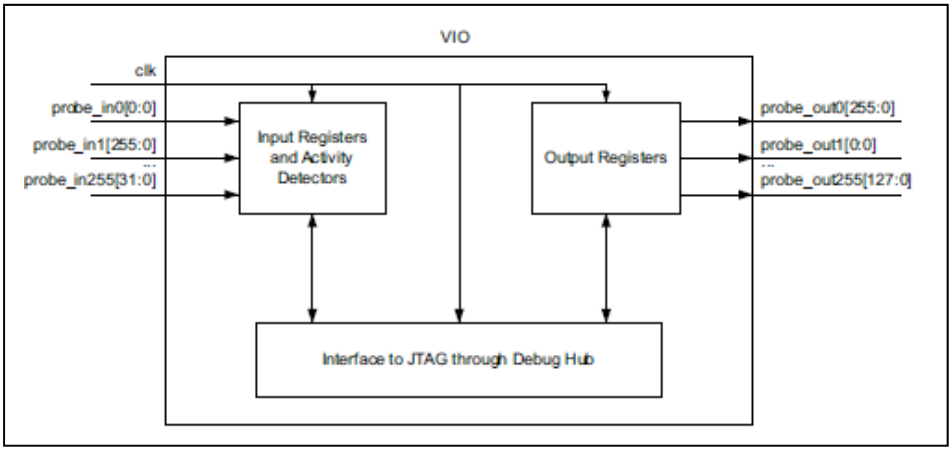

Figure 1.21 VIO block diagram
Taken from Xilinx (2018)

As shown in Figure 1.22, in this project, the VIO uses seven output probes as below:

- **Probe_out0[5:0]:** A 6-bit probe used to select the number of delay elements in a block called the buffer chain, which corresponds to the first (upstream) part of the first delay line(buffer delay line). In this buffer chain, the delay elements are buffers (LUT1), and it could be increased up to 64 in the design. Furthermore, this 6-bit probe is connected to LEDs to monitor the buffer chain delay element's number. The used LEDs are DS17, DS18, DS22, DS21, DS20, and DS19. The LEDs output pins are W10, V7, W5, W17, D15, E15.
- **Probe_out1[0:0]:** A 1-bit probe used to control a first part of the noise generator block in the design.
- **Probe_out2[0:0]:** A 1-bit probe used to control a second part of the noise generator block in the design.
- **Probe_out3[5:0]:** A 6-bit probe used to select the number of delay elements in a block called the carry4 chain, which corresponds to the first (upstream) part of the second delay line (carry4 delay line). In this carry4 chain, the delay elements are carry4 logics, and it could be increased up to 36 in the design.
- **Probe_out4[0:0]:** A 1-bit probe used to control a third part of the noise generator block in the design.

- **Probe_out5[0:0]:** A 1-bit probe used to control a fourth part of the noise generator block in the design.
- **Probe_out6[0:0]:** A 1-bit probe used to select between even and odd periods during the measurements.

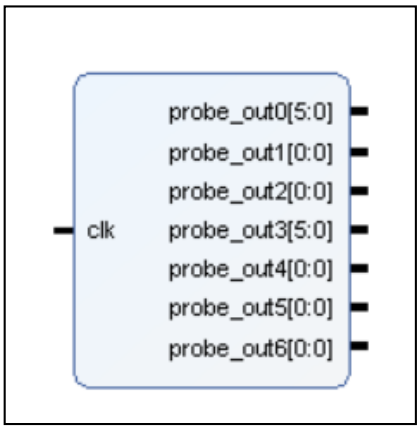

Figure 1.22 Design's VIO details

1.7.3 FIFO generator

The Xilinx logic core FIFO generator is used to produce a memory to store data that can be for example accessed from outside the FPGA. As shown in Figure 1.23, which presents the native interface FIFOs signal diagram, a FIFO includes reading and writing parts, storage, and some logics. A synchronous FIFO, which is used in our design, is a FIFO where the same clock is used to write and read. Different methods are used to implement the native FIFO, like common clock BRAM, independent clocks BRAM, and independent clocks distributed RAM. In this design, the independent clocks distributed RAM scheme is used to implement the FIFO, which is based on LUTs. Write and read clocks in our design are the same, and both are connected to the reference clock. The data are written in the FIFO with the next writing clock rising edge when the write enable is high. The data are read from the FIFO with the next reading clock

rising edge when the write enable is high. The empty level is changed when the reading part is finished, and it is synchronous with the read clock.

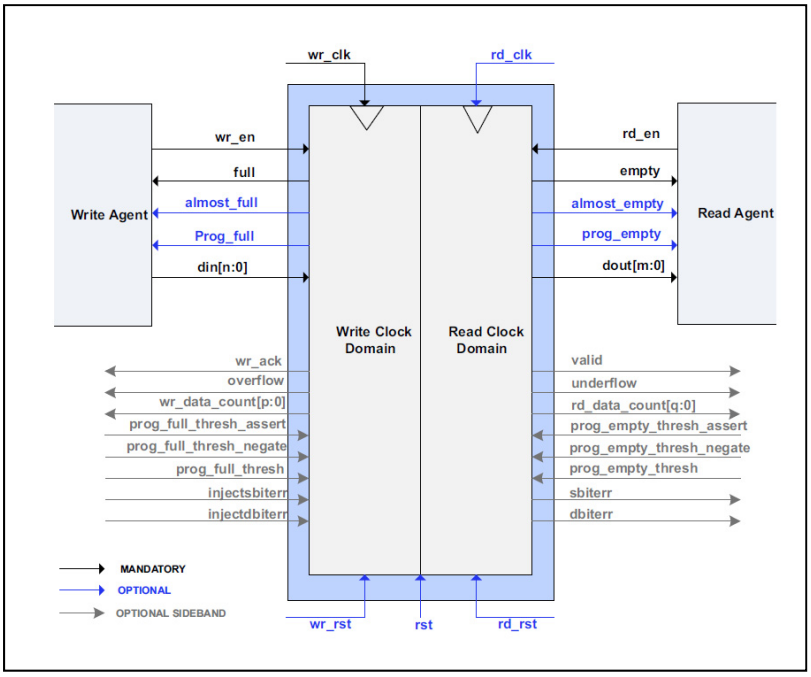

Figure 1.23 Native interface FIFOs signal diagram
Taken from Xilinx (2017)

1.8 Design implementation in Vivado software

The Vivado software is used to support the Xilinx ZYNQ-7000, Ultra-Scale, and all more recent FPGAs families. Vivado supports both VHDL and Verilog source files. In this project, we used Verilog. As part of a Register Transfer Level (RTL) design flow, Vivado presents a new feature that used IPs in the design. The different IPs are located in the IP integrator environment. The Xilinx FPGA design steps in Vivado are as following:

- **Design entry:** This step is started by creating a new project and defining inputs and outputs. The hardware description language (Verilog), the required IP sources, and logical and physical constraints need to be chosen at this step. After finishing the design entry, the behavioral simulation could be done, which shows the design's inputs and output

waveform. This simulation does not consist of delay details and ensures that the design code is working correctly.

- **Synthesis:** This step is used to transform an RTL-specified design into a gate-level description that is mapped into the available FPGA resources. Vivado is used in a well-optimized way to decrease the number of components in the design. To avoid replacing any module in the design with the optimization tool before this step, we need to set the attribute called (* DONT_TOUCH = "yes" *). Setting this attribute to “yes” for a module will prevent any changes during the synthesis process.
- **Implementation:** Vivado's implementation involves the essential steps of placing and routing the design and its related mapped FPGA resources. The placement is corresponded to selecting the physical location of the mapped FPGA resources, and the routing is the operation by which the connections among the blocks are created using the available FPGA interconnects. Post simulation with all the delays and static timing analysis, which shows the worst-case scenario delays for all the design elements, are accessible after this step.
- **Bitstream and debug probes files generation:** This step is used to convert the implementation design details to a bitstream, which is downloaded in the FPGA to implement the design. In our case, it includes both ILA and VIO that are used to probe and control the design. The use of the ILA and VIO cores trigs the generation of a debug probes file, which includes information about the probes used by the ILA and VIO in the design.

Figure 1.24 and 1.25 show the FPGA internal view before, and after implementation in Vivado. The pink color shows the delay block placement, and the yellow color shows the measurement part placement of the design in Figure 1.25.

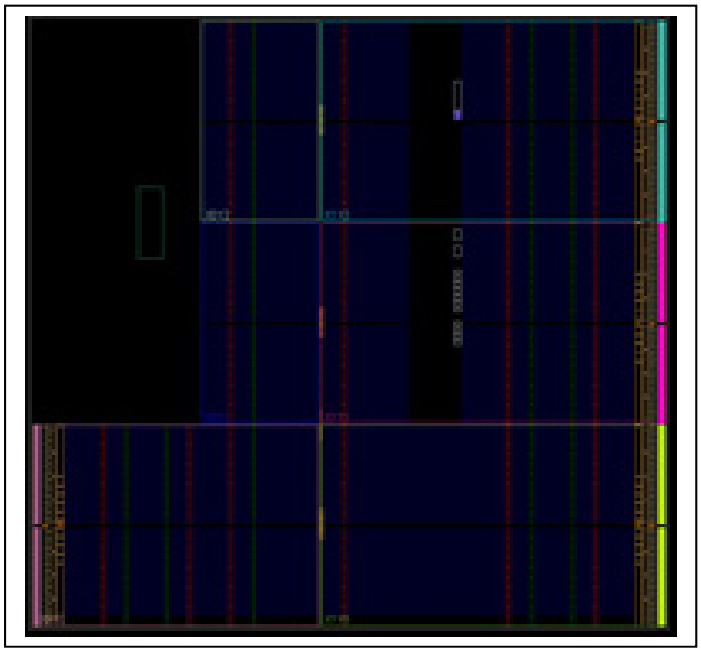

Figure 1.24 FPGA internal view before implementation

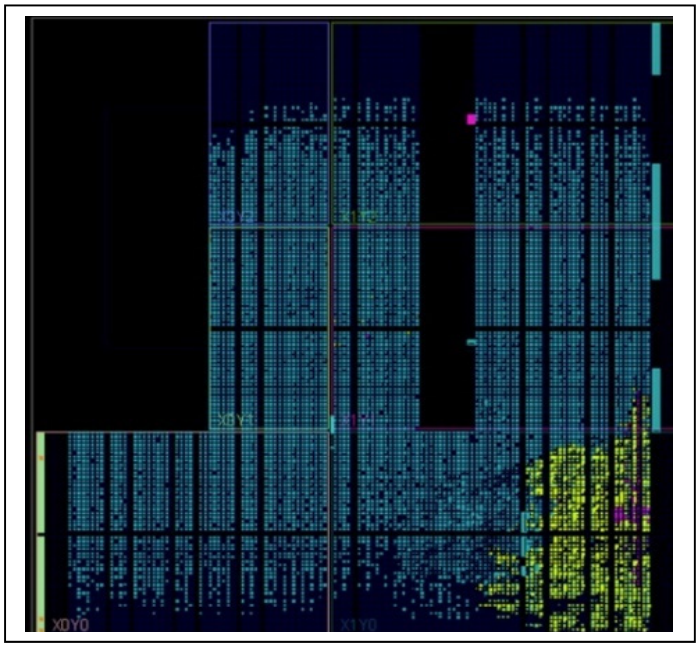

Figure 1.25 FPGA internal view after implementation

Table 1.1 shows the design utilization report, which shows the components used in each part of the design and the entire design.

Table 1.1 Design utilisation report

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	LUT Flip Flop Pairs (53200)	Block Ram Tile (140)	Bonded IOB (200)	IBUFDS (192)	BUFGCTRL (32)	MMCM (4)	PLL (4)	BSCANE2 (4)
Top	34261	36371	42	4	10053	33503	758	32168	2.5	11	1	17	2	2	1
Clk_gen	1	0	0	0	1	1	0	0	0	0	1	14	2	2	0
Clk_div	1	5	0	0	2	1	0	1	0	0	0	0	0	0	0
Dbg_hub	484	739	0	0	222	460	24	313	0	0	0	1	0	0	1
Debounce_ce	1	2	0	0	1	1	0	0	0	0	0	0	0	0	0
Debounce_rst	1	2	0	0	2	1	0	0	0	0	0	0	0	0	0
Delay	92	59	0	4	118	92	0	6	0	0	0	0	0	0	0
FSM	34	13	0	0	10	34	0	12	0	0	0	0	0	0	0
ILA	983	1677	32	0	418	825	158	544	2.5	0	0	0	0	0	0
Noise_gen	30000	30000	0	0	8156	30000	0	30000	0	0	0	2	0	0	0
Sum	2540	3595	0	0	1047	1964	576	1183	0	0	0	0	0	0	0
VIO	124	279	1	0	76	124	0	90	0	0	0	0	0	0	0

1.9 Conclusion

In this chapter, initially, the basic FPGA concepts were described to provide the design requirements. Afterward, the review of related works was done to determine the elements to complete this master's project. According to this review, the most suitable power consumption technique is the clock gating when the Xilinx BUFGCE primitives are used. Also, the TDC delay line is selected to measure the delay line because it is easy to create. Two delay elements are selected to use in the delay lines, namely buffer and CARRY4. Finally, an existing noise source that will be adapted to fulfill our needs was presented. Since we are not interested in using different clock frequencies as they add unwanted noise sources to the design and that we need to inject the noise more instantly, some changes to the selected noise block are required.

CHAPTER 2

PROPOSED DESIGN

2.1 Introduction

In this research, we want to design an FPGA-based measurement circuit to observe the impact of power noise on delay at different frequencies, based on the following specifications:

- To allow different types of measurements, for comparison purposes, such as delays and delay margins, through the use of different delay elements.
- To allow measurements with clock frequencies from 100 to 220 MHz, and to ensures that the delay elements stay unchanged and that the routing remains identical in the FPGA, from one frequency to another.
- To allow injection of noise on the power grid to emulate the clock gating and its impact of adjacent modules.
- To provide means to automatically generate files (CSV format) of the results.

This chapter will explain the details of the proposed design.

- To apply three different noise modes in the design
- To measure two transient phases in the design

2.2 Measurement circuit

This section describes the design structure in the FPGA. Figure 2.1 shows the measurement circuit top-level block diagram. There are four inputs and two outputs. The inputs are *CLOCK_IN_P, CLOCK_IN_N, CE*, and *Reset. CLOCK_IN_P* and *CLOCK_IN_N* are two differential inputs of the 200 MHz system clock, which is used as an input clock in the design. *CE* and *Reset* are provided by two user pushbuttons on the board. The first output is the Light Emitting Diode (LED) bus signal, *LED[5:0]*, whose purpose will be described later. The second output is the ILA Output that we could access on PC. As explained in Chapter 1, ILA is located inside the FPGA and monitors any required internal signals in a design. An auto-

instantiated debug core hub is used as a link to transfer the ILA captured data to the PC. The BSCANE2 primitive inside the debug core hub is connected to the dedicated PL pins, which are later connected to the JTAG interface. The Xilinx platform cable is used to connect the board (JTAG interface) to the PC (Vivado software). The signals can be monitored from the ILA waveform window in the Vivado software or generated as a CSV file for post-processing purposes. The CSV file in a particular format that provides data in a table-structured form. The CSV file contains all the design outputs and, because of that, involves multiple bits. The outputs in the CSV file are a 1-bit output to create the Most Significant Bit (MSB) output of a counter fed by the reference clock, 8-bit output to determine the delay values, and a 48-bit to show the q-carry4 output in the carry4 delay line, which will be described in following sections.

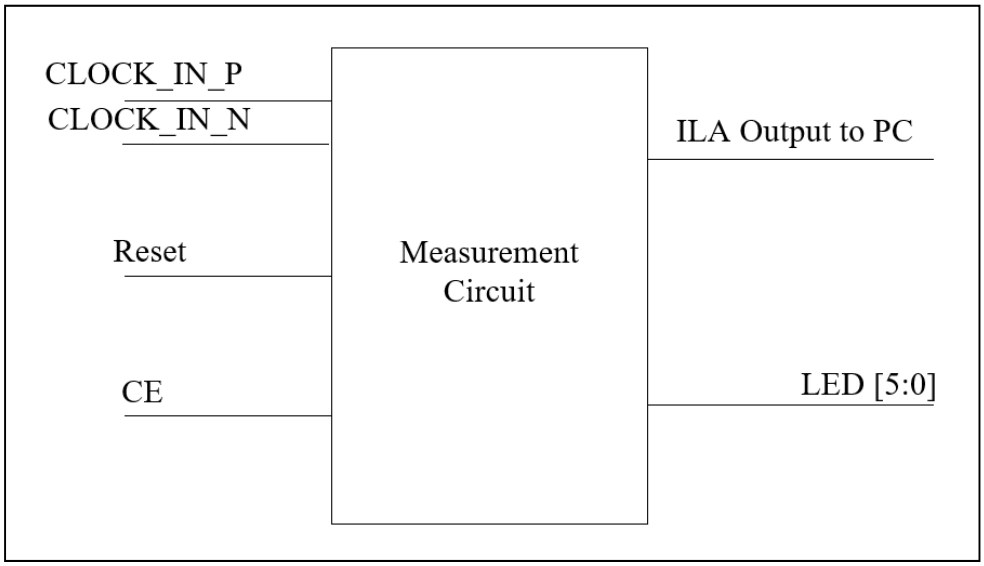

Figure 2.1 Measurement circuit top-level block diagram

Figure 2.2 shows the measurement circuit expanded block diagram that includes ten main blocks, which are:

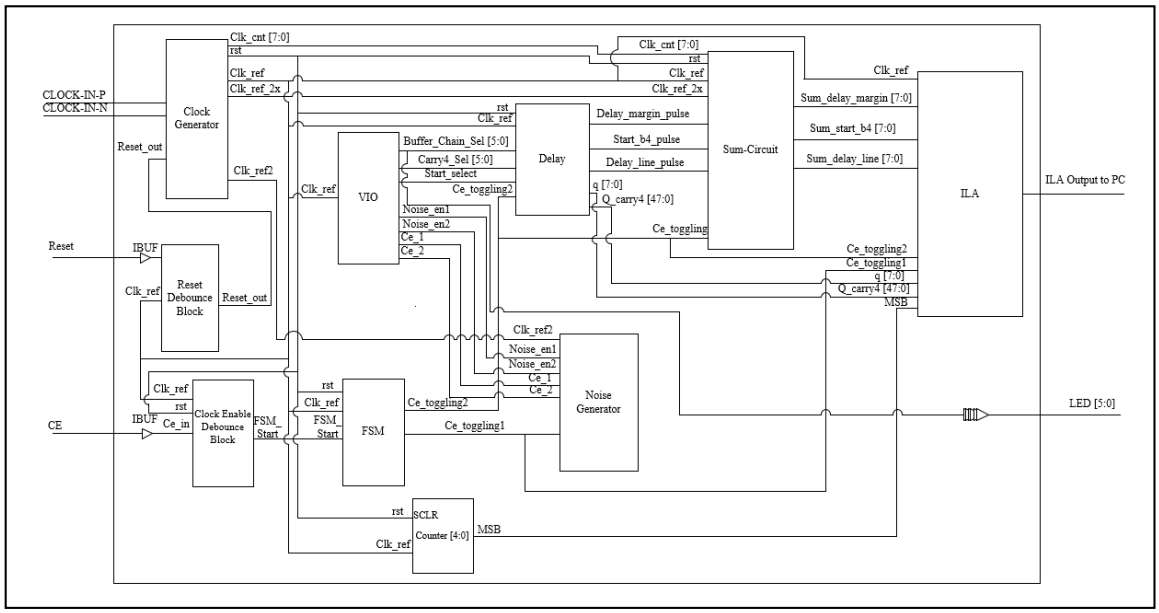

Figure 2.2 Measurement circuit expanded block diagram

- **The delay block:** This block is used to create five different outputs encapsulating the measured delays using two separate delay lines. Three of the outputs are under the form of a pulse whose width is a good approximation of the related delay. The two others are busses using thermometer encoding of the measured delays. The three pulse-shaped outputs are sent to the Sum_Circuit block to be post-processed. The delay block is described in more details in Section 2.2.1.

- **The noise generator block:** This block creates a power noise source based on the FPGA's limitations and characteristics, mainly using toggling flip-flops. By enabling and disabling this block, it creates transient phases allowing the emulation of adjacent block clock gating. The noise generator supports a few noise modes. This block is described in more details in Section 2.2.2.

- **The Sum_Circuit block:** This block is used to measure the three pulses widths generated by the delay block. This block is described in more details in Section 2.2.3.

- **The ILA block:** As mentioned in Chapter 1, this block is a Xilinx IP core, and it is used to monitor internal signals of the design. This block is described in more details in APPENDIX I.

- **The VIO block:** As mentioned in Chapter 1, this block is also a Xilinx IP core, and it is used to control internal FPGA signals. VIO allows modifying the length of both delay lines. It also controls the noise generator block. This block is described in more details in APPENDIX II.

- **The clock generator block:** This block provides the required clocks in the design that are: *Clk_cnt [7:0]*, *Clk_ref*, *Clk_ref_2x*, and *Clk_ref2*. *Clk_cnt [7:0]* is a 450 MHz clock bus with eight different phases used by the Sum_Circuit block. *Clk_ref* is the reference clock in the design. *Clk_ref_2x* is the *Clk_ref* multiplied by two, used by the Sum_Circuit block. *Clk_ref2* is a perfect copy of *Clk_ref* used as an input in the noise generator block. Also, this block generates the system reset, which is the *rst* signal. This block is described in more details in APPENDIX III.

- **The reset debounce block:** This block is used to prevent the impacts of using a user pushbutton as a *Reset* input. The user pushbutton sw7 on the board is used to reset the system manually after each experiment. Using any pushbutton causes bounces in the signal. The reset debounce block ensures that *Reset* signal is clean and without unwanted bounces. This block is described in more details in APPENDIX IV.

- **The clock enable debounce block:** This block prevents the impact of using the user pushbutton sw5 on the board as a measurement clock enable, which is *CE* input. The clock enable debounce block ensures that the *CE* signal is clean and without unwanted bounces. This block is described in more details in APPENDIX V.

- **The FSM block:** This block contains a finite state machine that creates two signals: *Ce_toggling1* and *Ce_toggling2*. *Ce_toggling1* is used by the noise generator block. It

indicates when the noise should be injected, if any. *Ce_toggling2* is used by the delay and the Sum_Circuit blocks. It indicates when the measurements should be done. This block is described in more details in APPENDIX VI.

- **The counter block:** This block generates a synchronization reference signal that corresponds to the *Clk_ref* divided by 32. This signal is read by ILA, as a means to make sure that the *Clk_ref* signal is at the right frequency. This block is described in more details in APPENDIX VII.

2.2.1 Delay block

In this section, the delay block is explained in detail. Figure 2.3 presents the delay block diagram. The delay block consists of two different delay lines, namely the carry4 and the buffer delay line, a T flip-flop, two D flip-flops, three NOT gates, a 2:1 multiplexer, and three AND gates. Both delay lines are fed by the same rising transition appearing at the start point. This rising transition is provided by the T flip-flop and can be delayed (or not) by the D flip-flop X before entering the delay lines.

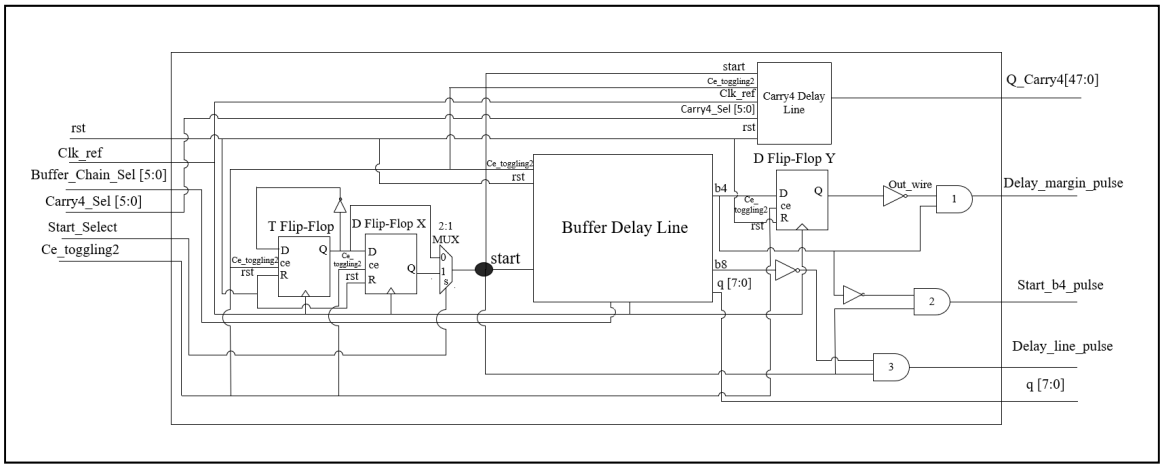

Figure 2.3 Delay block diagram

As the setup measures delay every two clock cycles, delaying the rising transition by one clock period allows measuring delays at each clock cycle (odd or even). Figure 2.4 displays the start point timing diagram.

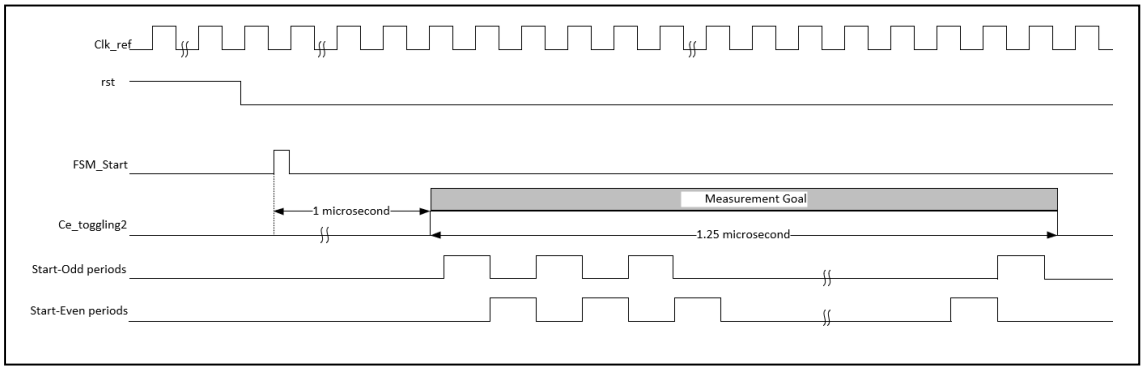

Figure 2.4 Start point timing diagram

The carry4 delay line only produces a 48-bit signal, *Q_Carry4[47:0]*, using a thermometer encoding. This delay line provides the measurements with the best resolution. Its length can be modified by the *Carry4_Sel [5:0]* signal (from VIO) to fit better the clock period used. As shown in Figure 2.5, the carry4 delay line includes a carry4 chain, 48 flip-flops, and 48 carry4 logics. There are five inputs and one input in this delay line. The inputs are: *start*, *Ce_toggling2*, *Clk_ref*, *Carry4_Sel [5:0]*, and *rst* and the output is *Q_carry4 [47:0]*.

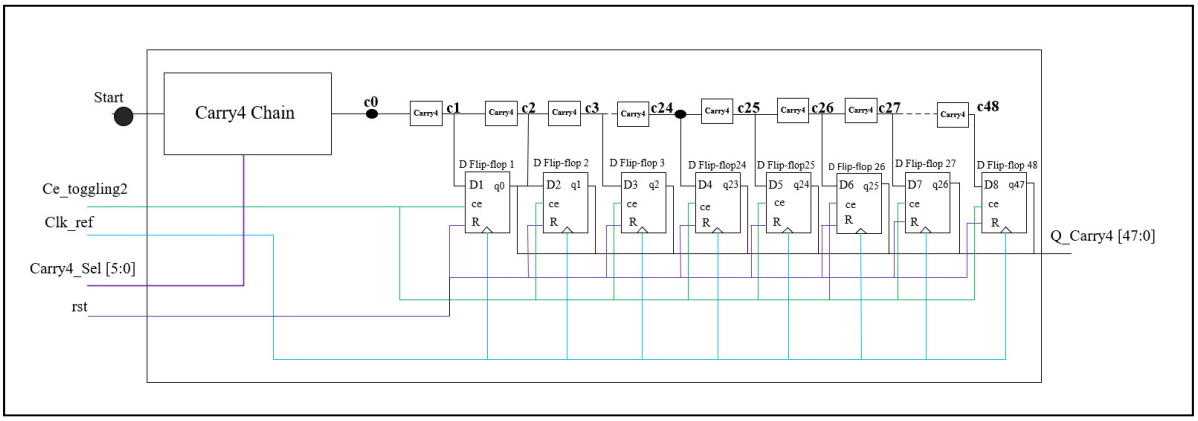

Figure 2.5 Carry4 delay line block diagram

Figure 2.6 presents the carry4 chain block diagram, which includes 36 carry4 logics and 13 LUTs. *Start* and *Carry4_Sel [5:]* are two inputs, and *c0* is the output in the carry4 chain block.

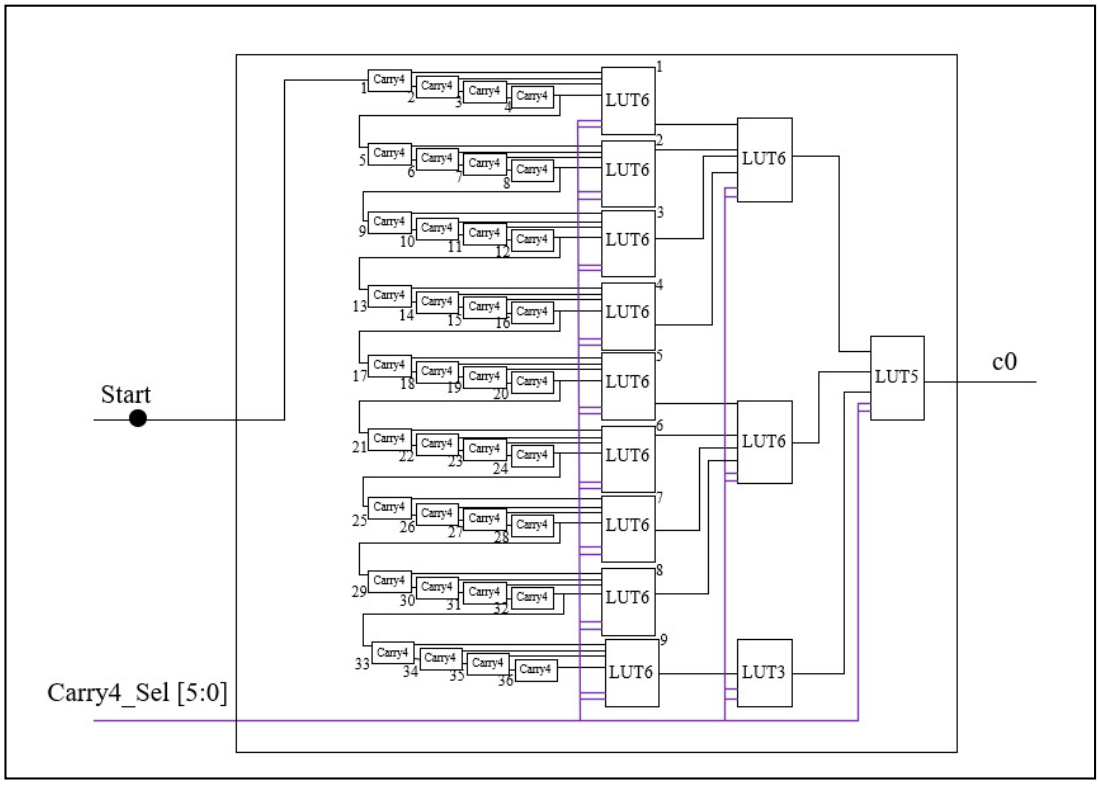

Figure 2.6 Carry4 chain block diagram

The carry4 chain inside the carry4 delay line is set to be controlled by the user. 36 carry4 logics are connected together to form the carry4 chain. Except for the first carry4, which *start* signal is connected to its CI input, in all other carry4s, the CO output is connected directly to the CI input of the next one. LUTs are used as 4:1 multiplexers where the four first pins are the inputs, and the last two are used as the select lines. Six select lines are connected to the *Carry4_Sel [5:0]*, which is controlled by the VIO. The c0 signal is connected to the first of 48 carry4 logics, each of them feeding a D flip-flop. The D flip-flops capture the data at the next *Clk_ref* rising edge, and any changes in the carry4 chain number will change the thermometer state. The carry4 chain value is adjusted to get the best value for *Q_carry4 [47:0]*, where the *Clk_ref* rising edge is in the middle of the last 48 carry4 logics. Figure 2.7 shows the carry4 delay line

timing diagram where the captured data is precisely in the middle value namely when *Q_carry4 [47:0]* contains 24 bits at zero and 24 bits at one.

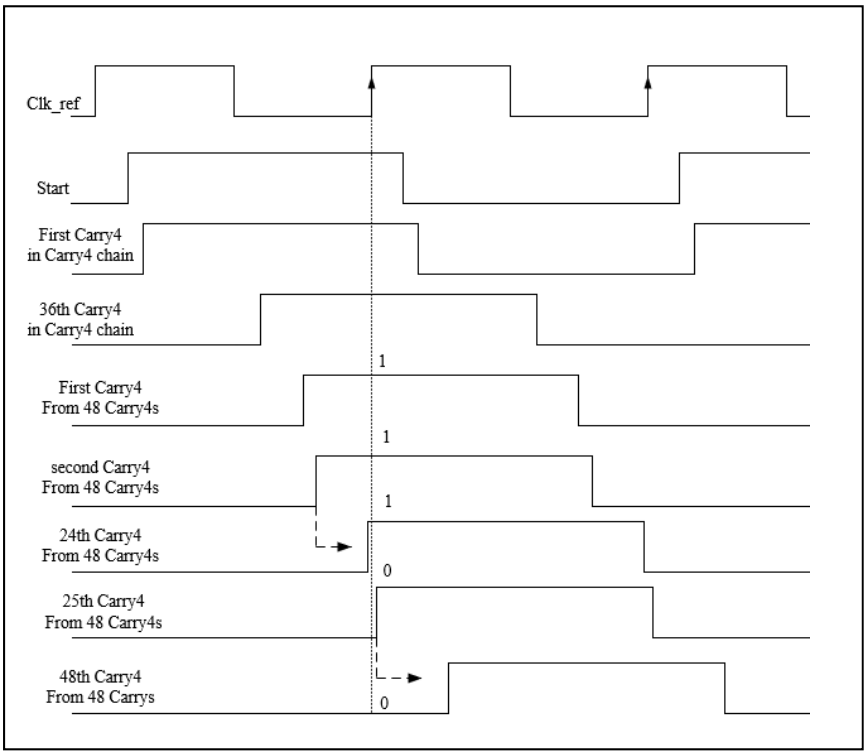

Figure 2.7 Carry4 delay line timing diagram

The buffer delay line is used to produce four different signals. The first one is an 8-bit signal, *q [7:0]*, also using a thermometer encoding. It is similar to the signal *Q_Carry4[47:0]* but with a coarser granularity. The second signal produced is called *Delay_margin_pulse*, a 1-bit signal containing a pulse whose width approximates the delay margin (namely the time between the arrival of the rising transition and the next clock rising edge). This delay margin is measured with respect to the *b4* point, which is a selected point inside the buffer delay line. The third signal produced is called *Start_b4_pulse*, a 1-bit signal containing a pulse whose width approximates the delay between the *start* and the *b4* points. Finally, the fourth signal produced is called *Delay_line_pulse*, a 1-bit signal containing a pulse whose width approximates the delay between the *start* and the *b8* (end of the delay line) points.

As shown in Figure 2.8 the buffer delay line includes a buffer chain, eight flip-flops, and eight buffers (LUT1). There are five inputs and one input in this delay line. The inputs are: *start*, *Ce_toggling2*, *Clk_ref*, *Buffer_Chain_Sel [5:0]*, and *rst*, and the outputs are *b4*, *b8*, and *q [7:0]*.

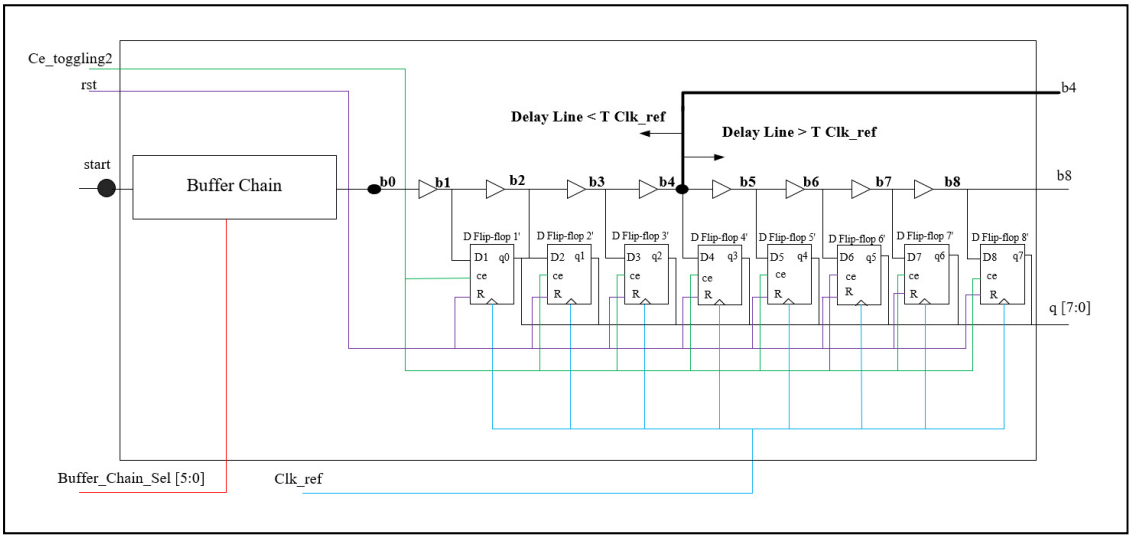

Figure 2.8 Buffer delay line block diagram

Figure 2.9 shows the buffer chain block diagram, which includes 64 buffers and a 64:1 multiplexer. *Start* and *Buffer_Chain_Sel [5:]* are two inputs, and *b0* is the output in the buffer chain block.

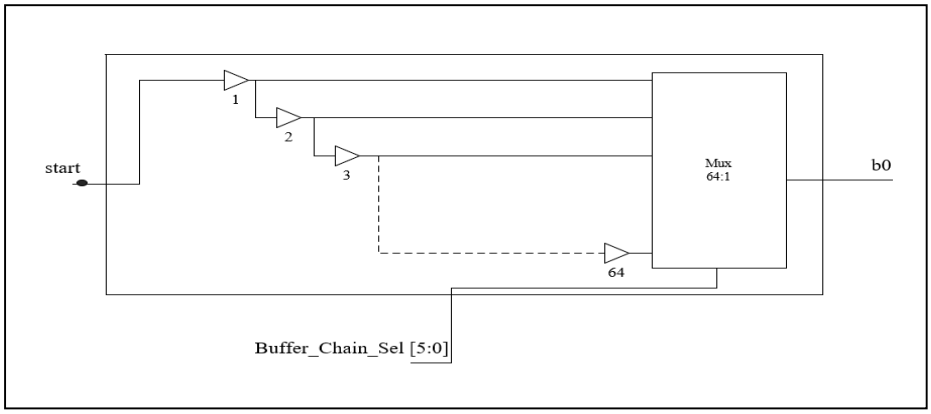

Figure 2.9 Buffer chain block diagram

The buffer chain is inside the buffer delay line and its number of buffers can be adjusted by the user. Sixty-four buffers are connected together to create the buffer chain. The first buffer input is the *start* signal, and its output connects to the next buffer. The chain is continued by connecting the output of the second buffer to the input of the third one and etc. All the sixty-four states are connected to the 64:1 multiplexer controlled by *Buffer_Chain_Sel [5:0]* from the VIO. The *b0* signal is connected to the first of eight buffers, each of them feeding a D flip-flop. D flip-flops capture the data at the next *Clk_ref* rising edge, and any changes in the buffer chain number will change the thermometer state. The buffer chain is adjusted to provide the most suitable value for *q [7:0]*, where the *Clk_ref* rising edge is ideally in the middle of the last eight buffers. Figure 2.10 shows the buffer delay line timing diagram where the captured data is precisely in the middle value namely when *q [7:0]* contains 4 bits at zero and 4 bits at one.

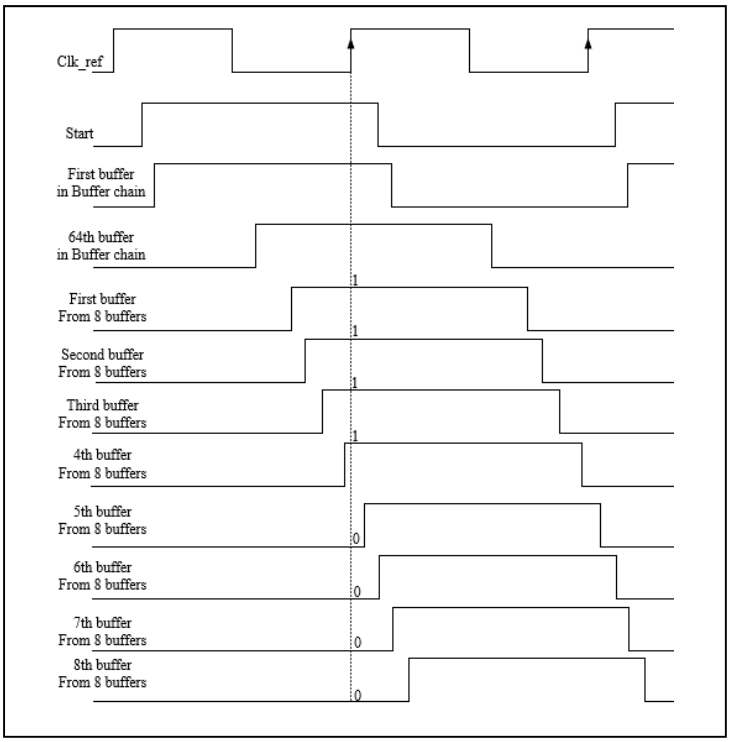

Figure 2.10 Buffer delay line timing diagram

The *Q-carry4 [47:0]* bus from the carry4 delay line and the *q [7:0]* bus from the buffer delay line are directly connected to the ILA to monitor the values. As the design goal is to observe the noise effect on delays, we need to apply a source of power noise described in detail in the following subsection. The generated pulses, which represent different delays in the design, in odd or even periods, as presented in the delay timing diagram in Figure 2.11, will go into the Sum_Circuit block, for post-processing. Sum_Circuit block diagram will be discussed in subsection 2.2.3

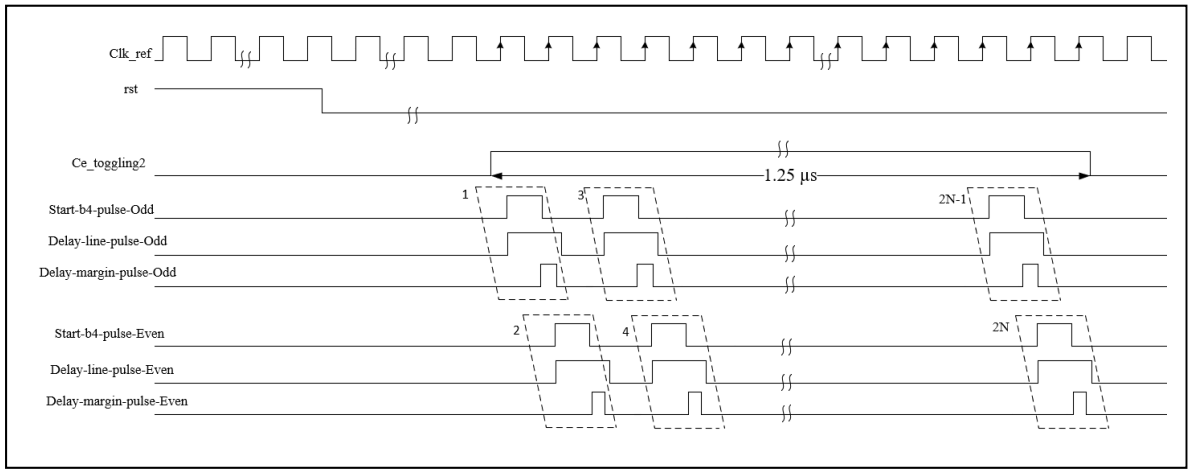

Figure 2.11 Delay block timing diagram

2.2.2 Noise generator block

The noise generator block is used to create a power noise source in the design. The design goal is to monitor the delays behavior in the presence of noise while we apply the clock gating. As presented in Chapter 1, the BUFGCE primitives could be used to create the clock gating. Enabling and disabling the noise generator block by BUFGCEs creates transient phases allowing adjacent block clock gating emulation. The main idea in the noise generator block is to use toggling flip-flops as primary noise elements. Controlling all the toggling flip-flops simultaneously by their clock or *CE* inputs will ensure that all the flip-flops are fed concurrently. Besides, there is no need to wait for many clock cycles to reach the maximum

adjusted noise because as there is no connection between the flip-flops, and the behavior of each toggling flip-flop is independent of the other ones. As well as controlling the noise generator block by BUFGCE and choosing the toggling flip-flops as noise elements, we prefer to apply two different approaches to activate the noise generator block in our design. First, by controlling the flip-flops by their clock inputs. Second, controlling the flip-flops by their inputs CE and separately from their clock.

To control the noise generator block, as shown in Figure 2.12, we need a specific *CE*, namely *Ce_toggling1*. There are two transient phases in the design. The first one occurs when *Ce_toggling1* is enabled and that the *Clk_noise1* and *Clk_noise2* are activated (if the other control signals, defined below, allow it). The second one occurs when *Ce_toggling1* is disabled, causing the *Clk_noise1* and *Clk_noise2* to be deactivated. The time between the first and second transient phases is set to be one microsecond to reach the steady-state condition.

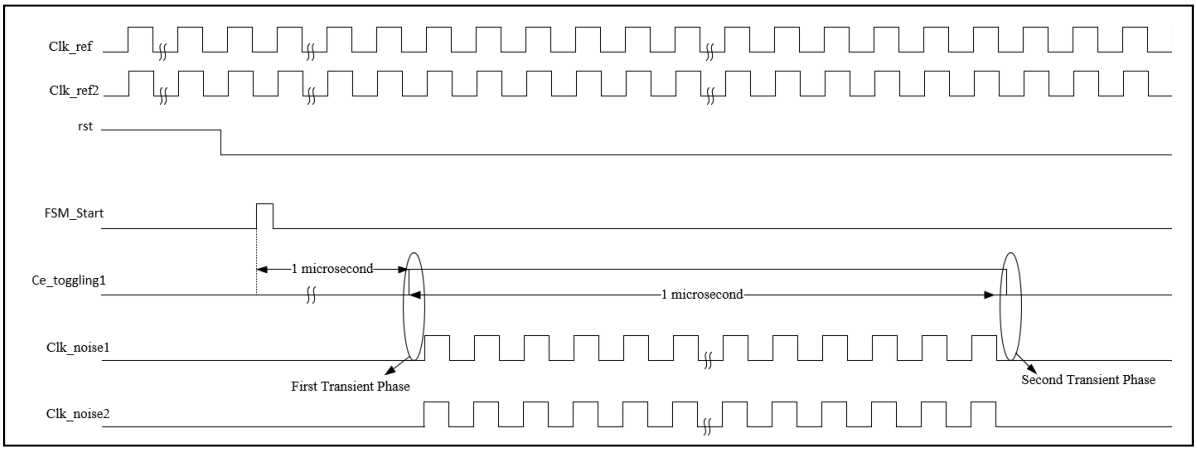

Figure 2.12 Noise generator timing diagram

Figure 2.13 presents the noise generator block diagram. The six inputs in this block are: *Clk_ref2*, *Noise_en1*, *Noise_en2*, *Ce_1*, *Ce_2* and *Ce_toggling1*. *Clk_ref2* is the perfect copy of *Clk_ref* and is created in the clock generator block. *Ce_toggling1* is one of the FSM outputs

and its activation duration is set to be one microsecond. The other inputs are controlled from VIO to change the state from enable to disable or vice versa.

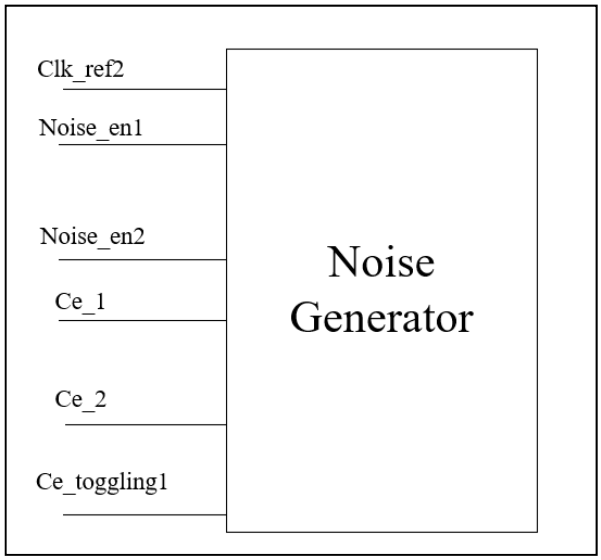

Figure 2.13 Noise generator block diagram

Figure 2.14 shows that the noise generator block includes two AND gates, two BUFGCE primitives, and two noise blocks.

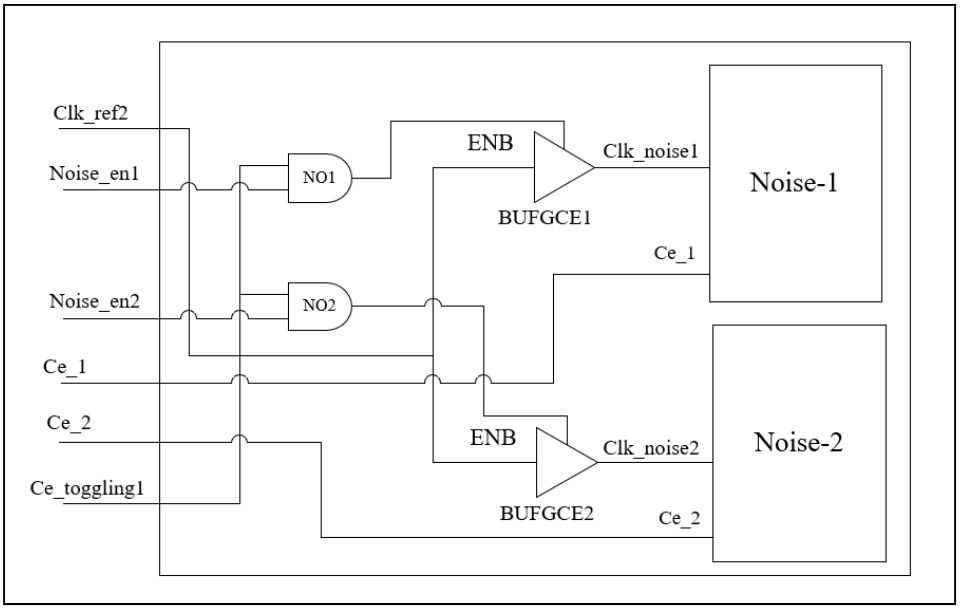

Figure 2.14 Noise generator expanded block diagram

Figure 2.15 displays the Noise_1 and Noise_2 blocks diagram in detail. Each noise block can be controlled separately. Each noise block can be set to one of the three following modes:

- **No noise:** The Noise_en1 (Noise_en2) is set to 0; there is no switching activity in the related clock signal distribution and no switching toggle flip-flops
- **Clock switching activity only:** The Noise_en1 (Noise_en2) is set to 1 but the Ce_1 (Ce_2) is set to 0; there is switching activity in the related clock signal distribution, but no switching toggle flip-flops.
- **Full switching activity:** The Noise_en1 (Noise_en2) is set to 1 as well as the Ce_1 (Ce_2); there is switching activity in the related clock signal distribution and switching toggle flip-flops.

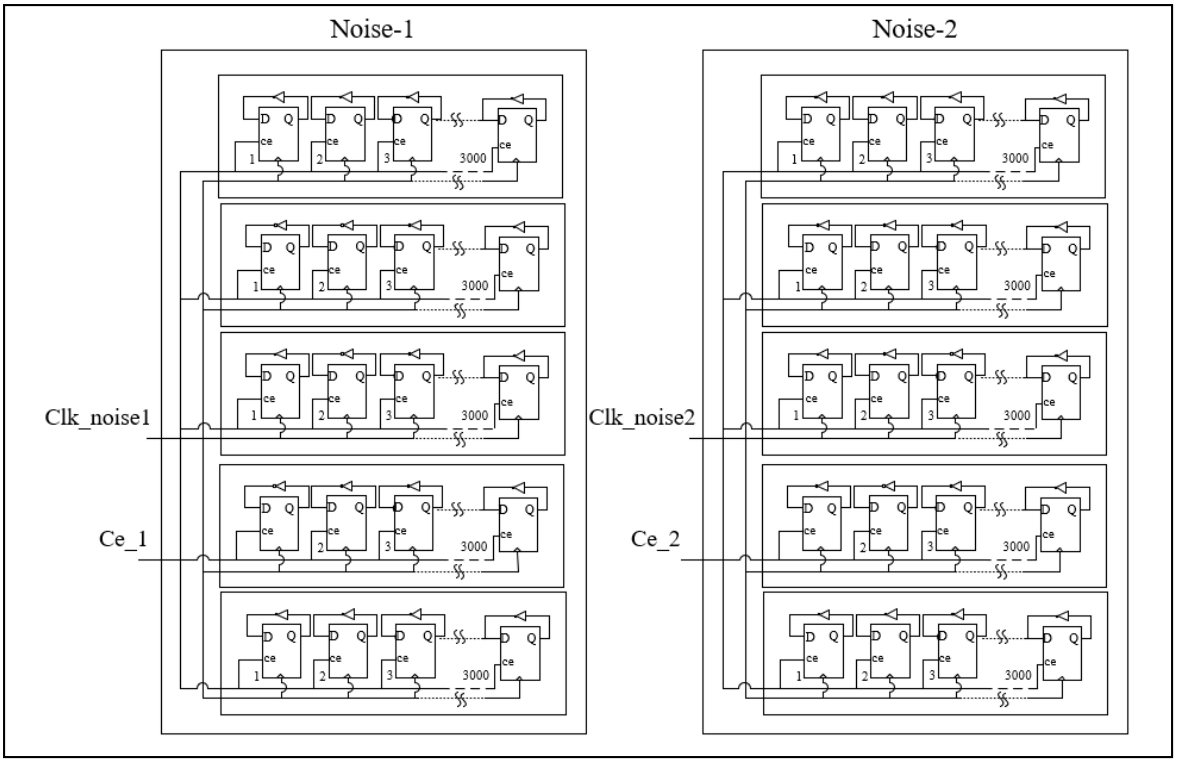

Figure 2.15 Noise_1 and Noise_2 block diagram

There are five lines of 3000 toggle flip-flops inside each noise block. *Clk_noise1* is feeding all the 15000 toggle flip-flops inside the Noise_1 block. *Clk_noise2* is feeding all the 15000 toggle flip-flops inside the Noise_2 block. The only difference between Noise_1 and Noise_2 blocks is their clock enable, which *Ce_1* is used for Noise_1 and *Ce_2* is used for Noise_2. For each toggle flip-flop in the noise generator's design, a flip-flop and a LUT are needed. Each CLB contains four flip-flops and eight LUTs. The best way to design a toggle flip-flop is to use LUT flip-flop pairs. The number of pairs inside the FPGA is limited to 53000, approximately 50% of the pairs are used to create the noise generator block in the design, and the remaining will be used for the other part of the design if required.

2.2.3 Sum_Circuit

The Sum_Circuit block is used to measure the pulses width generated by the delay block. As explained in the delay block section, the three pulse-shaped outputs are representative of the

different delays in the design. The mechanism used in the measurement is counter-based. A high-speed clock (*Clk_cnt* = 450 MHz) is used to measure each pulse width. Measurement is taken eight times with eight different *Clk_cnt* phases to increase the design resolution and have better width estimations of the generated pulses in both odd and even measurement modes. The measurement is performed in the period of 1.25 microseconds when the *Ce_toggling2* is enabled. This time exactly starts when the noise block is enabled but continues 0.25 microseconds more than the time, we disable the noise block. Figure 2.16 illustrates the Sum_Circuit block diagram.

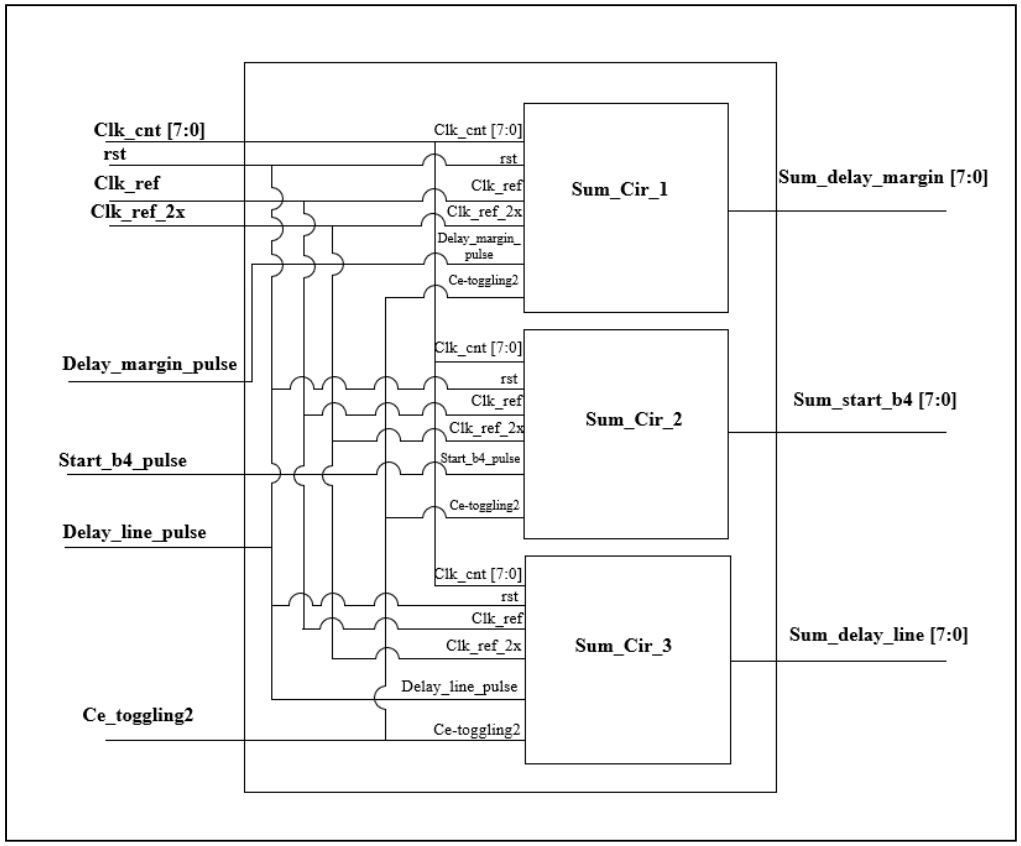

Figure 2.16 Sum_Circuit block diagram

The eight inputs in this block are *Clk_cnt [7:0]*, *rs*t, *Clk_ref*, *Clk_ref_2x*, *Delay_margin_pulse*, *Delay_line_pulse*, *Start_b4_pulse* and *Ce_toggling2*. The three outputs in this block are *Sum_delay_margin [7:0]*, *Sum_start_b4 [7:0]*, *Sum_delay_line [7:0]*.

Three blocks inside the Sum_Circuit are used to measure and store the pulses width at eight different *Clk_cnt* phases. Sum_Cir_1, Sum_Cir_2, and Sum_Cir_3, respectively, are used to measure the *Delay_margin_pulse*, *Start_b4_pulse*, and *Delay_line_pulse* width.

As shown in Figures 2.17 to 2.19, each Sum_Cir block is used to measure one pulse width and contains eight identical Meas blocks, each of them working with a different *Clk_cnt* phase. The output of each of Sum_Cir block is the sum of all eight Meas block outputs.

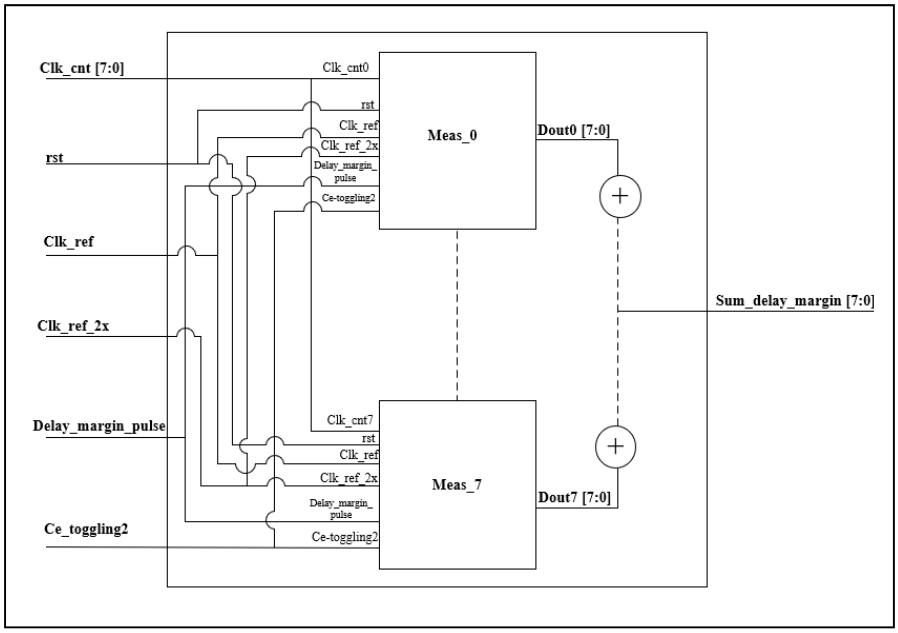

Figure 2.17 Sum_Cir_1 block diagram

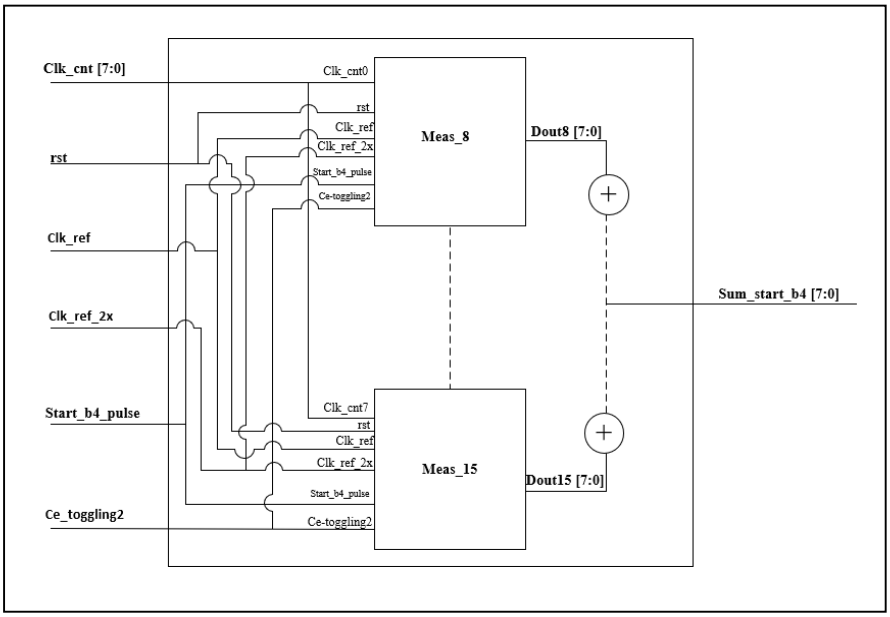

Figure 2.18 Sum_Cir_2 block diagram

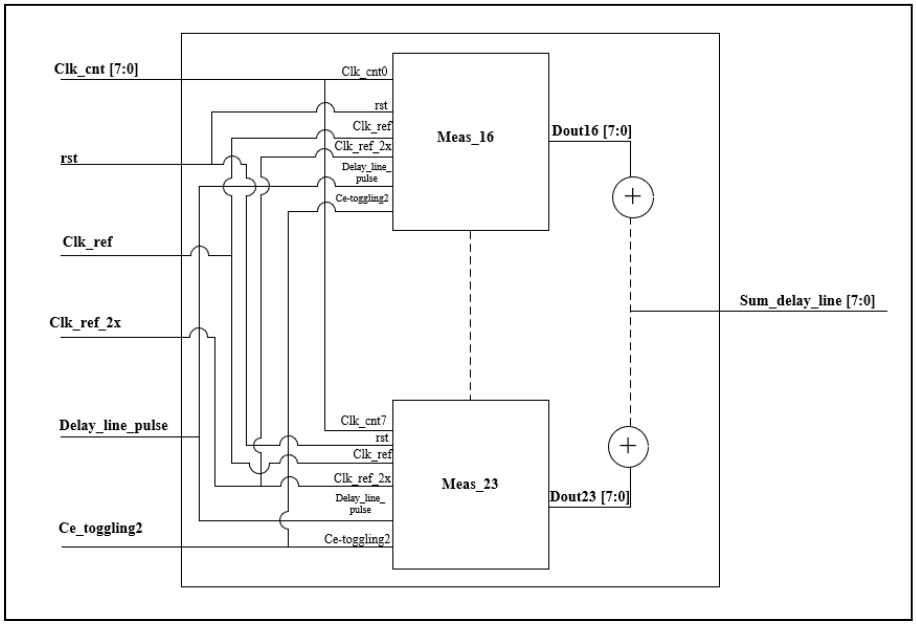

Figure 2.19 Sum_Cir_3 block diagram

Sum_Cir blocks are identical, only their input pulses and their outputs differ. Each Meas block contains one AND gate, two NOT gates, a M_counter block, a FIFO, and a register. As an example, Figure 2.20 shows the Meas0 block diagrams in detail.

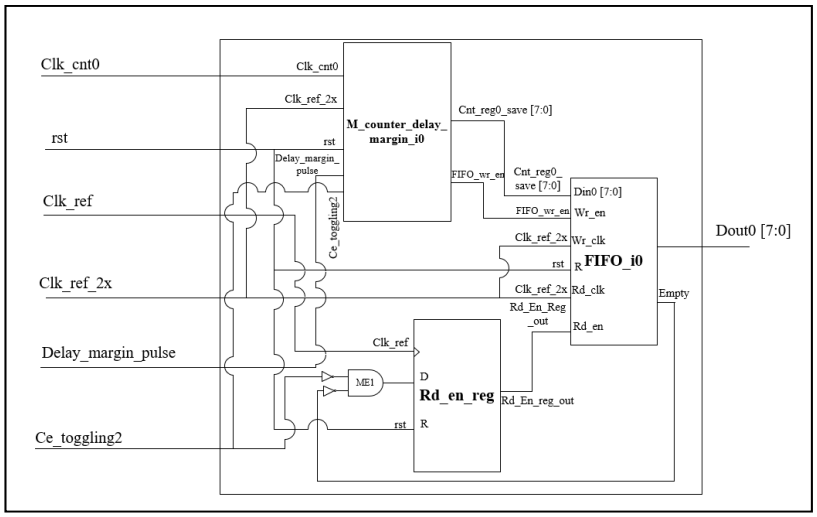

Figure 2.20 Meas0 block diagram

As we can find in Figure 2.20, the Meas0 block is used to measure the *Delay_margin_pulse* width. M_counter_delay_margin_i0 is the part that is counting the pulse width based on the number of *Clk_cnt0* period with respect to *Ce_toggling2* enabling time. The M_counter_delay_margin_i0 block diagram design details are provided in Appendix VIII.

2.3 Conclusion

One of the purposes of this design is to ensure that we could implement the experiments under similar conditions and avoid modification. To do that, the entire block's position in the implementation is fixed. Besides, as explained in subsection 2.2.1, two different delay elements are used to create delay lines in the design to have a wide frequency range for the measurement.

Furthermore, the noise source is created based on the explanations that allow applying three different modes: no noise, clock switching activity only, and full switching activity in the measurement circuit. Finally, as explained in subsection 2.2.3, the Sum_Circuit block is used to measure the three pulses widths generated by the delay block.

CHAPITRE 3

RESULTS AND ANALYSIS

3.1 Introduction

From the proposed design presented in Chapter 2, the following measurements are taken:

- *Q_carry4 [47:0];*
- *q [7:0];*
- *Start_b4_pulse;*
- *Delay_line_pulse;*
- *Delay_margin_pulse.*

As mentioned in Chapter 2, the measurements are done on either the odd or the even samples, in different noise modes. The first two measured values (*Q_carry4 [47:0]* and *q [7:0]*) are directly connected to the ILA. The last three measured values (*Start_b4_pulse*, *Delay_line_pulse*, *Delay_margin_pulse*) are pulses and are first processed using an 8-phases clock circuit. The resulting measurements, called *Sum_start_b4 [7:0]*, *Sum_delay_line [7:0]*, and *Sum_delay_margin [7:0]*, represent 8 times the related pulse width expressed in terms of *Clk_cnt* periods (one period = 2.2ns). Those are values sent to the ILA. Some processing is therefore required to get the final pulse width estimation, namely divide them by 8 and then multiply them by 2.2, leading to three following final estimations: *Ave_Start_b4, Ave_Delay_line, Ave_Delay_margin.*

The TDC delay was adjusted such that *Start_b4_pulse* < T_{clk_ref} , *Delay_line_pulse* > T_{clk_ref} , where T_{clk_ref} is one *Clk_ref* period.

As also mentioned in Chapter 2, each noise block of the noise generator can be set to 3 different modes: 1) No noise, 2) clock switching activity only, and 3) full switching activity.

To ensure that the *Clk_ref* is at the desired frequency during the experiments, this signal is used to clock a 5-bit counter, whose MSB is monitored by the ILA, which is also clocked by the *Clk_ref* signal. Figure 3.1 shows the example of the monitored counter MSB (whose frequency is 1/32 of that of *Clk_ref*) by the ILA.

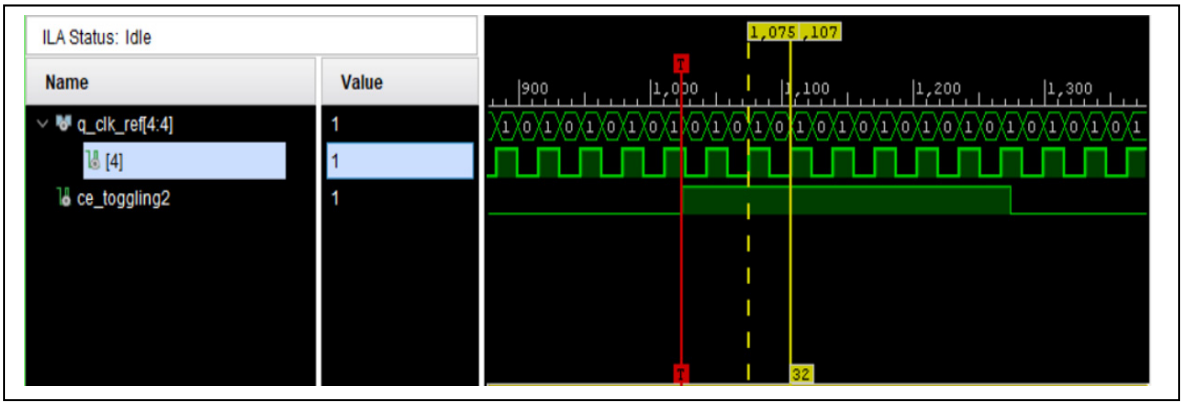

Figure 3.1 Example of the counter MSB monitored by the ILA

As shown in the global design timing diagram in Figure 3.2, the measurement period is 1.25 microseconds. Also, with the same start point, the noise, if any, is activated for one microsecond. As mentioned before, there are two transient phases during this period. The first transient phase always happens at the beginning of the measurement, and the second one is happening after the time we stop the noise by deactivating the *Ce_toggling1*. The widths of three measured pulses are estimated in both even and odd periods, based on the number of *Clk_cnt* periods sampled at eight different phases. The odd and even results are sorted and added separately, are sent to the ILA to monitor, and then to the PC by using the communicaton channel embedded in the JTAG interface. The data are combined and interpret in the CSV file with a fully automatic process in the last step.

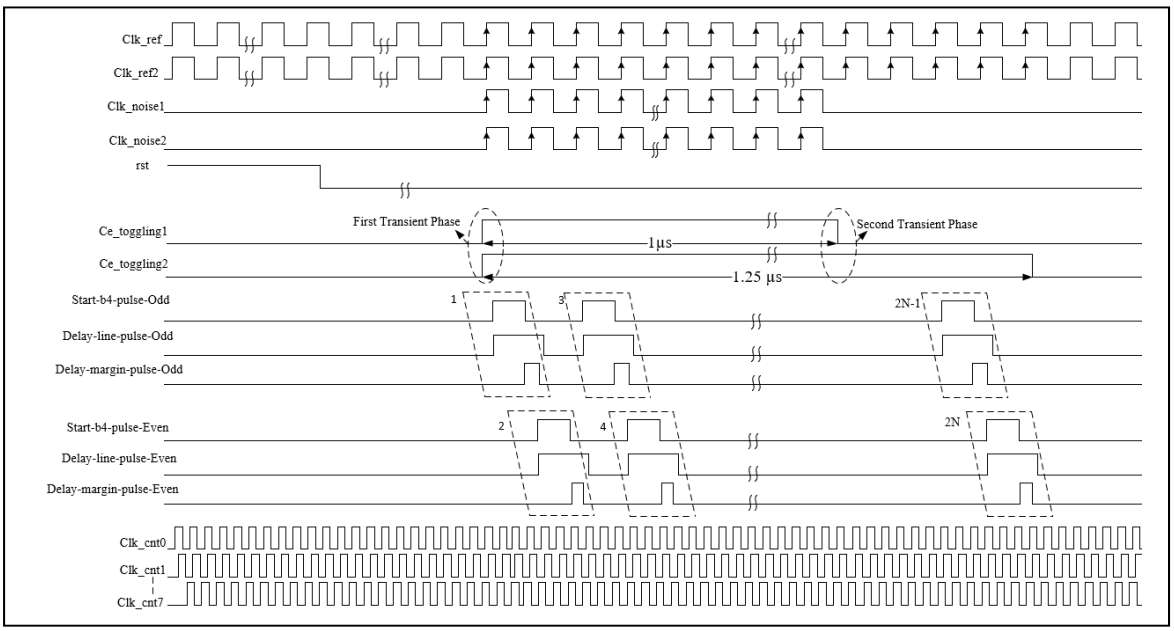

Figure 3.2 Global design timing diagram

3.2 Adjustments with respect to frequency

One design goal is to collect the measurements at different frequencies, from 100 to 220 MHz. Since the measurement time is set to be 1.25 microseconds, and that the noise activation time is set to be one microsecond, the determination of the total sample number, the location of both transients, and of the best delay elements value at each frequency are essential. The following subsections describe them in detail.

3.2.1 Sample number at each frequency

Indeed, at each frequency, the total sample number varies based on the *Clk_ref* period. Increasing the clock speed will cause a decrease in the clock period, and as a result, the total sample number will be increased. Table 3.1 presents the total sample number at each frequency. Also, the location of the transients based on the sample number is provided in this table. The total samples' number at each frequency is obtained when the measurement time

(1.25 microsecond) is divided by the *clk_ref* period. The first transient is always located at the beginning of the measurement and in sample three. The first sample for both even and odd periods is zero because the whole register's values are initially zero. The second transient location is also essential because we need to find where the noise effect, due to the clock gating disabling (if any), appears during the measurements. The second transient location is obtained by dividing the noise activation time (one microsecond) by the *Clk_ref* period.

Table 3.1 Total sample number and the location of the transients at each frequency

Clk_ref (MHz)	Clk_ref Period (ns)	Total Sample Number	First Transient Location-Sample Number	Second Transient Location-Sample Number
100	10	125	3	100
120	8.33	150	3	120
140	7.14	175	3	140
160	6.25	200	3	160
180	5.55	225	3	180
200	5	250	3	200
220	4.54	275	3	220

3.2.2 Delay element value at each frequency

As mentioned before, there are two delay lines in our measurement circuit. Recall that each of them starts with a programmable part containing series of delay elements that can be adjusted, via the VIO, the best possible measurement setup. As described in Chapter 2, the delay element in the buffer delay line is a buffer or LUT, and in the carry4 delay line, the delay element is a carry4 logic. Preliminary experiments were performed in order to find the best number of delay elements to put in the programmable part of each delay line, which is the one providing the delay line middle value, namely when *Q_carry4 [47:0]* contains 24 bits at zero and 24 bits at one, and when *q [7:0]* contains 4 bits at zero and 4 bits at one, respectively. Table 3.2 shows the best delay element number in the programmable part of each delay line.

Table 3.2 Best delay element number in the programmable part of each delay line

Frequency (MHz)	Carry4-Num	Buffer-Num
100	-	13
120	36	8
140	19	5
160	14	3
180	8	1
200	1	-
220	1	-

3.3 Results collection and analysis

The results are divided into two parts:

1) Carry4 delay line results;
2) Buffer delay line results.

For both parts, sampling is done in two separate stages, which include odd periods sampling and even period sampling. Each experiment is repeated twenty times for both odd and for even periods and averaged to reduce non-correlated noise effects. After the generation of both (even, odd) sample set, the results are merged in an interleaved way, and their variance is calculated. The steady-state value, which is the value just before the beginning of the second transient phase, is used as the reference value because it is stable, and the fluctuations are minimum. In order to normalize the results over the frequency range used, we subtract the reference value from the value of measurements taken in the two transient phases. The following formula is used to calculate this difference

$$\text{Diff\_merged} = \text{Diff}(i,0) \text{ if } i \text{ Even} , \text{Diff}(i,1) \text{ if } i \text{ Odd} \quad (3.1)$$

Where:

$$\text{Diff}(i,0) = \text{Value}(i,0) - \text{Reference Value} \quad (3.2)$$

$$\text{Diff}(i,1) = \text{Value}(i,1) - \text{Reference Value} \quad (3.3)$$

3.3.1 Carry4 delay line results

As mentioned in Chapter 2, *Q_carry4 [47:0]*, the carry4 delay line output, is used to find the noise effect on delays. *Q_carry4 [47:0]* is connected directly to the ILA and it uses thermometer encoding. From the literature review, we found that the delay element value in this delay line has the low value around 60 ps, and there is no net delay to connect each carry4 logic to the next one. Various experiments were done at different frequencies to provide an estimate for carry4 logic delay. The frequency range for this delay line is from 120 to 220 MHz.

As detailed in Appendix IX, the estimated carry4 logic delay value is about 55 ps. This short delay value allow the detection of rather small delay changes. It also allows using a greater frequency range when compared to the buffer delay line, which is discussed later.

As explained the *Q_carry4 [47:0]* values are generated 20 times for both even and odd periods and their variance is calculated. Finally, the Diff-merged value is calculated. Figures 3.3 and 3.4 show the Diff_(I,1) and Diff_(I,0) *Q_carry4 [47:0]* at 220 MHz with full switching noise activity, for even and odd periods, respectively. The results are presented in the time domain (x-axis), by multiplying the *Q_carry4 [47:0]* sample number by the clock period. This allows comparing results obtained at different frequencies. To get the complete picture, results from odd and even periods need to be merged in an interleaved fashion. The result of this merging operation appears in Figure 3.5.

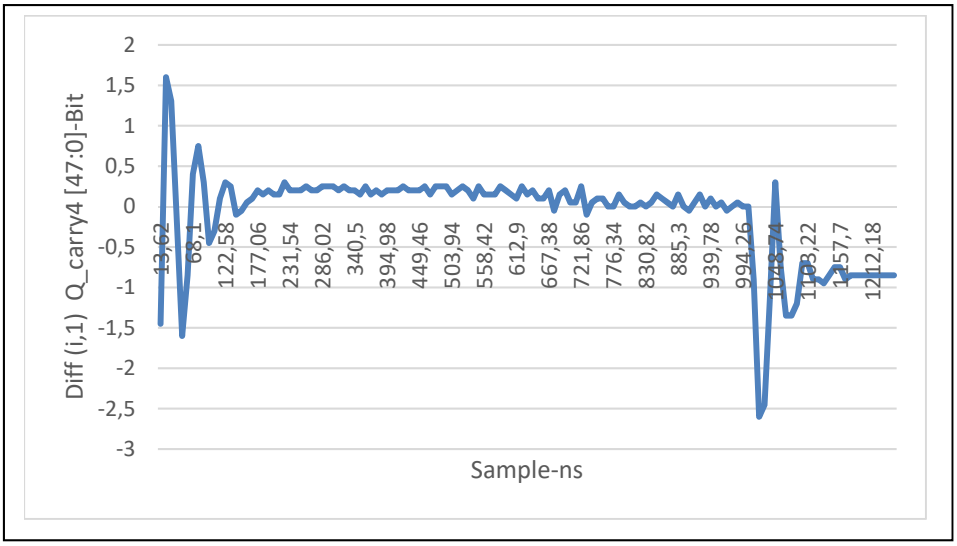

Figure 3.3 Results from odd periods (Diff (i,1)) for Q_carry4 [47:0] at 220 MHz, full switching noise activity

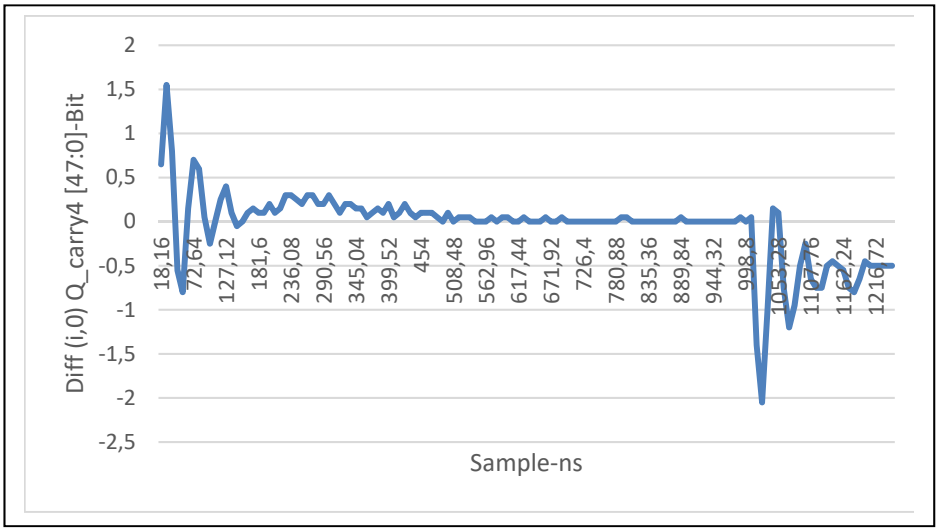

Figure 3.4 Results from even periods (Diff (i,0)) for Q_carry4 [47:0] at 220 MHz, full switching noise activity

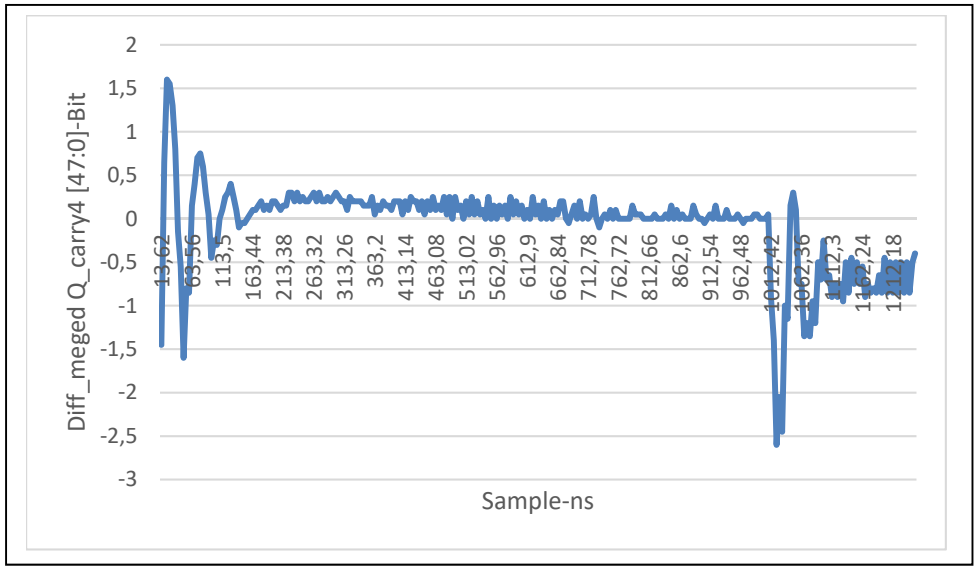

Figure 3.5 Merged results (Diff_merged) for Q_carry4 [47:0] at 220 MHz, full switching noise activity

The experiments were performed (full switching noise activity) at other frequencies by changing the *Clk_ref* to find a reliable and robust pattern. The merged results obtained at 200 and 180 MHz are shown in Figures 3.6 and 3.7, respectively. Additional *Q_carry4 [47:0]* results are presented in APPENDIX X.

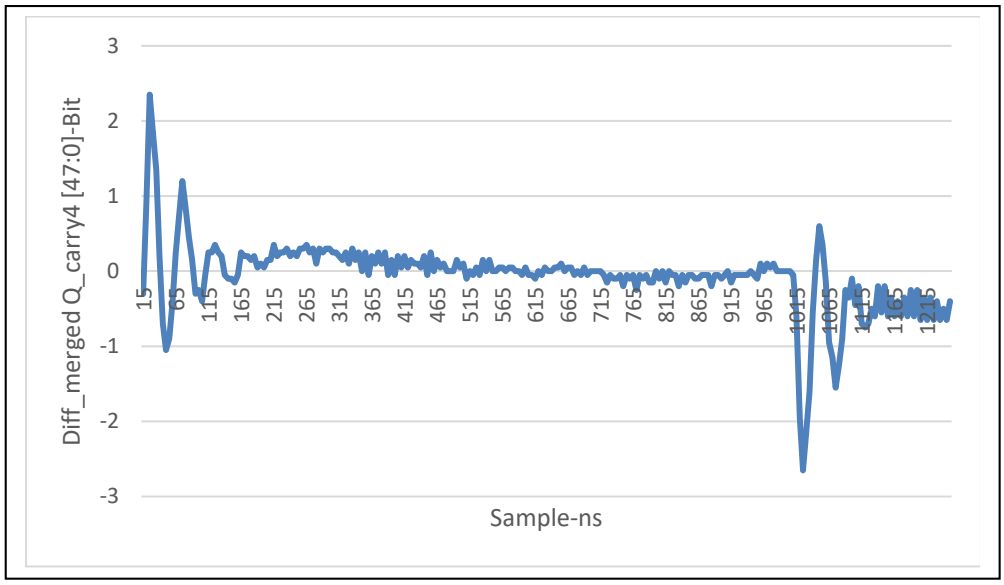

Figure 3.6 Merged results (Diff_merged) for Q_carry4 [47:0] at 200 MHz, full switching noise activity

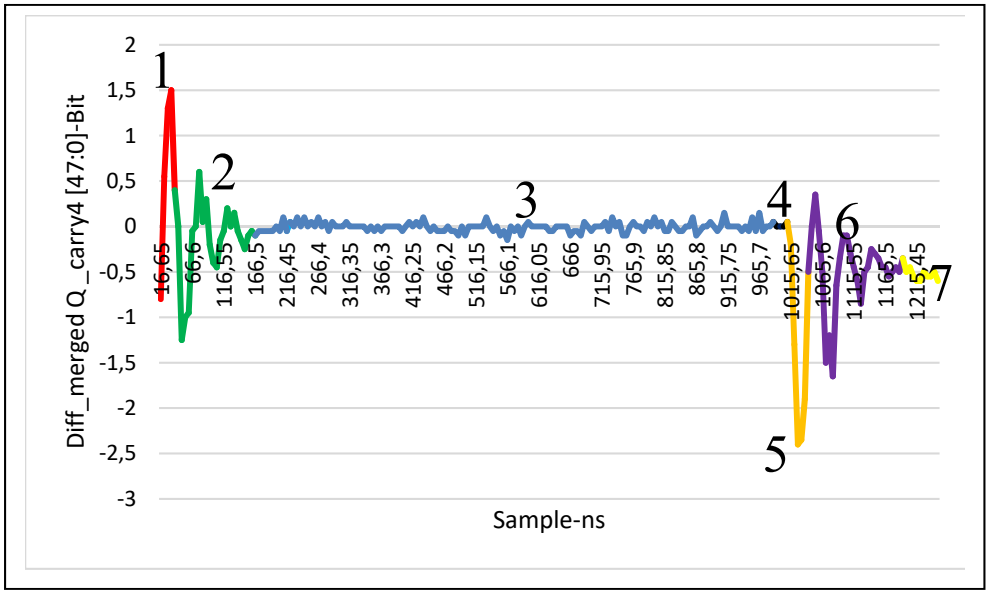

Figure 3.7 Merged results (Diff_merged) for Q_carry4 [47:0] at 180 MHz, full switching noise activity

In the previous figures, we could observe that the changes follow a similar pattern. As shown in Figure 3.7 when the *Clk_ref* is 180 MHz, the pattern obtained from the results at different frequencies can be divided into seven parts as below:

3) There is a sharp increase in the delay value when the noise starts. The first transient starts at this moment (highlighted in red).

4) After the first dramatic rise, some (medium size) fluctuations occur in the delay. This part usually ends before the twentieth sample after the first transient (highlighted in green).

5) There are small fluctuations in the delay (highlighted in blue).

6) There is a rather constant short behavior before the second transient. This part (highlighted in black) shows that the delay value have reached the steady-state condition. .

7) There is a sharp decrease in the delay value when the noise stops. The second transient starts at this moment (highlighted in orange).

8) After the second dramatic decrease, some (medium size) fluctuations occur in the delay. This part usually ends before the twentieth sample after the second transient (highlighted in purple).

9) There is a delay reduction in this part. The delay value is reduced until it reached the value of the no-noise state in the design (highlighted in yellow).

As the delay of a gate increases when the VDD value decreases (and vice and versa), we can assume, in absence of direct VDD measurements, that the shape of the delay curves gives us some hints about the shape of the VDD ones. The sharp increase in the delay value at the beginning of the first transient phase is coherent with the fact that the start of the noise activity induced a sharp increase in current consumption, which in turn leads to a VDD droop. This type of behavior was expected and in line with other similar observations in the literature.

In another experiment, we compare the Diff_merged results for different noise modes at 220 MHz. As shown in Figure 3.8, three modes are compared. The first one (highlighted in black) is the no noise mode. The second one ((highlighted in orange) is the clock switching activity

only. The third one (highlighted in blue) is the full switching activity. The results show that fluctuations in the clock switching activity reach about the same maximum and minimum values but last shorter than the ones observed with full switching activity. This suggests that the noise induced by the clock distribution network itself is more important than the noise induced by the switching FFs.

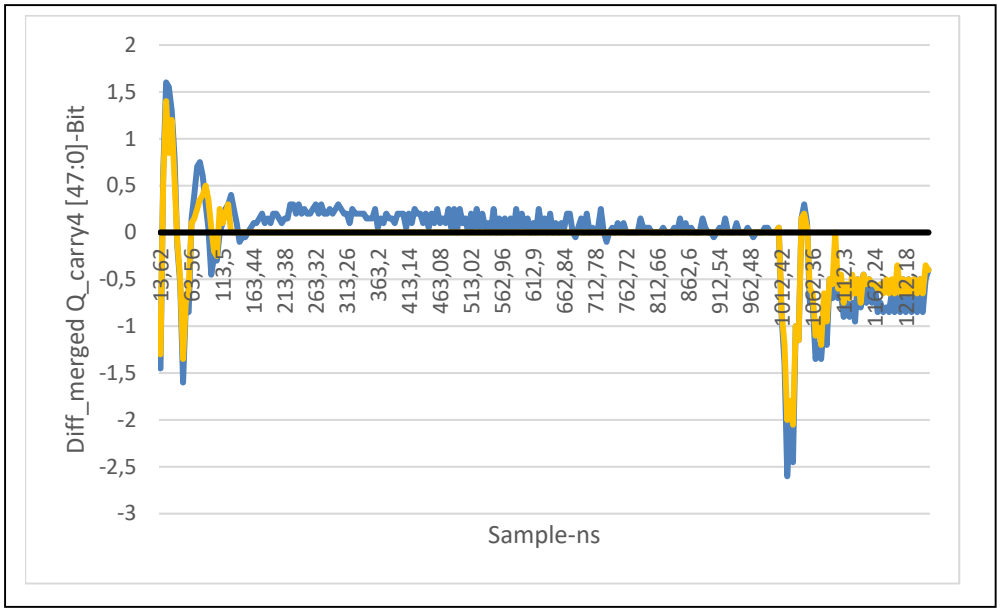

Figure 3.8 Merged results (Diff_merged) for Q_carry4 [47:0] at 220 MHz for different noise modes

Note that we expect enabling only one noise block at the time would lead to results with less significant fluctuations in the delay values.

Table 3.3 shows the delay fluctuation for different frequencies and noise modes. The first column from the left shows the frequency and the second column shows the three different noise modes at each frequency. The four other columns show the maximum and minimum values for the first and second transient phases. All Table 3.3 results are calculated from Diff-merged results. For example, at 220 MHz when no noise is applied, there is no observed fluctuation in the first and second transient phases, all the values

being equal to zero. Furthermore, always at 220 MHz, in the Clock Switching Activity Only (CSAO) mode, the first transient maximum value is estimated at 77 ps. This value is obtained by multiplying the peak value of 1.4 (Fig. 3.8) by the estimated carry4 delay (55 ps). The biggest delay fluctuations were obtained at 200 MHz, in full switching activity mode, with a maximum value of 129 ps for the first transient phase and a minimum value of -146 ps for the second transient phase.

Table 3.3 Delay fluctuations (ps) for different frequencies and noise mode for Q_carry4 [47:0] results (CSAO = Switching Activity Only, FSA = Full Switching Activity)

Frequency (MHz)	Noise mode	First Transient Max Value	First Transient Min Value	2nd Transient Max Value	2nd Transient Min Value
120	No	0	0	0	0
	CSAO	69	-17	0	-63
	FSA	85	-34	0	-85
140	No	0	0	0	0
	CSAO	69	-25	0	-74
	FSA	96	-41	0	-96
160	No	0	0	0	0
	CSAO	85	-30	0	-80
	FSA	96	-41	0	-96
180	No	0	0	0	0
	CSAO	69	-58	14	-116
	FSA	83	-69	19	-132
200	No	0	0	0	0
	CSAO	107	-50	74	-107
	FSA	129	-58	96	-146
220	No	0	0	0	0
	CSAO	77	-74	13	-121
	FSA	88	-88	17	-143

3.3.2 Buffer delay line results

This section describes the results obtained from five outputs, which are: *LED [5:0]*, *Ave_start_b4, Ave_delay_line*, *Ave_delay_margin* and *q [7:0]*. The delay element value in this delay line has a larger value than the one of the carry4 delay line. Based on the worst-case

delay in static timing, the buffer element delay value is around 124 ps plus the net delay. Because of that, the application *Clk_ref* frequency range is reduced to [100 MHz, 180 MHz]. As mentioned before, the *LED [5:0]* output is there to verify that the correct buffers' number is used during the experiments, which is visible on the board. Figure 3.9 shows a *LED [5:0]* example when the buffer number is set to seven. The 6-bit binary output, in this case, is 000111, which represents the number seven in the adjustable buffer chain in this delay line. The results collection method is similar to the one used for *Q_carry4 [47:0]*, meaning that results must be merged in an interleaved way and that equations 3.1 to 3.3 must be used for *Ave_start_b4*, *Ave_delay_line*, *Ave_delay_margin* and *q [7:0]*.

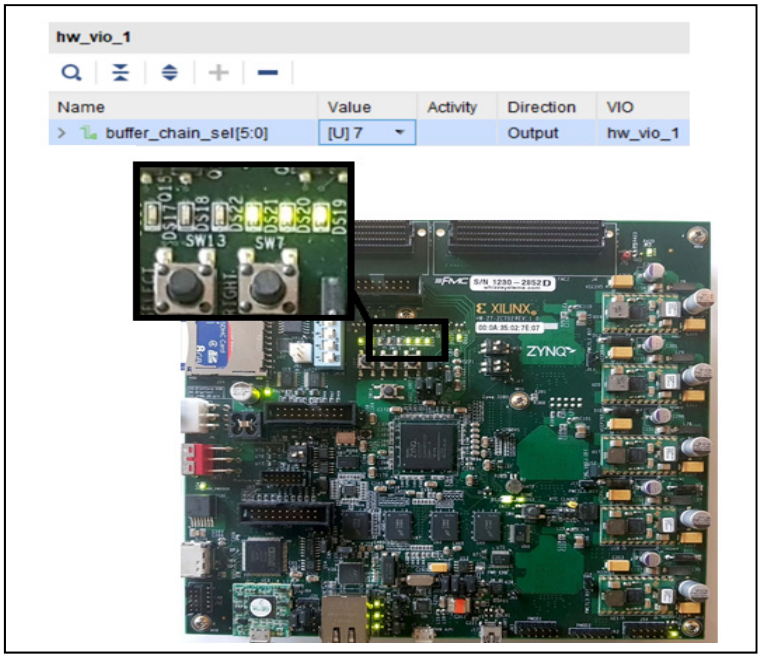

Figure 3.9 LED [5:0] example when the buffer number is seven in the VIO

As explained in Chapter 2 and shown in Figure 3.10, the delay block is used to create *q[7:0]*, *Sum_start_b4 [7:0]*, *Sum_delay_line [7:0]* and *Sum_delay_margin [7:0]*.

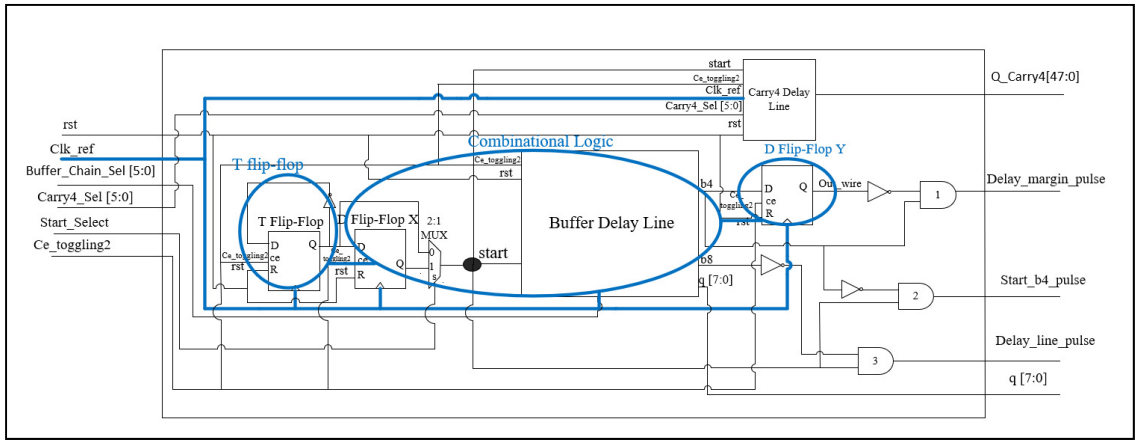

Figure 3.10 Delay block

As also described in Chapter 2, while the *q[7:0]* is directly sent to the ILA, the three others are pulses post-processed by the Sum_Circuit block, which produces *Sum_delay_margin*, *Sum_start_b4* and *Sum_delay_line.* These three signals are then monitored by the ILA, and post-processed (namely divided by eight and multiplied by the 450 MHz clock period, 2.2ns) in an Excel file to get *Ave_delay_margin, Ave_start_b4* and *Ave_delay_line,* respectively. Let us recall that these three last signals represent the avarage value of the delay margin (time between *b4* and the next *Clk_ref* rising edge), the delay between the *start* and the *b4* points, and the delay between the *start* and the *b8* points, respectively (see Fig. 3.11).

In some cases, we need to consider the delay between the FF launching the rising transition and the start point.This delay was estimated and taken into account when relevant. Details are provided in APPENDIX XI. Taking into account this delay allows dealing with the difference between the odd and even measurement paths and gives a better estimate of the delay form the rising edge of the clock for *Ave_start_b4* and *Ave_delay_line*.

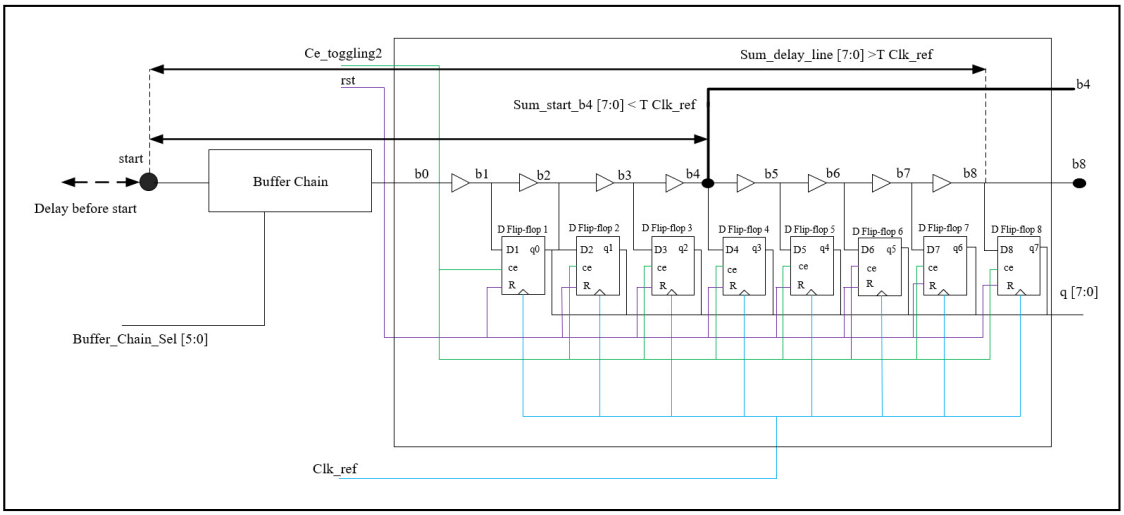

Figure 3.11 Buffer delay line measurement details

As mentioned earlier, each experiment for both even and odd periods is repeated 20 times, and the Diff_merged results obtained for *Ave_start_b4* with *Clk_ref* at 180 MHz and with full switching activity is shown in Figure 3.12. The other frequencies' results are presented in APPENDIX X.

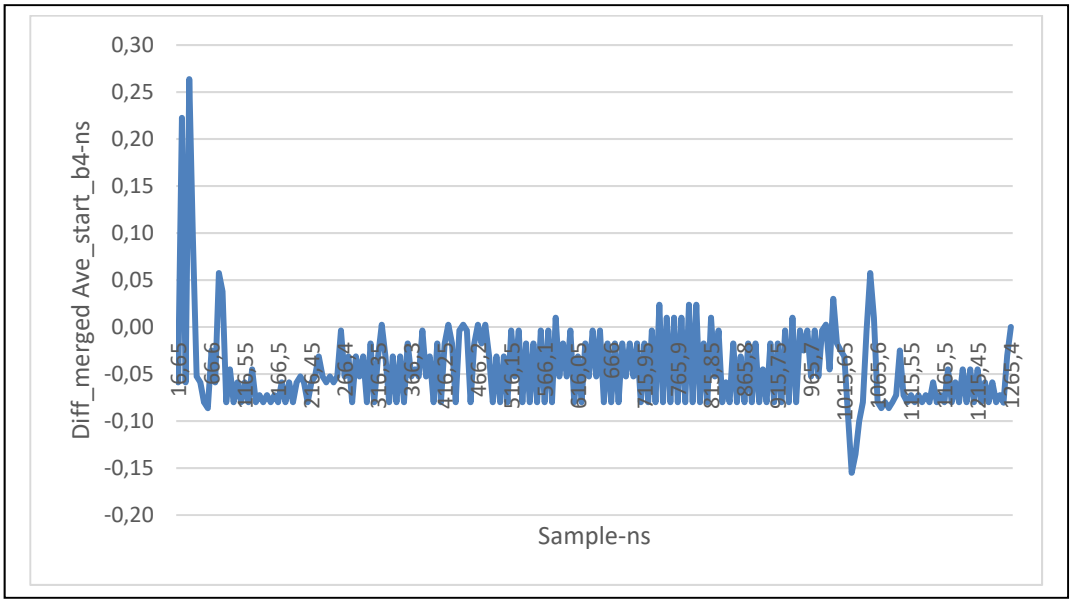

Figure 3.12 Merged results (Diff_merged) for Ave_start_b4 at 180 MHz, full switching activity

Additionally, the Diff_merged results gathered for *Ave_delay_line* at 180 MHz presents in Figure 3.13. The other results are shown in APPENDIX X.

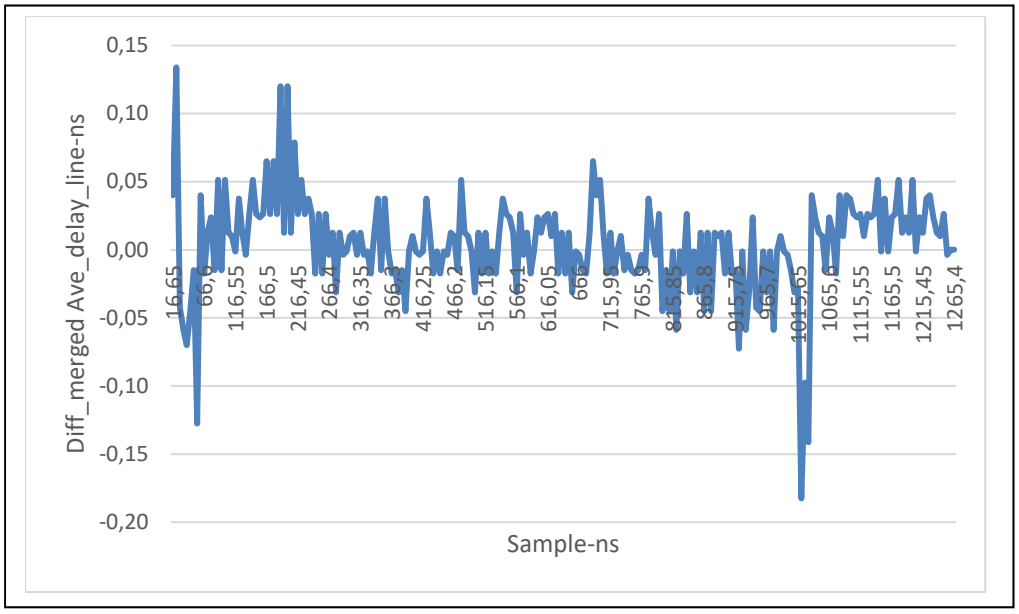

Figure 3.13 Merged results (Diff_merged) for Ave_delay_line at 180 MHz, full switching activity

The results show a pattern similar to one observed with *Q_carry4 [47:0]*, namely a significant delay increase at the beginning of the first transient phase, and a significant delay decrease at the beginning of the second transient phase. Overall, mainly due to the Sum_Circuit block's limited resolution, the results are noisier compared to *Q_carry4 [47:0]* results

The other output we investigated is Diff_merged *Ave_delay_margin*. As presented in Figure 3.14, the results show that the minimum and maximum values are happening after the first and second transient phases, respectively. This behavior is coherent with the previous results as an increase in a combinational delay (*Ave_start_b4*) leads to a decrease in the delay margin. Overall, the results for *Ave_delay_margin* also suffer from the limited resolution of the Sum_Circuit block. The additional results are presented in APPENDIX X.

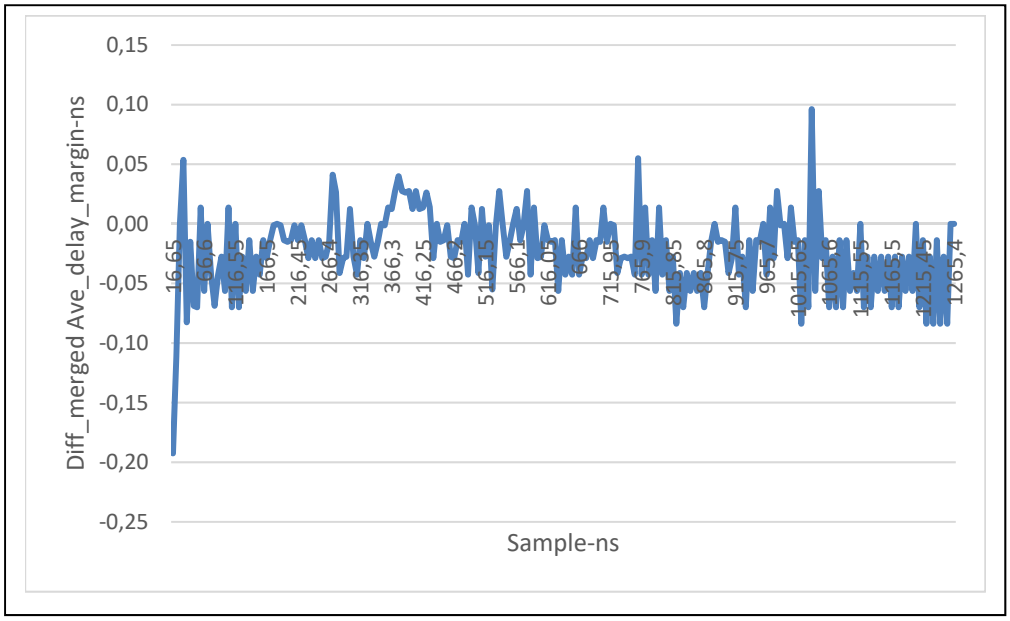

Figure 3.14 Merged results (Diff_merged) for Ave_delay_margin at 180 MHz, full switching activity

Theoretical delay margin calculation method and comparison with the measurement results with the complete timing diagram details are provided in APPENDIX XII.

Table 3.4 shows the delay fluctuation for different frequencies and noise modes for three delay outputs in the buffer delay line. As for Table 3.3, the first and second columns from the left are shown the frequency and the three different noise modes at each frequency. The twelve other columns show the maximum and minimum values for the first and second transient for each output, namely *Ave_start_b4*, *Ave_delay_line*, and *Ave_delay_margin*. All Table 3.4 results are also calculated from Diff-merged results. For example, for *Ave_start_b4* at 180 MHz when no noise is applied, there is no observable fluctuation in the Diff-merged results, and all the values are zero. Furthermore, always at 180 MHz, in the Full Switching Activity (FSA) mode, the first transient maximum and minimum values are 260 ps and -90 ps, respectively.

Table 3.4 Delay fluctuations (ps) for different frequencies and noise mode for Ave_start_b4, Ave_delay_line and Ave_delay_margin (CSAO = Clock Switching Activity Only, FSA = Full Switching Activity)

Frequency (MHz)	Noise mode	Ave_start_b4				Ave_delay_line				Ave_delay_margin			
		First Transient Max Value	First Transient Min Value	2nd Transient Max Value	2nd Transient Min Value	First Transient Max Value	First Transient Min Value	2nd Transient Max Value	2nd Transient Min Value	First Transient Max Value	First Transient Min Value	2nd Transient Max Value	2nd Transient Min Value
100	No	0	0	0	0	0	0	0	0	0	0	0	0
	CSAO	65	-60	0	-20	40	-10	0	-50	15	-25	0	-15
	FSA	75	-75	0	-25	55	-15	0	-55	20	-40	0	-15
120	No	0	0	0	0	0	0	0	0	0	0	0	0
	CSAO	65	-70	0	-30	75	-25	0	-45	35	-35	0	-20
	FSA	80	-80	0	-45	105	-30	0	-50	40	-45	0	-25
140	No	0	0	0	0	0	0	0	0	0	0	0	0
	CSAO	100	-45	55	-75	80	-35	75	-95	60	-175	100	-65
	FSA	140	-60	70	-90	130	-40	80	-110	60	-210	120	-70
160	No	0	0	0	0	0	0	0	0	0	0	0	0
	CSAO	75	-45	90	-30	85	-140	20	-190	55	-100	85	-50
	FSA	90	-50	100	-40	110	-160	30	-240	70	-140	130	-70
180	No	0	0	0	0	0	0	0	0	0	0	0	0
	CSAO	180	-80	50	-120	120	-85	40	-150	50	-140	85	-65
	FSA	260	-90	60	-160	130	-130	40	-180	50	-190	100	-80

In this design, as shown in detail in Figure 3.15, the sum of *Start_b4_pulse, Delay_Margin_pulse* and the delay before the start point is in theory equal to T_{clk_ref}. For example, the average of the odd measured samples at 180 MHz (the maximum frequency for buffer delay line) when full switching activity is applied in the design leads to the following values: *Ave_start_b4* = 2.77 ns (as an approximation of *Start_b4_pulse)* and *Ave_delay_margin* = 0.84 ns (as an approximation of *Delay_Margin_pulse*). Considering that the delay before the start point in this case is about 2 ns, summing the last three values leads to 2.77 + 0.84 + 2 = 5.61 ns $\simeq$ T_{clk_ref} (5.55 ns). This shows that the results are consistent with the design goals.

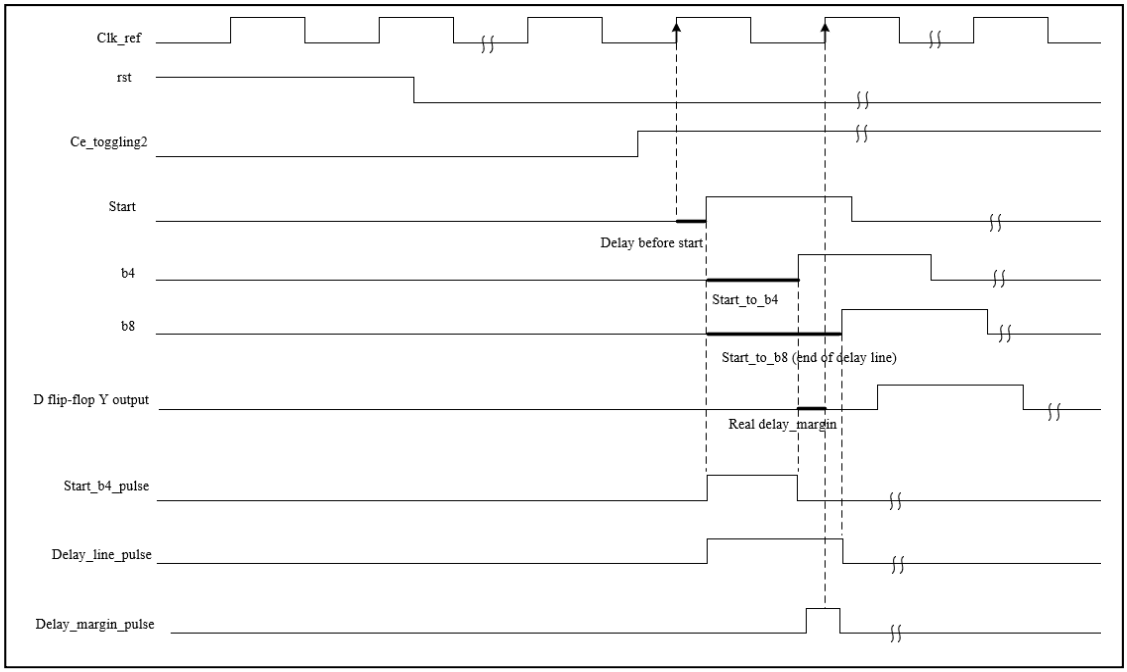

Figure 3.15 Delay's relation in the design

The last output in buffer delay line is *q [7:0]*. As mentioned before, *q [7:0]* is similar to the *Q_carry4 [47:0]* but provides coarser results, and it is post-processed to obtain the Diff_merged *q [7:0]* values. Comparing the Diff_merged *q [7:0]* results with the Diff_merged *Q_carry4 [47:0]* results shows the same pattern. The Diff_merged *q [7:0]* at 180 MHz with full switching activity is shown in Figure 3.16. Additional results are presented in APPENDIX X.

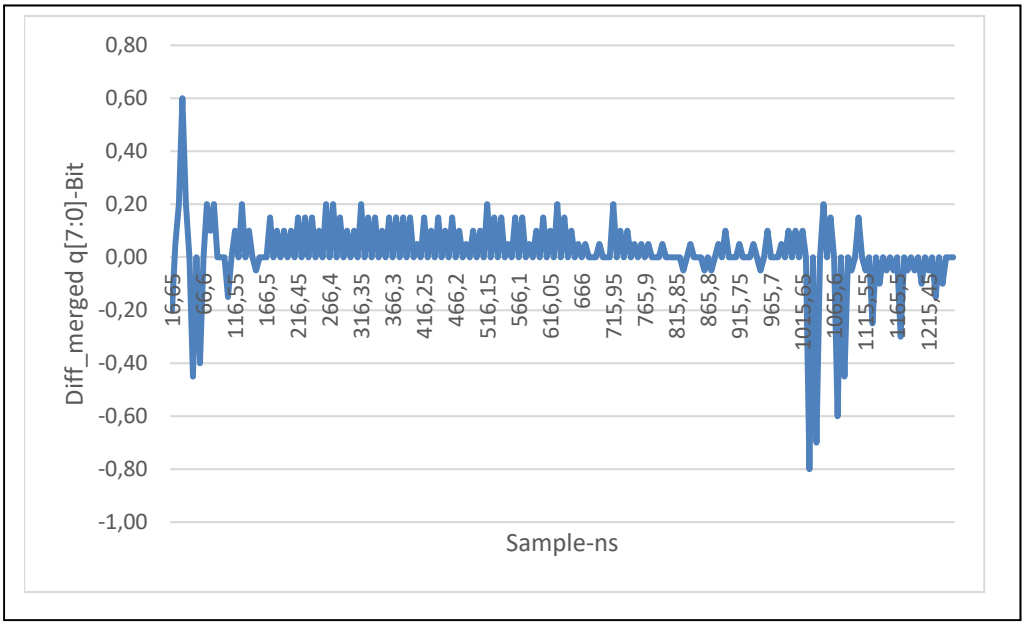

Figure 3.16 Merged results (Diff_merged) for q [7:0] at 180 MHz, full switching activity

Table 3.5 shows the delay fluctuation for different frequencies and noise modes for *q [7:0]* output. The calculation method is similar to the one used for Table 3.3, which shows the *Q_carry4 [47:0]* results. The only difference is that the delay of each buffer is approximately 124 ps and needs to multiply by the Diff-merged values to get the delay fluctuation for *q [7:0]* output.

Table 3.5 Delay fluctuations (ps) for different frequencies and noise mode for q [7:0] (CSAO = Clock Switching Activity Only, FSA = Full Switching Activity)

Frequency (MHz)	Noise mode	First Transient Max Value	First Transient Min Value	2nd Transient Max Value	2nd Transient Min Value
100	No	0	0	0	0
	CSAO	25	-25	6	-19
	FSA	37	-50	12	-31
120	No	0	0	0	0
	CSAO	31	-31	14	-19
	FSA	56	-43	25	-37
140	No	0	0	0	0
	CSAO	68	-56	25	-149
	FSA	99	-105	43	-192
160	No	0	0	0	0
	CSAO	74	-40	43	-118
	FSA	99	-62	62	-149
180	No	0	0	0	0
	CSAO	56	-37	15	-74
	FSA	74	-56	25	-99

3.4 Discussion

3.4.1 Temperature effect on the measurement

The potential temperature effect on the measurements needs to be considered. We performed an experiment to quantify this effect, using the FPGA inner temperature monitoring capabilities. The experiment started at room temperature (23°C) when the board was off. The maximum temperature was expected to occur in the full switching activity mode. After turning on the board, programming the FPGA and adjusting the VIO to have the maximum noise, the board temperature reached around 33°C. Then measurements were taken for about an hour. Figures 2.16 and 2.17 show the board temperature during the measurements taken at 120 and 220 MHz, respectively. Note the results are shown from 0 to 24 minutes, then after an hour.

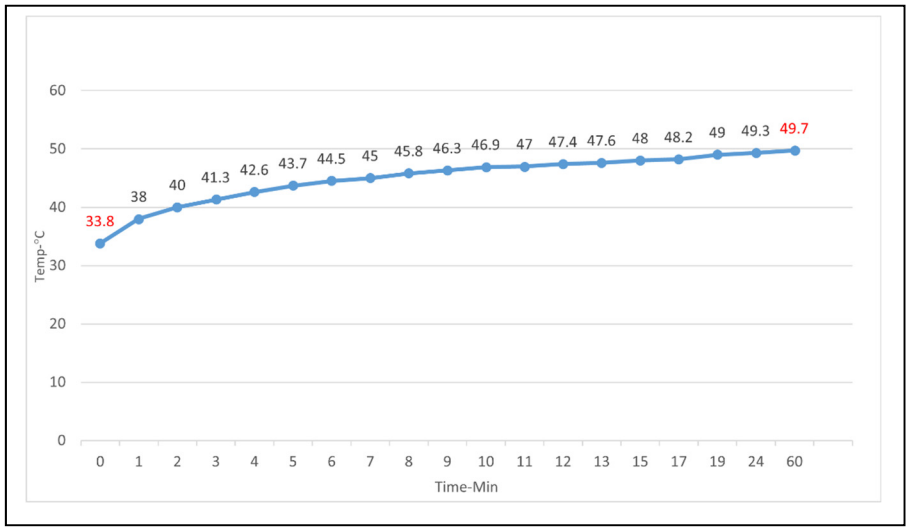

Figure 3.17 Board temperature while experiments for Q_carry4 [47:0] results at 120 MHz

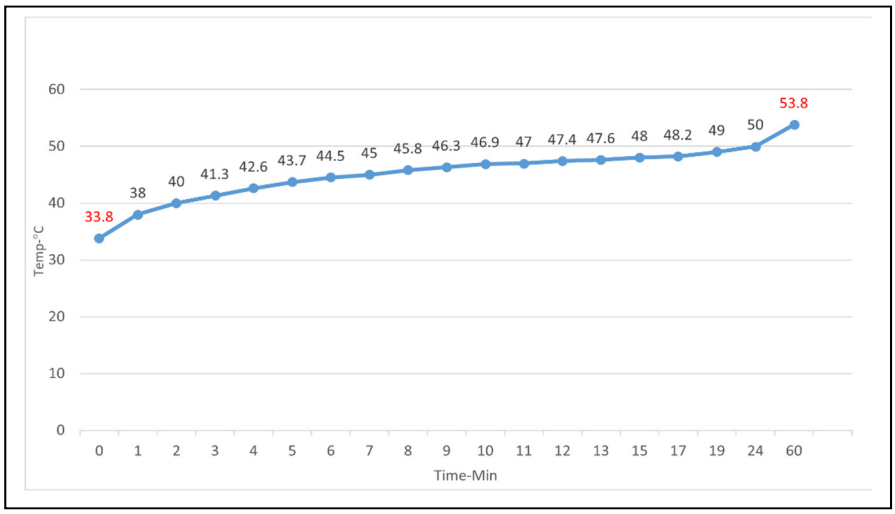

Figure 3.18 Board temperature while experiments for Q_carry4 [7:0] results at 220 MHz

Comparing the results shows that the maximum temperature after one hour at 120 MHz is 49.7°C and at 220 MHz is 53.8°C, which is a 4.1°C increase. Based on that, we could conclude

that, for every 20 MHz increase in frequency, the temperature range increases by 0.82°C, which does not show a significant increase.

Furthermore, examining the *Q_carry4 [47:0]* results could give us a better conclusion in terms of temperature relationship. Tables 3.6 and 3.7 show the correlation coefficients between the samples and the temperature. Nineteen sets of *Q_carry4 [47:0]* results in the full switching activity mode were collected at two different frequencies, namely 120 and 220 MHz. To quantify the correlation between samples and temperature, Pearson's correlation coefficient was applied. This coefficient gives values are between +1 and -1. Coefficient absolute values of +0.70 and higher are usually interpreted as a sign of strong correlation (Mindrila & Balentyne, 2017). Examining the 200 correlation coefficients at 120 MHz and 275 correlation coefficients at 220 MHz shows that the maximum absolute value was 0.4, which means a weak correlation with temperature. Based on these results, we concluded that the temperature effect was negligible during the experiments.

Table 3.6 Correlation coefficients between Q_carry4 [47:0] and temperature at 120 MHz

Sample / Temp -°C	3	4	5	6	7	….	146	147	148	149	150
33.8	24	29	27	28	25	….	27	24	28	24	28
38	25	28	27	29	24	….	28	24	28	24	28
40	25	29	26	28	24	….	27	24	28	24	28
41.3	25	28	28	30	25	….	27	25	27	25	27
42.6	25	30	27	29	24	….	28	24	28	24	28
43.7	25	29	26	29	24	….	27	24	28	24	28
44.5	24	29	27	28	24	….	27	24	27	24	28
45	25	28	26	30	26	….	28	24	28	24	28
45.8	24	30	26	30	24	….	27	24	28	24	28
46.3	23	30	26	28	24	….	28	24	27	24	28
46.9	25	30	26	28	24	….	27	24	27	24	28
47	24	30	27	30	24	….	27	24	28	24	28
47.4	25	28	26	29	25	….	27	24	27	24	28
47.6	25	28	26	30	24	….	27	24	27	24	28
48	24	28	26	28	24	….	28	24	27	24	28
48.2	23	29	27	30	24	….	28	24	28	24	28
49	25	30	26	28	24	….	27	24	27	24	28
49.3	24	28	26	30	25	….	27	24	27	24	28
49.7	24	30	26	29	24	….	27	24	27	24	28
Correlation Coefficient	-0.24	0.16	-0.05	-0.29	0.07	….	0.17	-0.21	0.26	-0.21	0.21

Table 3.7 Correlation coefficients between Q_carry4 [47:0] and temperature at 220 MHz

Sample / Temp -°C	3	4	5	6	7	….	271	272	273	274	275
33.8	40	44	43	45	43	….	39	43	39	43	40
38	39	43	42	44	42	….	39	43	40	43	40
40	40	44	43	45	42	….	39	43	40	43	39
41.3	39	44	42	45	42	….	40	42	40	42	40
42.6	39	44	42	45	42	….	40	42	40	42	40
43.7	39	43	42	44	42	….	40	43	40	42	40
44.5	39	43	42	44	42	….	40	42	40	42	40
45	39	43	42	44	42	….	40	43	40	42	40
45.8	39	44	42	45	42	….	40	42	40	43	40
46.3	40	43	43	44	42	….	40	43	40	42	40
46.9	39	44	42	44	42	….	40	43	40	43	40
47	39	44	42	44	42	….	40	42	40	42	40
47.4	39	44	42	44	42	….	40	42	40	42	40
47.6	39	44	42	45	42	….	40	42	40	43	40
48	39	44	43	44	42	….	40	43	40	42	40
48.2	40	44	42	44	42	….	40	43	40	43	40
49	39	44	43	45	43	….	40	42	40	42	40
50	39	43	42	45	42	….	39	42	39	42	39
53.8	39	44	42	45	42	….	39	42	39	42	39
Correlation Coefficient	-0.36	0.09	-0.24	-0.07	-0.29	….	0.28	-0.40	-0.06	-0.39	-0.26

3.4.2 Results repeatability

In order to verify that the measurements are not drawn in the noise caused by the lack of resolution and that the results were repeatable, ten sets of *Ave_delay_margin* values were collected and analyzed. These sets were collected with the *Clk_ref* is 140 MHz, in the full switching activity mode. The Pearson correlation coefficients between every two sets of samples were calculated. Table 3.8 lists the resulting correlation coefficients.

Table 3.8 Correlation coefficients between measurement sets

Experiment	EX1	EX2	EX3	EX4	EX5	EX6	EX7	EX8	EX9	EX10
EX 1	1									
EX 2	0.83	1								
EX 3	0.89	0.83	1							
EX 4	0.71	0.87	0.78	1						
EX 5	0.79	0.75	0.81	0.75	1					
EX 6	0.76	0.74	0.78	0.72	0.79	1				
EX 7	0.72	0.86	0.74	0.88	0.80	0.70	1			
EX 8	0.71	0.81	0.77	0.89	0.84	0.75	0.82	1		
EX 9	0.85	0.86	0.70	0.70	0.82	0.79	0.80	0.77	1	
EX 10	0.82	0.88	0.79	0.79	0.86	0.78	0.81	0.82	0.88	1

The results' examination shows that the correlation coefficients between every two sets of samples are greater than +0.7, suggesting a rather strong correlation, and showing that the results were repeatable. However, that fact that these coefficient values do not exceed 0.90 also indicates the presence of noise.

3.4.3 Delay margin systematic error

The systematic error in the delay margin estimation can be evaluated using the static timing results. Details of this evaluation are provided in APPENDIX XII. According to this evaluation, this error is a positive offset of 0.064 ns.

Relatively speaking, the systematic error percentage can be expressed :

$$\text{Systematic error percentage} = (\text{error} / \text{Measurement Value}) * 100 \qquad (3.4)$$

Considering that the minimum delay margin at 180 MHz is 0.84 ns, the systematic error would be 7.6 %, which is acceptable.

3.5 Conclusion

The main objective of this chapter was to present and analyze measurements from our experimental setup.
The measurements were done on either the odd or the even samples at different frequencies for both delay lines. The first two measured values (*Q_carry4 [47:0]* and *q [7:0]*) were directly connected to the ILA and post-processed ion an excel file. The last three measured values (*Start_b4_pulse*, *Delay_line_pulse*, *Delay_margin_pulse*) were pulses-shaped outputs needing some processing to get the final pulse width estimation.

To analyze the noise effect on delays, we investigated the *Q_carry4 [47:0]* fluctuations. The results clearly showed a dramatic increase after the first transient, while after the second transient, there is a notable decrease. The experiment results obtained from the other delay line showed a similar but noisier pattern. The overall behavior was also supported by the *Ave_delay_margin* results, which, by definition, reacted in the opposite direction.

Our results also shown that temperature had a negligible impact and that the delay margin systematic error was acceptable.

CONCLUSION AND RECOMMENDATIONS

This research aimed at quantifying the impact of using a popular power consumption reduction technique, namely clock gating, on the delays of surrounding circuits in an FPGA. The design implemented in the FPGA includes two delay lines, some control modules, some processing and storage modules, noise generators emulating clock gating, and an embedded logic analyzer. An experimental setup was developed on a commercial FPGA board. This setup allowed measurements at different clock frequencies without modifying the designed routing.

The most critical challenges in this research project were, first, to get the minimum delay value for each delay element using the most compact placement. We did the implementation many times to find the best place for the delay block and the measurement part of the design. Second, to compare the results at different frequencies, we needed to keep the routing unchanged. In this way, the delay path is identical from one frequency to another. Finally, since the number of components inside the FPGA is fixed, using LUT and register pairs to design a noise generator block had to be considered. Implementing the noise generator block in the case we wanted to design with more than 50% of pairs was impossible.

An analysis of the results showed that, as expected, both enabling and disabling clock gating were inducing transient phases affecting delays. It caused an increase in delays right after enabling clock gating. The sharp increase in the delay value was followed by medium-size fluctuations, corresponding to the first transient phase. Also, there is a decrease in delays right after disabling the noise, followed by medium-size fluctuations, corresponding to the second transient phases. Results also showed small fluctuations in the delay between two transient phases while the delay value reached the steady-state condition. In the absence of direct VDD measurements, we can assume that the shape of the delay curves gives us some hints about the shape of the VDD ones. The sharp increase in the delay value at the beginning of the first transient phase is coherent with the fact that the start of the noise activity induced a sharp

increase in current consumption, which in turn leads to a VDD droop. Results also revealed that temperature had a negligible impact on the results.

The Xilinx FPGA used in this master project is based on a 28 nm technology. Using more recent FPGAs families could be considered for future work. The newer technology could be used to increase the processing speed and also to use the FPGAs s which are bigger in size, to increase the noise in the design.

Moreover, direct measurements of VDD would be an interesting extension of this work. This measurement would help find the relation between the result from this work, which is monitored by the noise effect on delay and the actual noise on VDD. Also, using MUXCY as a delay element in order to increase the resolution would be another case of interest. According to the placement theory of Carry4 logic and the MUXCY inside, we expect to increase the resolution four times when we compare it with the results from this work, at the expense of a more challenging manual placement, as the number of components would increase.

APPENDIX I

ILA BLOCK DESIGN DETAILS

As explained in Chapter 1, the ILA is used to monitor relevant signals of the design. *Sum_delay_margin [7:0]*, *Sum_start_b4 [7:0]*, *Sum_delay_line [7:0]*, *Ce_toggling2*, *Ce-toggling1*, counter *MSB* output, *q [7:0]*, *Q_carry4 [47:0]* are the ILA inputs that we want to monitor. The ILA is clock by the *Clk_ref* signal. The ILA is connected to the JTAG interface from the specific channel, allowing connection to the PC through the Xilinx platform cable. Figure-A I-1 shows the ILA block diagram.

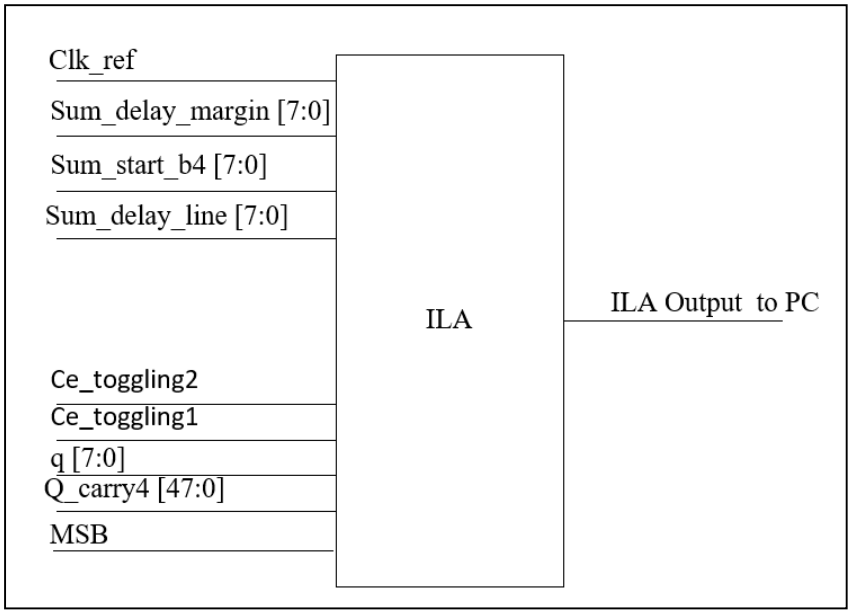

Figure-A I-1 ILA block diagram

APPENDIX II

VIO BLOCK DESIGN DETAILS

As explained in Chapter 1, VIO is used to control some internal FPGA signals, including:

- *Buffer_Chain_Sel [5:0]*, *Carry4_Sel [5:0]*, and *Start_Select* . which are signals controlling the delay block, which is discussed in subsection 2.2.1.
- *Noise_en1*, *Noise_en2*, *Ce_1*, and *Ce_2*, which are signals used to control the noise generator block, which is explained in detail in subsection 2.2.3.

The only input in this block is *Clk_ref*. Figure-A II-1 shows the VIO block diagram.

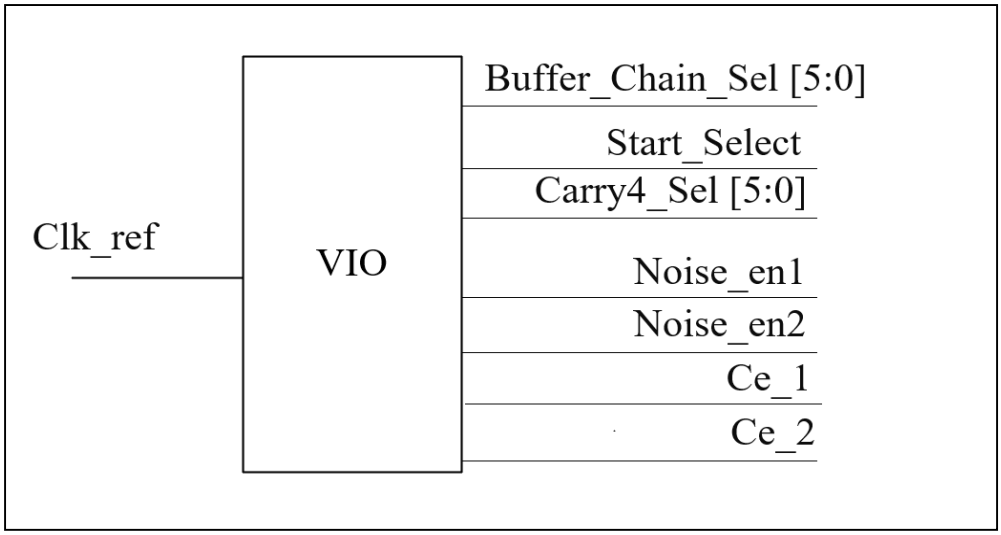

Figure-A II-1 VIO block diagram

APPENDIX III

CLOCK GENERATOR DESIGN DETAILS

Figure-A III-1 illustrates the clock generator block diagram. As explained in Chapter 1, the system clock is a 200 MHz differential input, and an IBUFDS is needed to get the single-ended clock. The IBUFDS output is connected to the reference clock input of the two PLLs and two MMCMs inside the clock generator block. The other input of this block, namely *Reset_out*, is provided by the reset debounce block (described in the following APPENDIX). *Clk_ref_2x* is the *Clk_ref* multiplied by two is used to create *Rst_en* and *FIFO_Wr_en* signals in the Sum_Circuit block.

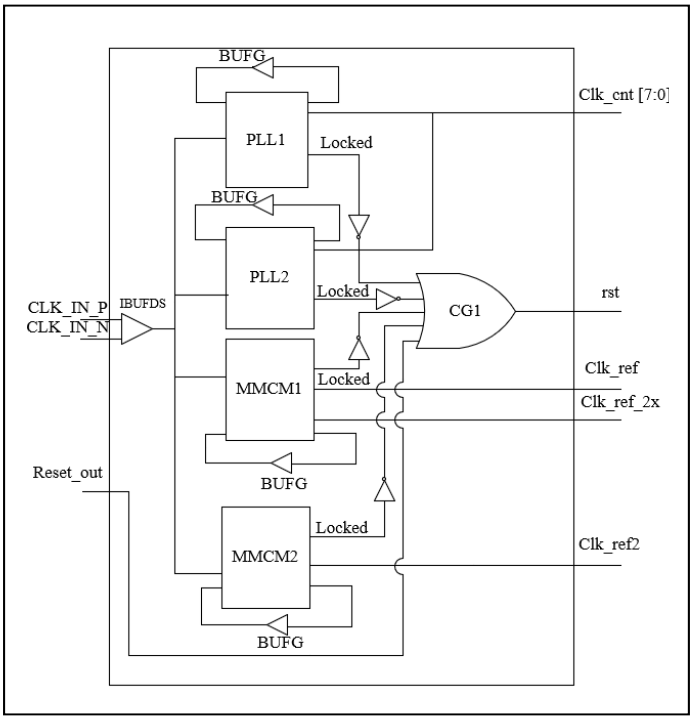

Figure-A III-1 Clock generator block diagram

The PLL1 and PLL2 outputs are used to create *Clk-cnt [7:0]*, eight clock signals whose frequency is equal to 450 MHz but with eight different phases to increase the measurement accuracy.0,90,180,270 degrees from PLL1 and 45,135, 225,315 degrees from PLL2. MMCM1 creates *Clk_ref* and *Clk_ref_2x*, which are used to create the required pulses in the design and measure their width. MMCM2 creates *Clk_ref2*, which is the perfect copy of *Clk_ref* to be used in the noise generator block. From each PLL and MMCM, the locked output connects to a NOT gate. The output of NOT gates connects to an OR gate CG1. Finally, the CG 1 output generates the *rst* signal.

APPENDIX IV

RESET DEBOUNCE BLOCK DESIGN DETAILS

The entire design needs to be reset manually after each experiment. To reset the system manually, we utilize the pushbutton sw7 on the board, which can lead to bounces affecting the signal. To ensure that the *reset_out* signal is clean and without unwanted bounces, we created the reset generator block. Figure-A IV-1 shows the reset debounce top level block diagram. *Reset_out* is a clean signal which goes inside the clock generator block and works as the fifth input of the OR gate CG1. Whenever we want to reset the system manually, we will be able to do that by pressing the pushbutton sw7. Figure-A IV-2 shows the expanded reset debounce block diagram.

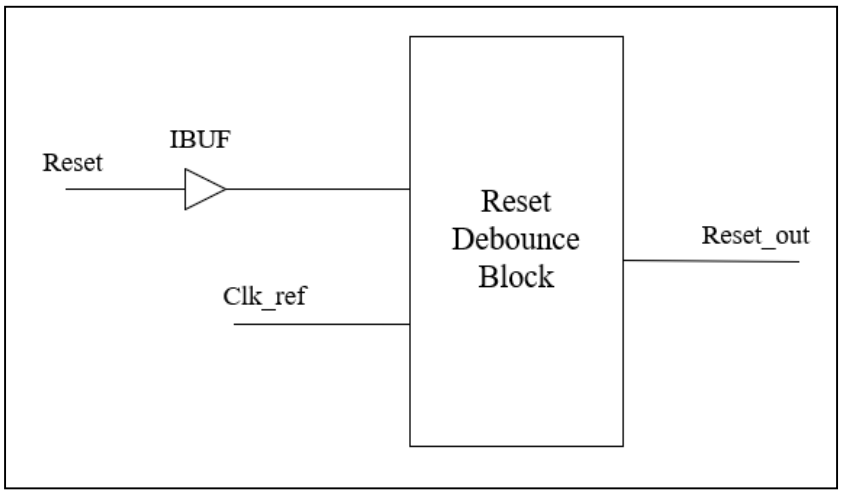

Figure-A IV-1 Reset debounce top level block diagram

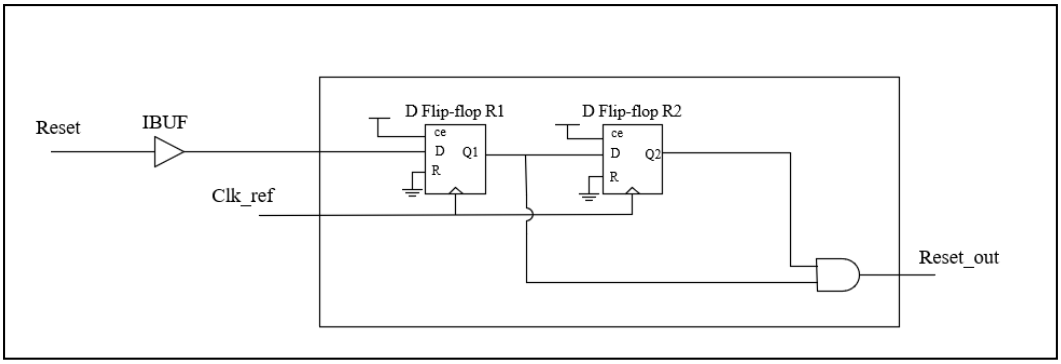

Figure-A IV-2 Reset debounce expanded block diagram

The IBUF output is the block input. To make a clear signal, the outputs of two D flip-flops R1 and R2 are connected to an AND gate. The output of the AND gate is the clean *reset_out* signal. Figure-A IV-3 displays the reset debounce block waveform in detail.

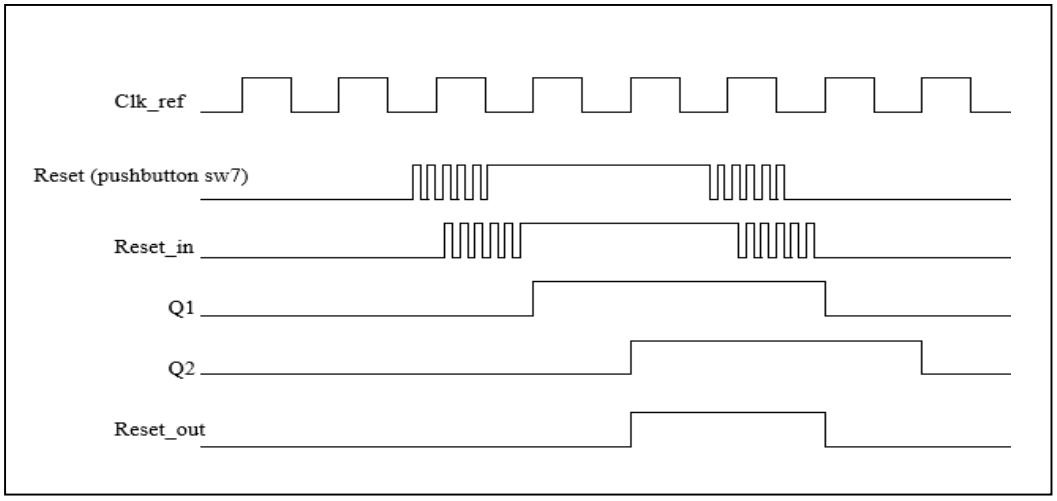

Figure-A IV-4 Reset debounce block timing diagram

APPENDIX V

CLOCK ENABLE DEBOUNCE BLOCK DESIGN DETAILS

This block is used to create, from an external signal (*CE*) coming from the pushbutton sw5 on the board, a clean signal, *FSM_Start* , which is the counter clock enable in the Finite State Machine (FSM) block. It is similar to the reset bounce block, with an additional reset signal (*rst*). *Clk_ref*, *rst*, and *CE* after going into an IBUF are the inputs, and the *FSM_Start* signal is the output in the clock enable debounce block. Figure-A V-1 shows the clock enable debounce top level block diagram.

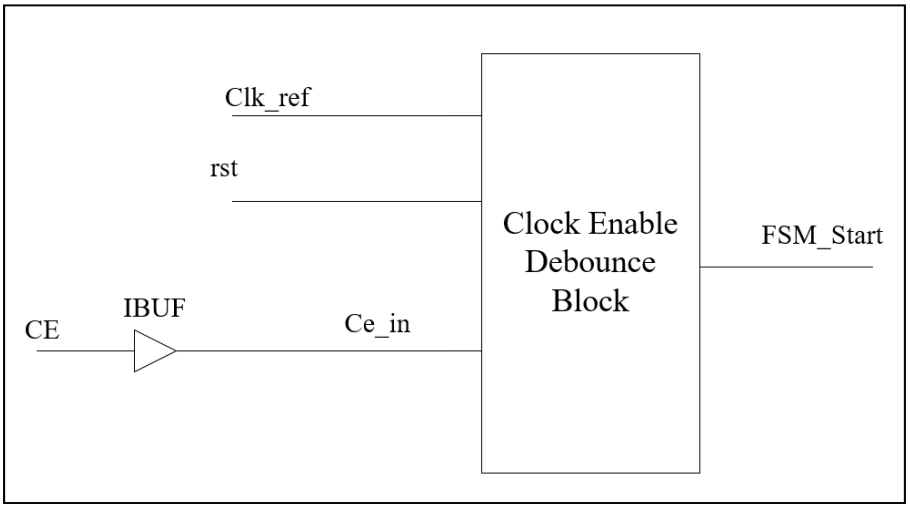

Figure-A V-1 Clock enable debounce top level block diagram

Figure-A V-2 show the clock enable debounce expanded block diagram.

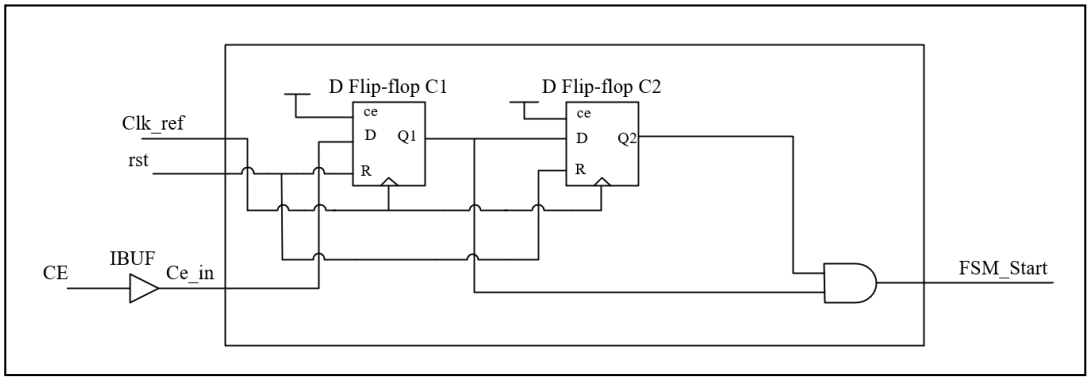

Figure-A V-2 Clock enable debounce expanded block diagram

The clock enable debounce includes two D flip-flops and an AND gate. To create the clean start signal without bounces, the outputs of two D flip-flops C1 and C2, connect to the AND gate. The output of the AND gate is the clean *FSM-Start* signal. Figure-A V-3 displays the *FSM-Start* generator timing diagram in detail.

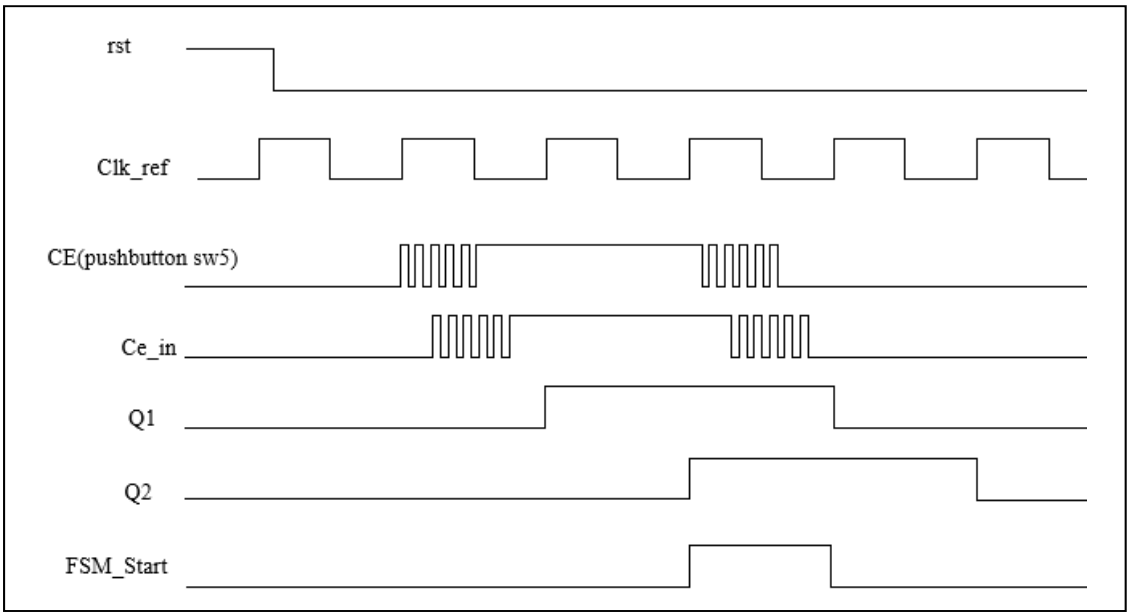

Figure-A V-3 Clock enable debounce block timing diagram

APPENDIX VI

FSM BLOCK DESIGN DETAILS

This appendix describes the FSM block, used to create *Ce_toggling1* and *Ce_toggling2* signals. *FSM_Start*, *Clk_ref*, and *rst* are three inputs in this block. *Ce_toggling1* is the noise block clock enable, and *Ce_toggling2* is the Sum_Circuit and delay blocks clock enable. Figure-A VI-1 shows the FSM block diagram.

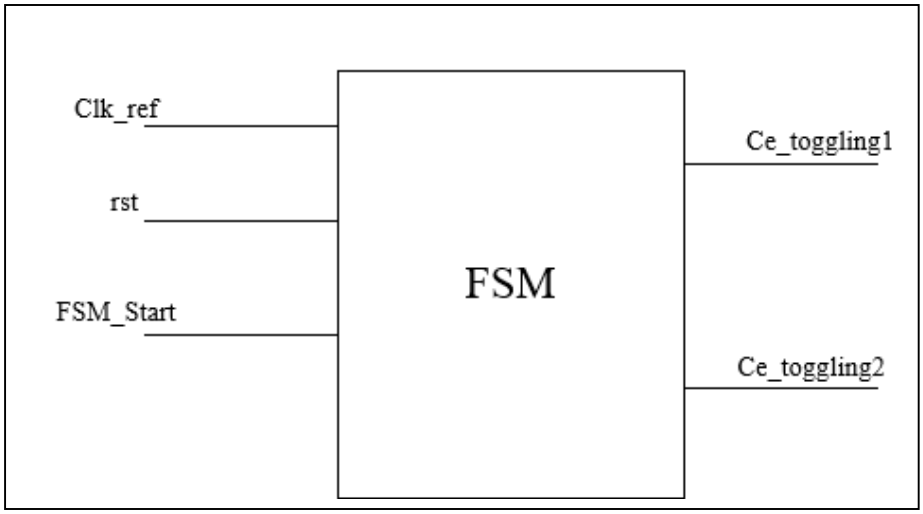

Figure-A VI-1 FSM block diagram

There is a 9-bit counter inside the FSM block. The *FSM_Start* pulse is used as a clock enable of the counter. After a reset, the counter output remains at zero as long as the *FSM_Start* is equal to zero, causing the *Ce_ toggling1* and *Ce_toggling2* signals to be zero. When the *FSM_Start* is enabled, from the next *Clk_ref* rising edge counter starts to count for one microsecond, and the *Ce_toggling1* and *Ce_toggling2* remains at zero. After this first 1-microsecond delay, *Ce_toggling1* becomes one for the next microsecond, and *Ce-_oggling2* becomes one for the next 1.25 microsecond. Figure-A VI-2 shows the FSM timing diagram.

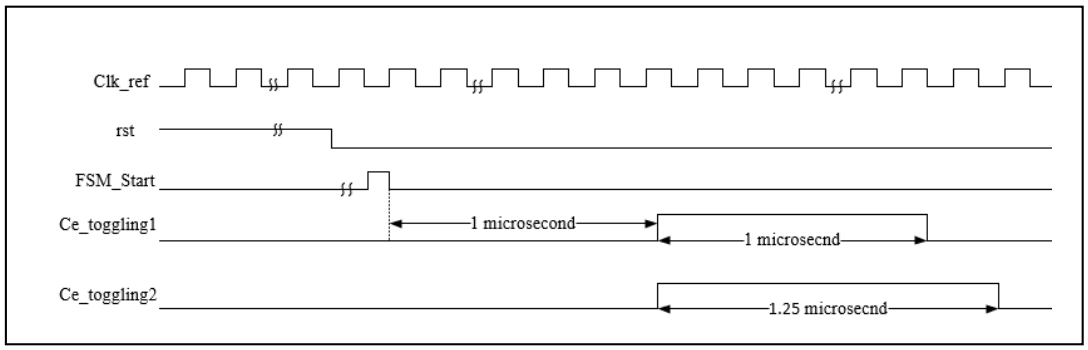

Figure-A VI-2 FSM timing diagram

APPENDIX VII

COUNTER BLOCK DESIGN DETAILS

This appendix describes the counter block used to divide the frequency of the *Clk_ref* signal by a factor of 32, such that it can properly be sampled by the ILA and allows us to make sure that the correct *Clk_ref* is used in the measurements. This block simply contains a 5-bit counter. *Clk_ref* and *rst* are two inputs connected to the counter clock and SCLR input of the counter, respectively. MSB is the counter output that is connected to the ILA. Figure-A VII-1 shows the counter block diagram.

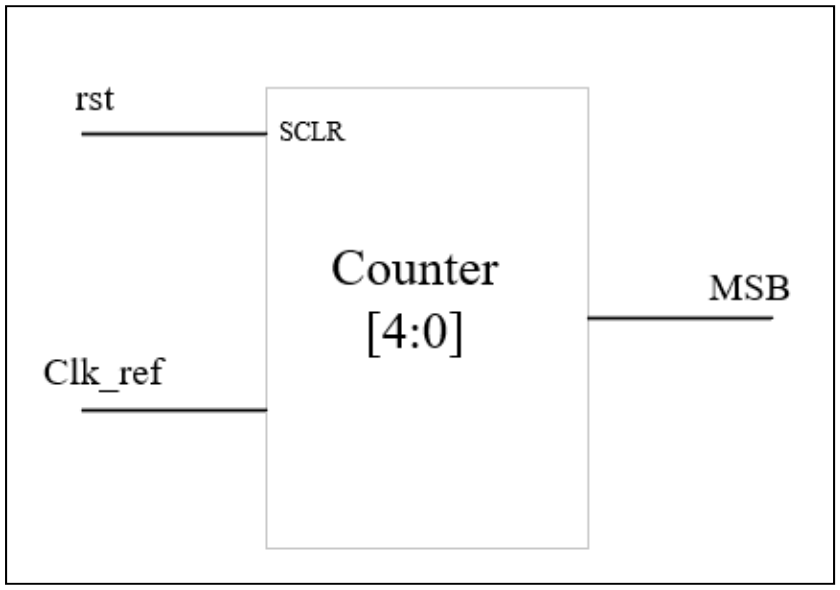

Figure-A VII-1 Counter block diagram

APPENDIX VIII

M_COUNTER_DELAY_MARGIN_I0 DESIGN DETAILS

Figure-A VIII-1 illustrates the M_counter_delay_margin_i0 block diagram design details.

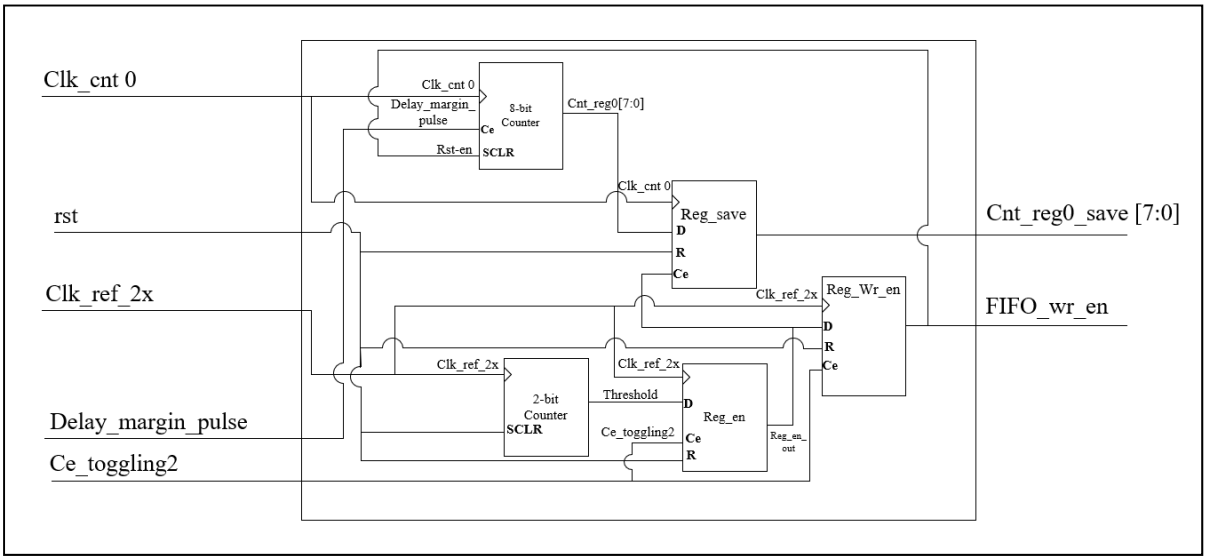

Figure-A VIII-1 M_counter_delay_margin_i0 block diagram

There is a 2-bit counter inside each M_counter, which works with *Clk_ref_2x*. In this part *Clk_ref_2x* is used instead of *Clk_ref* to have more clock rising edges between two pulses when a faster reaction is needed. The 2-bit counter output (generically named *threshold* in the previous figure) is called *threshold2* for Sum_Cir_1 and *threshold3* for Sum_Cir_2 and Sum_Cir_3. The 2-bit counter output later is shifted for two *Clk_ref_2x* clock cycles by using two registers Reg_en and Reg_Wr_en. The Reg_Wr_en output is used as a *FIFO_Wr_en* and SCLR inputs in the 8-bit counter. The other inputs in the 8-bit counter are *Clk_cnt0* and *Delay_margin_pulse*. After finishing the pulse width counting process, the output, which is the *Cnt_reg0 [7:0]* value, will be saved by another register, Reg_save. The Reg_save output,

that is *Cnt_reg0_save [7:0]*, will be stored by a FIFO in the next step. Figure 2.22 illustrates the M_counter_delay_margin_i0 timing diagram.

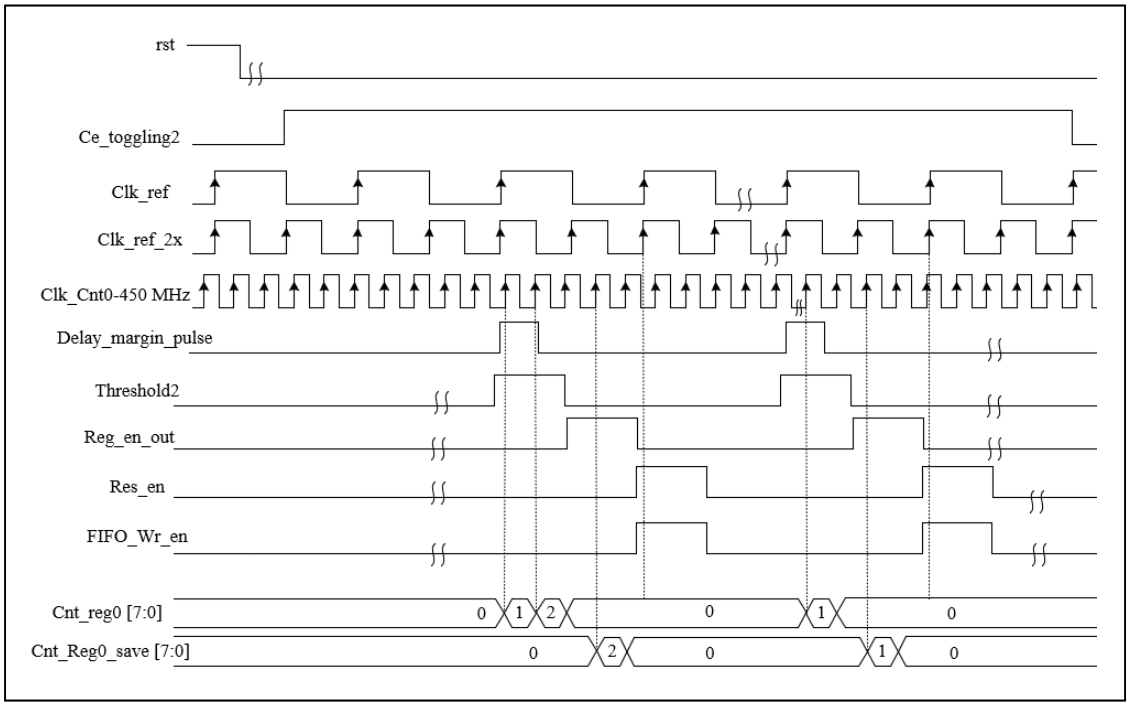

Figure-A VIII-2 M_counter_delay_margin_i0 timing diagram

APPENDIX IX

CARRY4 LOGIC DELAY VALUE ESTIMATIONS

As shown in Table-A IX-1, there are different possibilities to find the carry4 logic delay. This estimation is performed when full switching activity is applied by comparing the results obtained from two different clock frequencies but with the same number of delay elements in the programmable part of the delay line. In Table-A IX-1, the first two columns (from the left) show the two clock frequencies (Frequency1 and Frequency2, where Frequency1 > Frequency2), and the next two the related clock periods. The fifth column lists the increase in clock period from the fastest (Frequency1) to the lowest (Frequency2) clock frequency. The sixth column indicates the selected number of delay elements in the programmable part of the delay line. The seventh and eighth columns give the number of remaining 0's in the delay line for Frequency1 and Frequency2, respectively, while the tenth column (F1-F2) is the difference between these two last numbers. Finally, the last column provides the estimation of the carry4 logic delay, obtained by dividing the value of the fifth column (increase in clock period) by the value of the tenth column.

Table-A IX-1 Different possibilites to estimate the carry4 logic delay

Frecueny1-MHz	Frecueny2-MHz	period1-ns	period2-ns	Increased Time-ns	Fixed Carry4 Chain Value	First value	2nd value	F1-F2	Carry4 Delay
220	200	4.55	5	0.45	1	40	29	11	0.0409
220	180	4.55	5.56	1.01	1	40	17	23	0.0439
220	160	4.55	6.25	1.7	1	40	4	36	0.0472
200	180	5	5.56	0.56	1	29	19	10	0.0560
200	160	5	6.25	1.25	1	29	4	25	0.0500
200	180	5	5.56	0.56	2	31	19	12	0.0467
200	160	5	6.25	1.25	2	31	8	23	0.0543
200	160	5	6.25	1.25	4	36	8	28	0.0446
200	160	5	6.25	1.25	7	39	13	26	0.0481
180	160	5.56	6.25	0.69	1	17	4	13	0.0531
180	160	5.56	6.25	0.69	2	19	6	13	0.0531
180	160	5.56	6.25	0.69	5	21	11	10	0.0690
180	160	5.56	6.25	0.69	10	32	18	14	0.0493
180	160	5.56	6.25	0.69	14	40	23	17	0.0406
160	140	6.25	7.14	0.89	25	41	29	12	0.0742
160	120	6.25	8.33	2.08	25	41	11	30	0.0693
140	120	7.14	8.33	1.19	19	24	6	18	0.0661
140	120	7.14	8.33	1.19	25	29	11	18	0.0661
140	120	7.14	8.33	1.19	31	36	19	17	0.0700
140	120	7.14	8.33	1.19	36	44	24	20	0.0595

For example, in the first result row, in Table 3.3, two frequencies, namely 220 and 200 MHz, are compared when the adjustable part is fixed to be one carry4 logic. The clock period is increased from 4.55 to 5 ns while the number of remaining 0's in *Q_carry4 [47:0*] is increased from 29 to 40 bits. When we divide the increased time (0.45 ns) by the increased value (11 bits), we have approximately 41 ps delay for each carry4 logic in this specific case. The average of all the possibilities gives a delay value of the 55 ps for each carry4 logic, which is a very short delay.

APPENDIX X

Q_CARRY4 [47:0] AT DIFFERENT FREQUENCIES

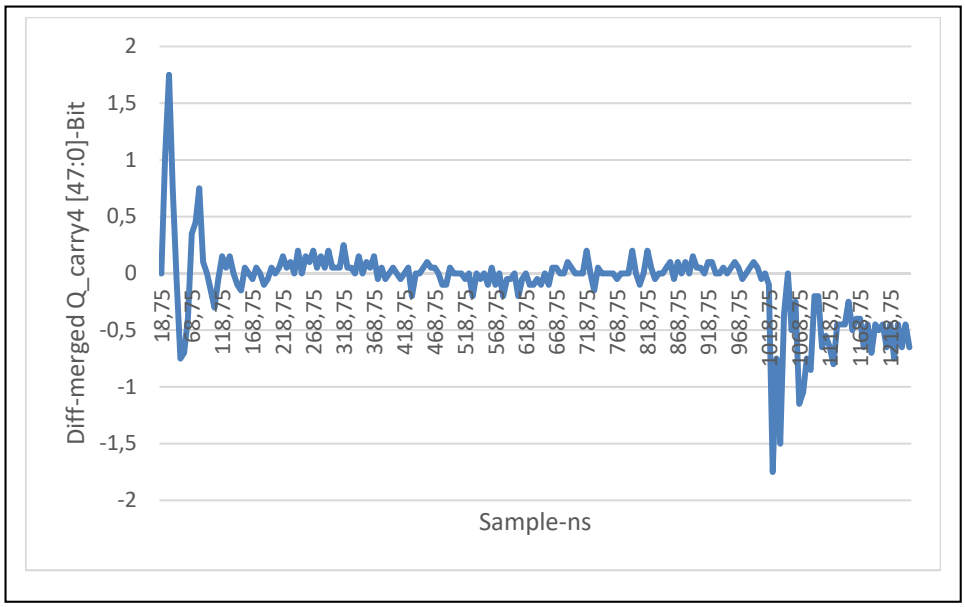

Figure-A X-1 Merged results (Diff_merged) for Q_carry4 [47:0] at 160 MHz, full switching noise activity

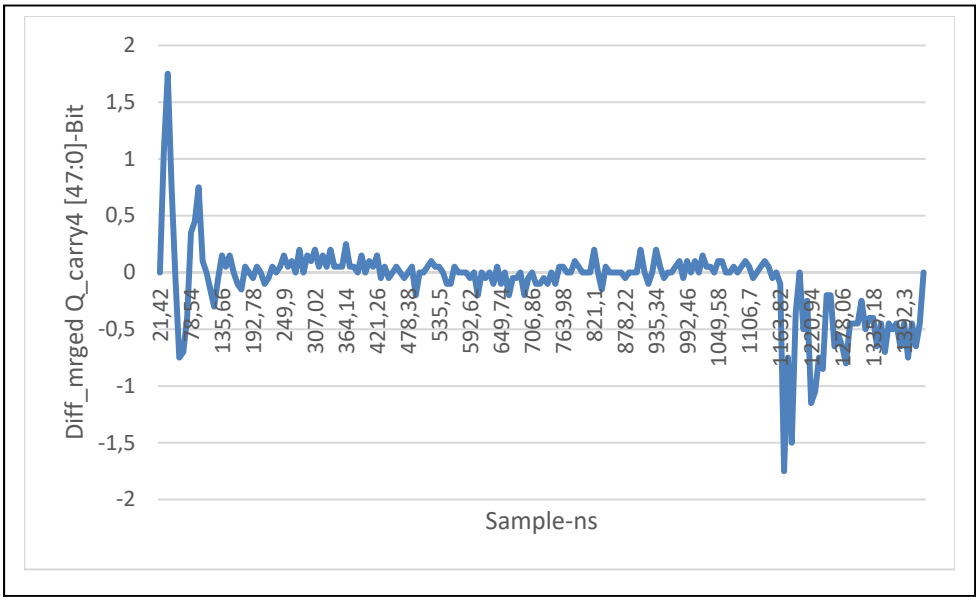

Figure-A X-2 Merged results (Diff_merged) for Q_carry4 [47:0] at 140 MHz, full switching noise activity

APPENDIX XI

BUFFER DELAY LINE RESULTS AT DIFFERENT FREQUENCIES

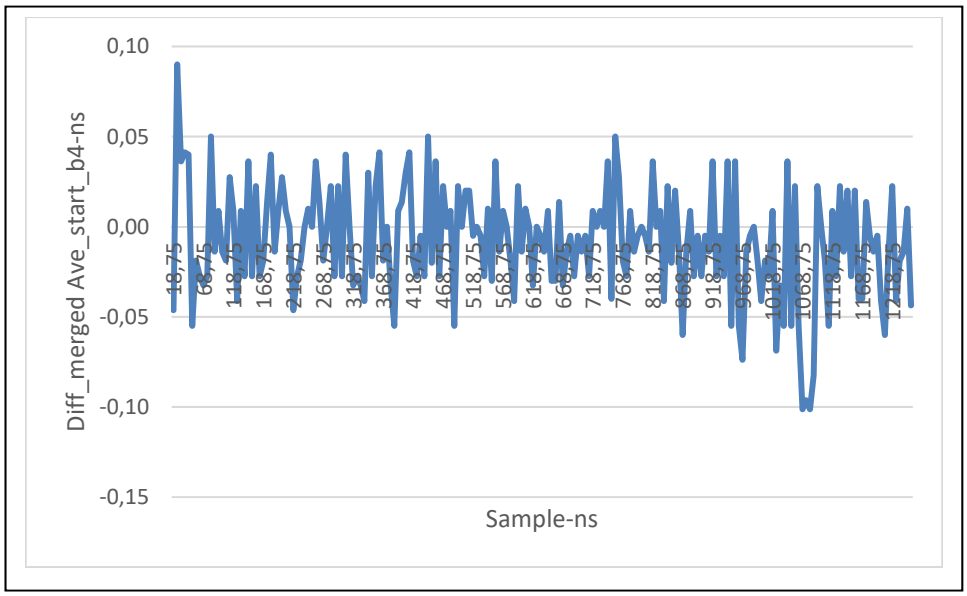

Figure-A XI-1 Merged results (Diff_merged) for Ave_start_b4
at 160 MHz, full switching activity

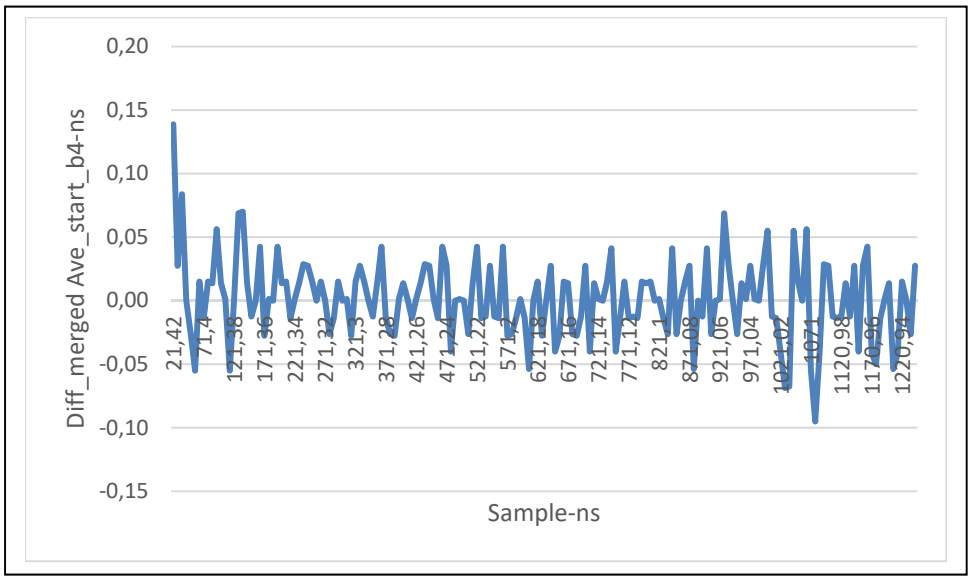

Figure-A XI-2 Merged results (Diff_merged) for Ave_start_b4
at 140 MHz, full switching activity

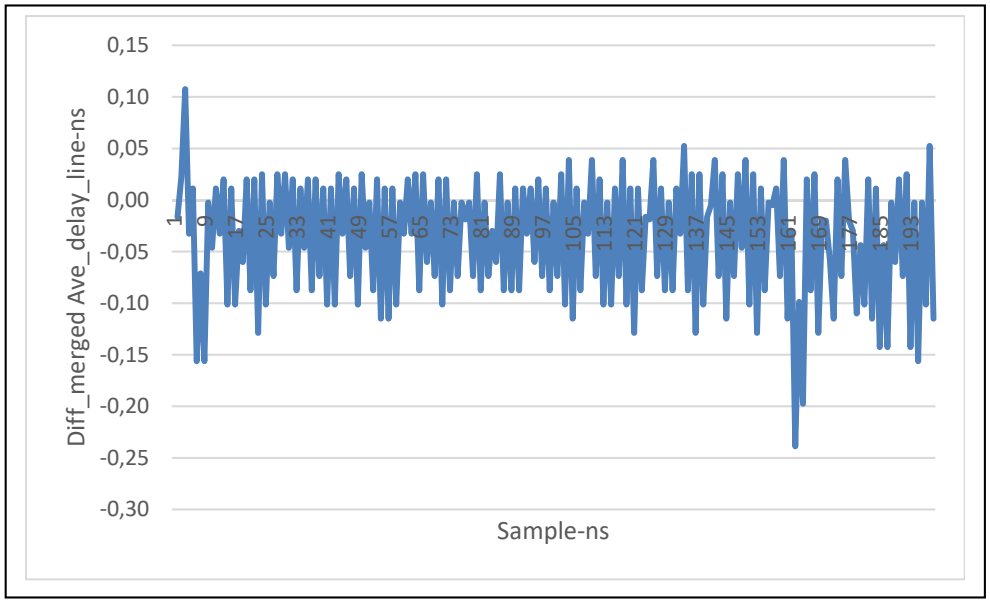

Figure-A XI-3 Merged results (Diff_merged) for Ave_delay_line at 160 MHz, full switching activity

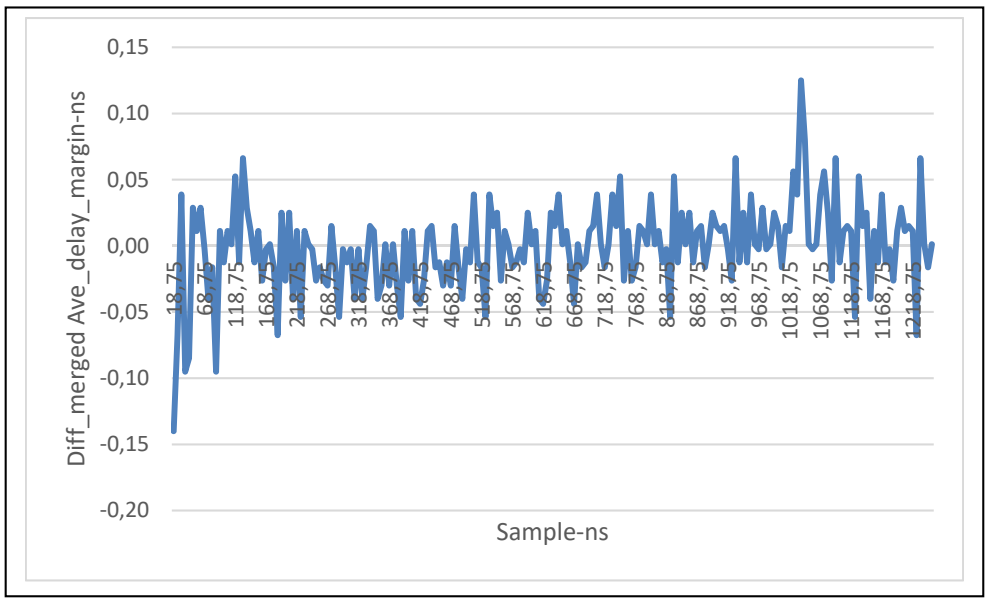

Figure-A XI-4 Merged results (Diff_merged) for Ave_delay_margin at 160 MHz, full switching activity

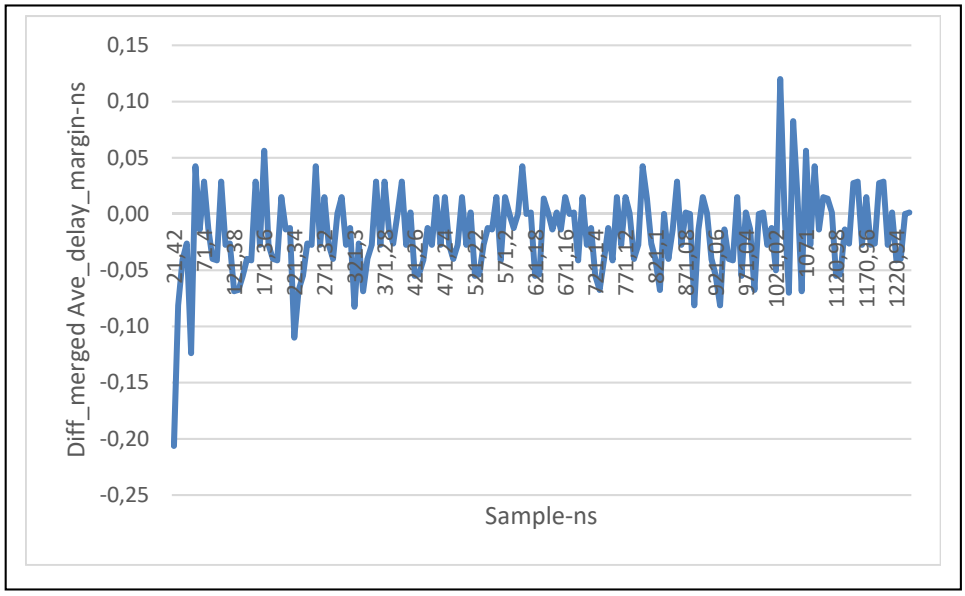

Figure-A XI-5 Merged results (Diff_merged) for Ave_delay_margin at 160 MHz, full switching activity

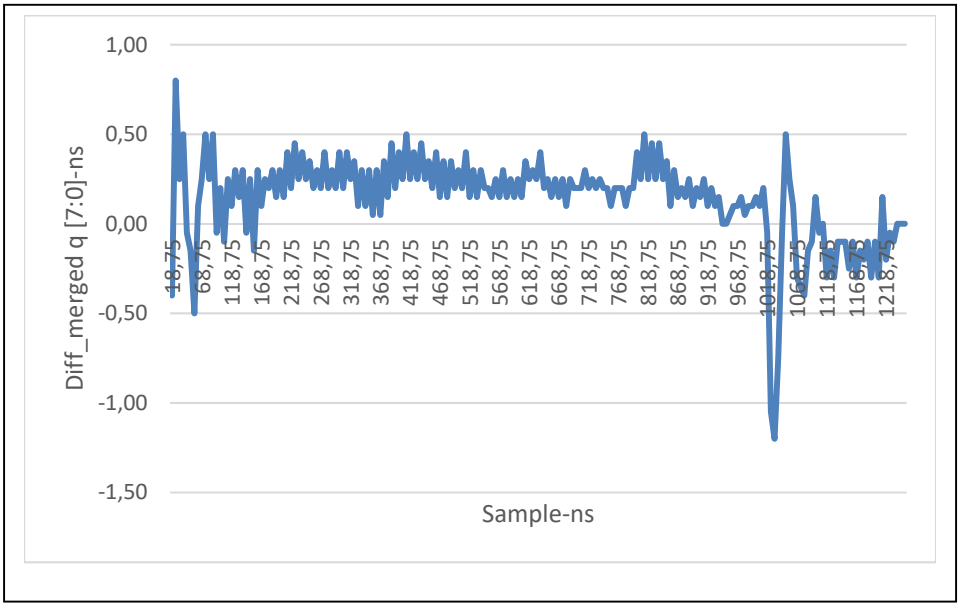

Figure-A XI-6 Merged results (Diff_merged) for q [7:0] at 180 MHz, full switching activity

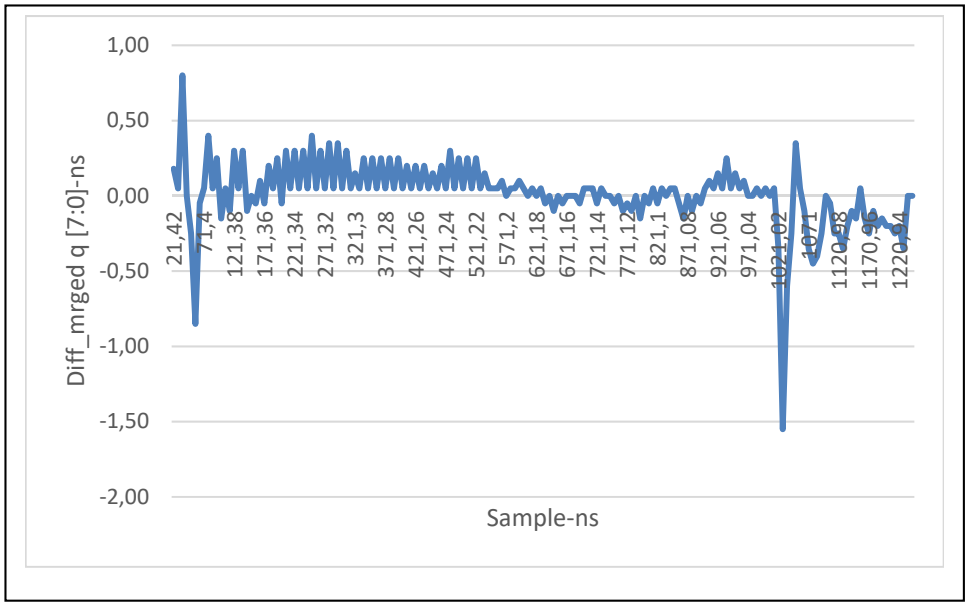

Figure-A XI-7 Merged results (Diff_merged) for q [7:0]
at 180 MHz, full switching activity

APPENDIX XII

DELAY BEFORE START POINT

The pulses are measured from the start point. The delay between the FF launching the rising transition and the start point has no impact, except when comes the time to:

- Compare the sum of the *Ave_start_b4* and *Ave_delay_margin* values to the *Clk_ref* period. Without this delay, both should be equal.
- Compare the *Ave_delay_line* value to the *Clk_ref* period; the former is supposed to be larger than the latter. Based on the static timing analyzer, the *Ave_delay_line* is equal to 8.8 ns when the *Clk_ref* is 100 MHz.

The delay before the start point for both odd and even periods needs to be calculated and taken into consideration in these situations.

Figure-A XII-1 shows the delay details before the start point.

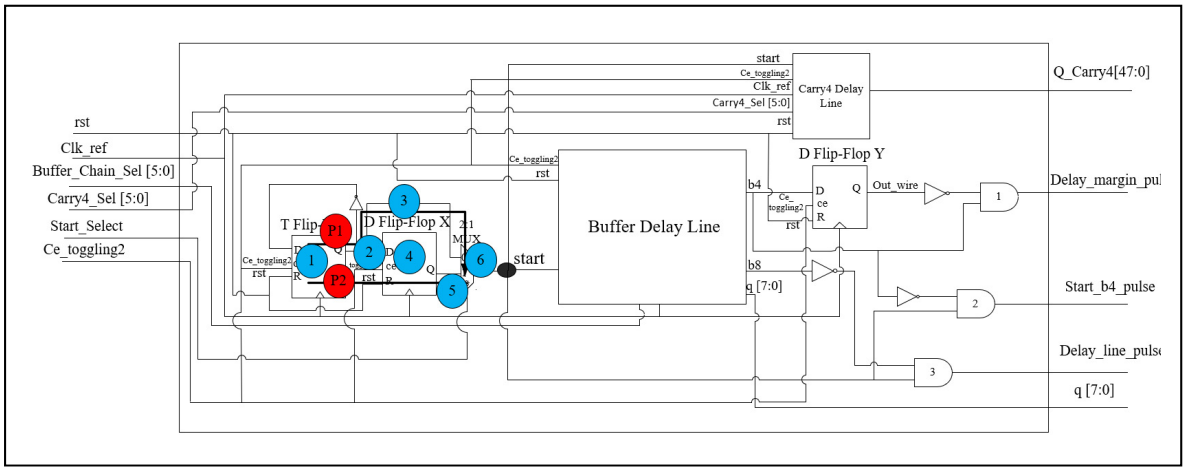

Figure-A XII-1 Delay details before the start point

P1 and P2 are the two paths before the start point, used for the odd and even periods, respectively. There are six delay elements. Table-A XII-1 presents the delay element values before the start point, according to the static timing analyzer.

Table-A XII-1 Delay elements values before the start point

Number	Delay Element	Value/ ns	Path
1	Flip-flop	0.518	P1 and P2
2	Net delay	0.354	P2
3	Net delay	0.485	P1
4	Flip-flop	0.518	P2
5	Net delay	0.449	P2
6	2:1 multiplexer	0.369	P1 and P2

When we add each relevant path delay element values together, the paths delay would be :

$$P1 = 0.518+ 0.485+ 0.369 = 1.372 \text{ ns} \quad \text{(A XII.1)}$$

$$P2 = 0.518+ 0.354+ 0.518+ 0.449+ 0.369 = 2.208 \text{ ns} \quad \text{(A XII.2)}$$

According to the previous example (Ave_delay_line = 8.8 ns), when these two delay values add to the even and odd measurement results, the *Ave_delay_line* = 8.8 + 1.372= 10.172 ns for odd periods and *Ave_delay_line* = 8.8 + 2.208= 11.008 ns for even periods, which both are more than 10 ns, and they meet the desired timing specifications. During the entire measurements, these constant values must be added to the experiment's results. Note that the difference between the odd and even paths before the starting point should not affect the merge the odd and even measurement sets, as there are expressed in a differential way with respect to a reference value. As the error due to the difference between the odd and even paths will affect in a similar fashion both each measurement and the reference value, the error will be canceled out.

APPENDIX XIII

DELAY MARGIN CALCULATION METHOD AND THE SYSTEMATIC ERROR

By definition, the delay margin is the time between the selected point representing the end of the combinational path (in our case *b4*) and the next clock rising edge. Figure-A XIII-1 shows the delay margin in theory.

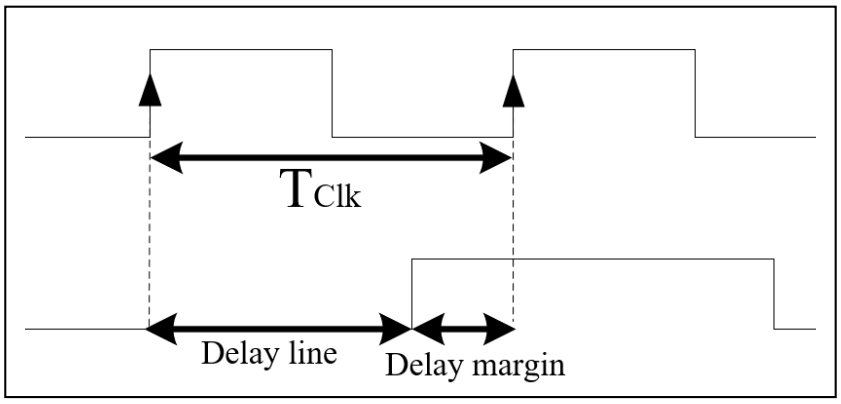

Figure-A XIII-1 Delay margin in theory

In our design, the rising edge on b4 is captured and processed to create a pulse whose width is a good estimate of the delay margin. As shown in Figure-A XIII-2 the real delay margin in our case would be:

Real delay margin = T_{clk_ref} – (Delay before start point +Start_to_b4) = T_{clk_ref} – clk_to_b4 (A XIII-1)

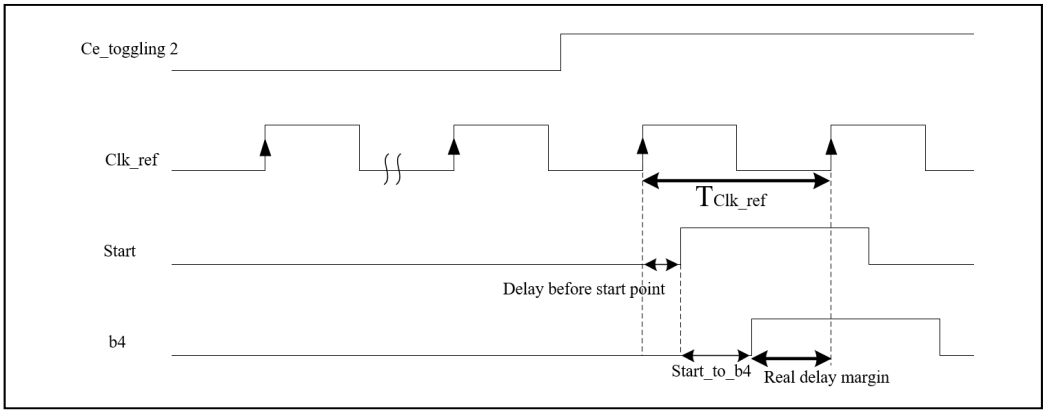

Figure-A XIII-2 Real delay margin timing diagram in the design

On the other hand to quantify the delay margin estimate based on the static timing we need to find each delay element as shown in Figure-A XIII-3 and 4.

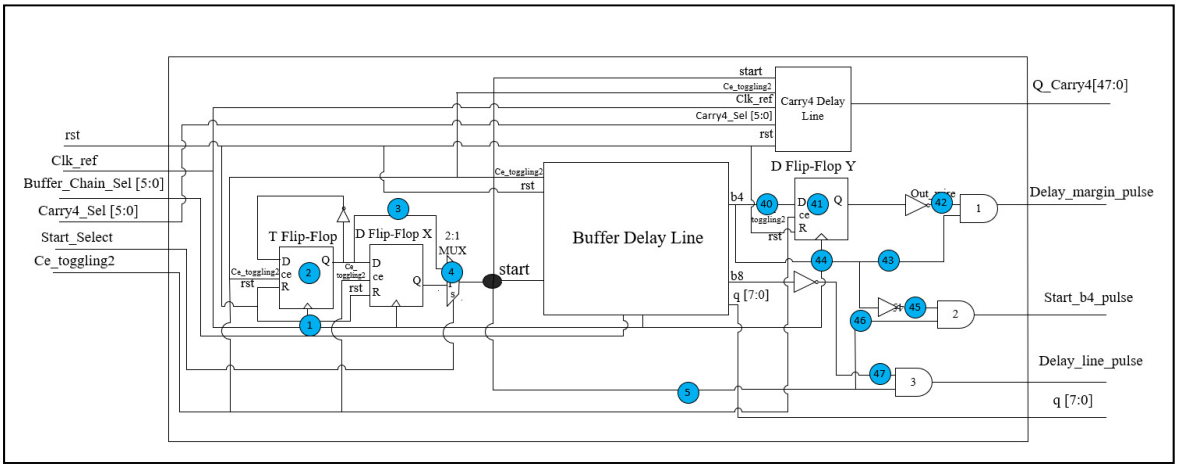

Figure-A XIII-3 Delay block static timing details

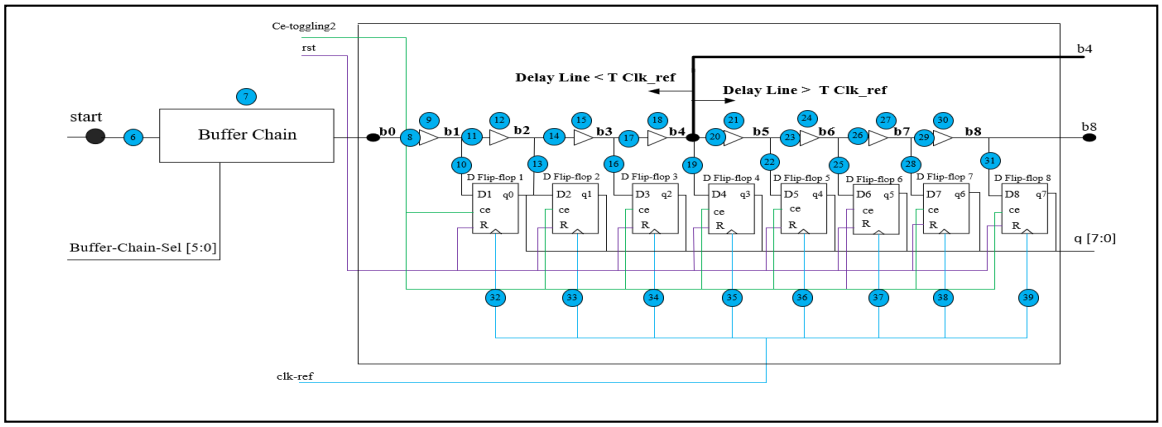

Figure-A XIII-4 Buffer delay line static timing details

The *Delay_margin_pulse* is created by AND gate one of two signals, namely *b4* and its previous value, inverted. To find the estimated delay margin formula from the timing diagram shown in Figure-A XIII-5 and 6, we would have:

$$\text{Estimated static delay margin} = A - B \qquad \text{(A XIII-2)}$$

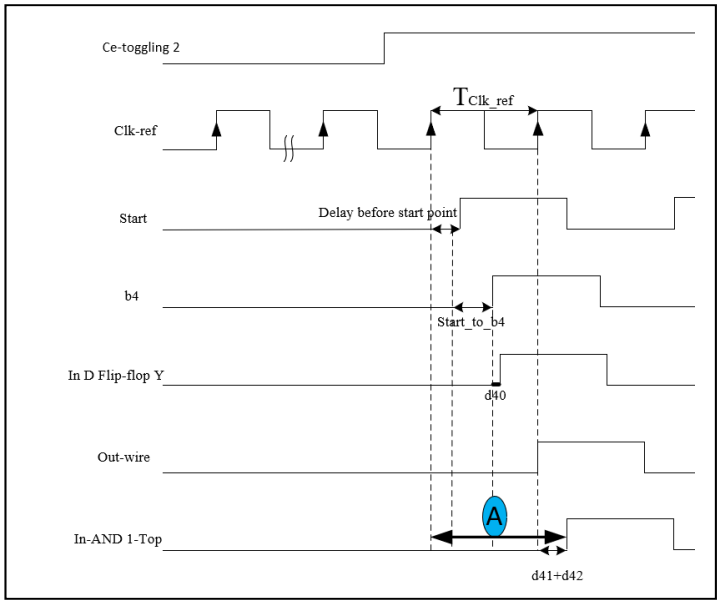

Figure-A XIII-5 Static delay margin timing diagram 1

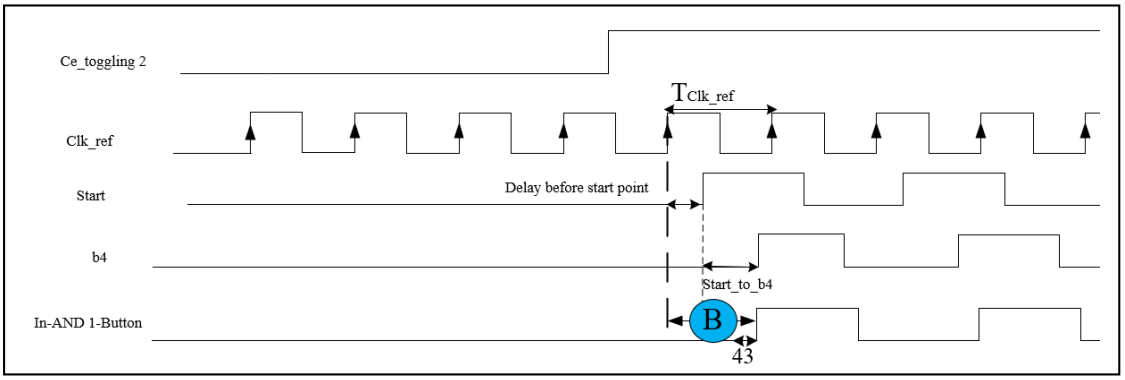

Figure-A XIII-6 Static delay margin timing diagram 2

If :

Delay before start point + Start_to_b4 = Clk_ref_to_b4 (A XIII-3)

By writing the equal delay values in XI-1 formula we have:

Estimated static delay margin = $(T_{clk\_ref} + d41 + d42) - (Clk\_ref\_to\_b4 + d43)$ (A XIII-4)

The systematic error is the difference between the measurement value in the worst-case scenario which is the static timing delay margin (formula XI-3) and the real delay margin (formula XI-1) as below:

$$(T_{clk\_ref} + d41 + d42) - (Clk\_ref\_to\_b4 + d43) - (T_{clk\_ref} - Clk\_ref\_to\_b4)$$
$$= d41 + d42 - d43 \quad \text{(A XIII-5)}$$

Where the delays in the worst-case, according to the static timing analyzer, are equal to:
D41=0.456 ns, D42=0.847 ns, D43= 1.239 ns

In this case XI-4 = 0.064 ns

BIBLIOGRAPHY

Altera. (April 4, 2018). " Clock Control Block (ALTCLKCTRL) IP Core User Guide." From https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_altclock.pdf.

Chan, A. H., & Roberts, G. W. (2004). A jitter characterization system using a component-invariant vernier delay line. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 12*(1), 79-95.

Gnad, D. R., Oboril, F., Kiamehr, S., & Tahoori, M. B. (2018). An experimental evaluation and analysis of transient voltage fluctuations in FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 26*(10), 1817-1830.

Katoh, K., Tanabe, T., Zahidul, H. M., Namba, K., & Ito, H. (2009). A delay measurement technique using signature registers. Dans *2009 Asian Test Symposium* (pp. 157-162). IEEE.

Larche, J. (2013). *Émulation et comparaison du mode test et du mode fonctionnel des circuits intégrés à horloges multiples* (École de technologie supérieure).

Matsumoto, T. (2005). High-resolution on-chip propagation delay detector for measuring within-chip variation. Dans *2005 International Conference on Integrated Circuit Design and Technology, 2005. ICICDT 2005.* (pp. 217-220). IEEE.

Mindrila, D., & Balentyne, P. (2017). Scatterplots and correlation. *Retrieved from*.

Narasimman, R., Prabhakar, A., & Chandrachoodan, N. (2015). Implementation of a 30 ps resolution time to digital converter in FPGA. Dans *2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)* (pp. 12-17). IEEE.

Oliver, J. P., Curto, J., Bouvier, D., Ramos, M., & Boemo, E. (2012). Clock gating and clock enable for FPGA power reduction. Dans *2012 VIII Southern Conference on Programmable Logic* (pp. 1-5). IEEE.

Pedram, M., & Rabaey, J. M. (2002). *Power aware design methodologies*. Springer Science & Business Media.

Pei, S., Li, H., & Li, X. (2009). A low overhead on-chip path delay measurement circuit. Dans *2009 Asian Test Symposium* (pp. 145-150). IEEE.

Raychowdhury, A., Ghosh, S., & Roy, K. (2005). A novel on-chip delay measurement hardware for efficient speed-binning. Dans *11th IEEE International On-Line Testing Symposium* (pp. 287-292). IEEE.

Ruffoni, M., & Bogliolo, A. (2002). Direct measures of path delays on commercial FPGA chips. Dans *6th IEEE Workshop on Signal Propagation on Interconnects*. Citeseer.

Shanmugasundaram, N. (2018). Clock gating techniques: an overview. Dans *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)* (pp. 217-221). IEEE.

Soni, Z., Patel, A., Panda, D. K., & Sarbadhikari, A. B. (2017). Comparative study of delay line based time to digital converter using FPGA. *International research Journal of Engineering and Technology (IRJET), 4*(9), 1169-1175.

Thibeault, C., & Gagnon, G. (2018). On the analysis and the mitigation of power supply noise and power distribution network impedance variation for scan-based delay testing techniques. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 26*(7), 1377-1390.

Torres, J., Aguilar, A., Garcia-Olcina, R., Martı, P., Martos, J., Soret, J., . . . Sanchez, F. (2013). Time-to-digital converter based on FPGA with multiple channel capability. *IEEE Transactions on Nuclear Science, 61*(1), 107-114.

Wong, J. S. (2011). Delay Measurements and Self Characterisation on FPGAs.

Wong, J. S., & Cheung, P. Y. (2011). Improved delay measurement method in FPGA based on transition probability. Dans *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays* (pp. 163-172).

Wong, J. S., Sedcole, P., & Cheung, P. Y. (2008). A transition probability based delay measurement method for arbitrary circuits on FPGAs. Dans *2008 International Conference on Field-Programmable Technology* (pp. 105-112). IEEE.

Xilinx. (August 13, 2012). " Analysis of Power Savings from Intelligent Clock Gating." From https://www.xilinx.com/support/documentation/application_notes/xapp790-7-series-clock-gating.pdf.

Xilinx. (April 4, 2018). " Virtual Input/Output LogiCORE IP Product Guide Vivado Design Suite." From https://www.xilinx.com/support/documentation/ip_documentation/vio/v3_0/pg159-vio.pdf.

Xilinx. (April 2, 2021). " Zynq-7000 SoC Technical Reference Manual UG585. "From https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

Xilinx. (July 2, 2018). " Zynq-7000 SoC Data Sheet: Overview." From https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.

Xilinx. (July 23, 2018). "7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital converter User Guide. "From https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf.

Xilinx. (July 30, 2018). "7 Series FPGAs Clocking Resources User Guide. "From https://www.xilinx.com/support/documentation/user_guides/ug472_7Series_Clocking.pdf.

Xilinx. (March 27, 2019). "ZC702 Evaluation Board for the Zynq-7000 XC7Z020 SoC User Guide." From http://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/ug850-zc702-eval-bd.pdf.

Xilinx. (October 4, 2017). " FIFO Generator LogiCORE IP Product Guide." From https://www.xilinx.com/support/documentation/ip_documentation/fifo_generator/v13_1/pg057-fifo-generator.pdf.

Xilinx. (September 27, 2016). "7 Series FPGAs Configurable Logic Block User Guide." From https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf.
Xilinx. (October 5, 2016). " Integrated Logic Analyzer v6.2 LogiCORE IP Product Guide. "From https://www.xilinx.com/support/documentation/ip_documentation/ila/v6_2/pg172-ila.pdf.

Yao, C. (2011). Time to Digital Converter used in ALL digital PLL.

Yeap, G. K. (2012). *Practical low power digital VLSI design*. Springer Science & Business Media.

Zhang, M., Wang, H., & Liu, B. (2020). A high precision TDC design based on FPGA+ ARM. Dans *Journal of Physics: Conference Series* (Vol. 1486, pp. 072054). IOP Publishing.

Zhang, Y., Roivainen, J., & Mammela, A. (2006). Clock-gating in FPGAs: A novel and comparative evaluation. Dans *9th EUROMICRO Conference on Digital System Design (DSD'06)* (pp. 584-590). IEEE.