Fachhochschule Dortmund
University of Applied Sciences and Arts

elmos

UNIVERSITY OF APPLIED SCIENCES AND ARTS, DORTMUND

MASTER THESIS

Digital Calibration, Closed Loop Regulation and Implementation of Digital Debugging Features for the Delay Asymmetry Compensation Logic of a 3D Polarization Camera Based on Time-of-Flight Principle

Author:
Jitikantha Sarangi

Supervisor:
Prof. Dr. Michael Karagounis

Examiner:
Mr. Felix Schneider

A thesis submitted in fulfillment of the requirements for the degree of Master of Engineering

in

Embedded Systems Engineering

August 29, 2023

Declaration of Authorship

I, Jitikantha Sarangi, declare that this thesis is titled as "Digital Calibration, Closed Loop Regulation and Implementation of Digital Debugging Features for the Delay Asymmetry Compensation Logic of a 3D Polarization Camera Based on Time-of-Flight Principle" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this university.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on the work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Jitikantha Sarangi

Place & Date: Dortmund, 29.08.2023

UNIVERSITY OF APPLIED SCIENCES AND ARTS, DORTMUND

Abstract

Embedded Systems Engineering
Department of Computer Science

Master of Engineering

Digital Calibration, Closed Loop Regulation and Implementation of Digital Debugging Features for the Delay Asymmetry Compensation Logic of a 3D Polarization Camera Based on Time-of-Flight Principle

by Jitikantha Sarangi

Flourishing technological developments have expedited the growth, usage and application of Time-of-Flight (ToF) cameras, which are extensively used in domains like automotive, robotics, computer vision, man-machine interaction etc. However, the actual performance of these cameras depend on many factors, such as properties of the target, intensity of background illumination, drivers and other circuits used, environmental conditions etc. The work presented in this thesis deals with the distance measurement aspect of a 3D Polarization ToF camera for automotive applications that uses a Time-to-Digital Converter (TDC) to measure the time interval between the emission of light from a source and its reception. Based on the measurement of the time interval, distance can be calculated by applying the equation of motion. In application, achieving an exact distance measurement is quite strenuous because the operating conditions of the design are susceptible to change due to environmental factors. Therefore, to achieve accuracy in distance measurement, the time interval between the emission and reception of light must be measured precisely. For this purpose, a delay asymmetry compensation logic is developed. This thesis elaborates the addition of debugging features, redesign of some components, digital calibration approach and the entire testbench environment of the delay asymmetry compensation logic. It also sheds light on the implementation of the designed logic for its successful realisation in real hardware. Lastly, it concludes by narrating future prospects and further scopes of development.

Acknowledgements

Throughout the writing of this report, I have received a great deal of support, assistance and guidance. First of all, I would like to thank my supervisor, Prof. Dr. Michael Karagounis, whose expertise was invaluable in formulating the ideas and methodologies, in particular.

All the learnings, results and outcomes would not have been possible without the guidance of my mentor Mr. Felix Schneider. His knowledge and expertise helped a lot in solving the problems I faced during the design and implementation phase. I would like to thank him for helping me out with all the probelms and especially providing me with enough appointments for discussion whenever I needed one.

I would like to take this opportunity to express my gratitude to the university administration, the lab staffs and most importantly, Elmos Semiconductor SE, for assisting me with all the resources, hardware and tools required for this project.

I would also like to thank my wonderful colleagues Ms. Ladan Alaee, Mr. Alexander Horstkötter, Dr. Wolfram Budde, Dr. Thomas Rotter and Mr. Rolf Paulus for helping and supporting me in various aspects, directly and indirectly throughout the project and during the writing of this thesis.

And, last but the most important one, I would like to express my profound gratitude to my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study at Fachhochschule Dortmund. This accomplishment would not have been possible without them.

Contents

List of Figures

List of Tables

List of Abbreviations

ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
DAC	**D**igital to **A**nalog **C**onverter
ECO	**E**ngineering **C**hange **O**rder
FSM	**F**inite **S**tate **M**achine
I2C	**I**nter **I**ntegrated **C**ircuit
IC	**I**ntegrated **C**ircuit
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
LED	**L**ight **E**mitting **D**iode
PID	**P**roportional **I**ntegral **D**erivative
SDC	**S**ynopsys **D**esign **C**onstraints
SDF	**S**tandard **D**elay **F**ormat
SPEF	**S**tandard **P**arasitic **E**xchange **F**ormat
STA	**S**tatic **T**iming **A**nalysis
ToF	**T**ime **o**f **F**light
TIA	**T**rans **I**mpedance **A**mplifier
TDC	**T**ime to **D**igital **C**onverter

Chapter 1

Introduction

There have been tremendous advancements in technology in the past two decades. This entire world has been revolutionized by technology. Surely, it has eased and enhanced the quality of our everyday lives. Starting from the emergence of internet to the prevalence of smartphones, every information is just a fingertip away from us. There have been exhilerating expansions in the field of electronics and computers as well. Modern day electonic devices are becoming more intelligent and miniature. Though technology is advancing more towards digital systems due to the ease digital signals can be handled with, we live in a world where the environment represents analog properties like light, heat, sound etc. To bridge the gap between these two domains, sensors have found their ways to electonic devices. In everyday lives as well as in industries, sensors provide efficiency, safety and comfort through automated interaction between man and machines. Knowingly or unknowingly, each and every one of us is interacting with many types of sensors on a daily basis. Starting from smartphones, smartwatches and cars to home automation devices and healthcare equipments, we come in contact with numerous sensors e.g. proximity snesors, light sensors, pressure sensors, temperature sensors, touch sensors etc.

Use of sensor technology in automotive applications is increasing day by day. Intelligent and autonomous vehicle functionalities like lane departure warning, adaptive cruise control, parking assist, emergency breaking and blindspot detection use a great varieties of sensors. Modern vehicles are equipped with a wide range of complex electronic sensor systems which help the vehicle make numerous decisions on its own based on the data provided by the sensors that are interfaced to the vehicles' on-board computer systems. Many of these sensors operate in rough and harsh conditions and are subjected to extreme temperature, pressure, vibrations etc. Therefore, robust design of these sensors is necessary for safety and effective functionalities of vehicles. Among these sensors, Time-of-Flight sensors are one of the most important ones.

1.1 Principle of ToF sensors

Time-of-Flight sensors fall under the category of optical sensors. Their principle of working is similar to sonar technology but instead of sound waves, they work on light waves. As known as a ToF camera, a ToF sensor measures the distance of an object from a source by emitting a beam of light and measuring the time interval between the light emission and its reception. With the help of light pulses, usually generated by a laser light source, high precision measurements can be carried out. This principle provides greater range, higher resolution, greater accuracy and faster

readings while still maintaining small size and low power. ToF sensors enable endless varieties of applications and use cases like gesture conrol, obstacle detection in case of robot navigation, object detection, vehicle monitoring, people counting etc.

In laser based ToF cameras, the entire scene is captured with light pulses. For distance measurement of an object which is at a distance d as shown in figure 1.1, the propagation time T_d of a light pulse originating from a light source, being reflected by the object, and then reaching the pixel of a sensor placed in the immediate vicinity of the light source is measured.

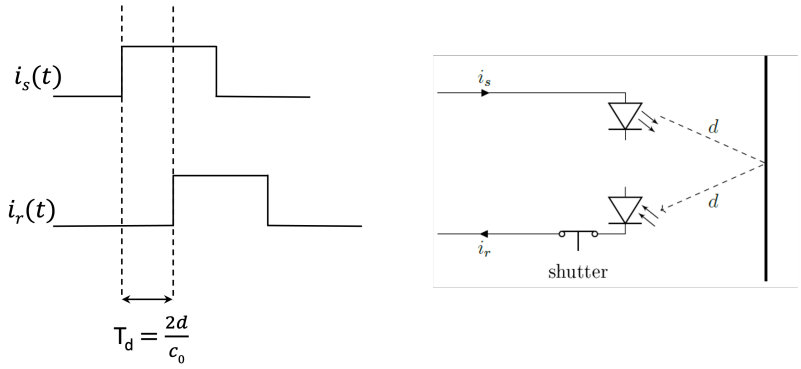

FIGURE 1.1: Time-of-Flight Distance Measurement Principle

The distance d can be calculated by using the following formula:

$$d = \frac{1}{2} c_0 * T_d \tag{1.1}$$

The relationship between the distance and time resolution evaluates to around 67 $\frac{ps}{cm}$ using the speed of light in vacuum. Hence, to achieve a distance resolution in centimeter precision, T_d needs to be measured with an accuracy below 100ps for each pixel independently.

1.2 Motivation & Goal

As safety concerns are of utmost importance in automotive vehicles, the ToF sensors used in the vehicles must function correctly even under harsh environmental conditions. Operating these sensors under these critical environmental conditions might lead to inaccuracies in measurements which could have hazardous impacts. So, these inaccuracies must be compensated by additional logics or electronic circuits to achieve precise results. As the propagation speed of light pulse is very high, small deviations in time measurements can lead to large deviations in distance measurement.

Elmos Semiconductor SE in Dortmund is developing a 3D camera known as the "Panoptes Testchip"[1] which is to be used in automotive applications. Using ToF principle, this camera is intended to generate precise distance images for the detection of objects and calculation of their distance from the vehicle. For measurement of time interval between the emission of light and its reception, a Time-to-Digital Converter[2] is used. To measure this time interval, an indirect approach is followed by

applying window functions to the photocurrent at the receiver which is explained in section 4.1.1 of chapter 3. For realising this indirect approach, another time value parameter needs to be measured but the changes in operating conditions might lead to an increase or decrease in the value of the time parameter. To ensure a successful and precise measurement of that time value parameter, a delay asymmetry compensation logic circuit[3] is designed. This thesis narrates the redesign of some parts of the delay asymmetry compensation logic and addition of some other logics and pararmeterized modules into it which ensure precision in measurement and overall performance efficiency. To gurantee the accuracy and precision of the new design approach, the system is also subjected to changes in parameters which could occur due to changes in operating conditions in real life environment and thus, verified against a wide range of parameters.

1.3 Thesis Outline

This thesis covers a variety of topics that have been clearly defined into chapters. It is broadly classified into 5 chapters. Chapter 2 consists of Theoretical foundations and studies which provide an overview of the topics that need to be understood before proceeding to the subsequent chapters. These basic concepts stand as the premises on which the following chapters are delineated. Chapter 3 introduces the history behind the Panoptes Testchip and covers the ideas, design approaches and challenges faced during design phase. It also covers the addition of new modules and features into the Testchip by dividing the chapter into different sections and subdividing the sections into different subsections for convenience and readability purposes. Chapter 4 covers the approaches for verifying the designed system for accuracy and precision to ensure faultless functionalities. Creation of a self checking testbench and the entire testbench environment containing all the modules in an interconnected fashion is the major part of this chapter. It also covers the implementation of the design into real life hardware to ensure proper realisation of the system in ICs. Chapter 5 contains some additional discussions, remarks and concludes with possible scopes for further development in this area.

Chapter 2

Theoretical Foundation

This chapter describes some basic fundamentals that need to be understood before moving on to design implementation. Some basic theoretical concepts covered in this chapter are the working principle of a PID controller[4], IEEE floating point standard[5] and the devised floating point algorithm[3] for the PID controller, basic structure, working principle[6] and some important concepts[7] associated with TDC, basics of I2C protocol[8] and how the communication happens over an I2C bus etc. In this work presented here, the devised floating point algorithm of the PID controller and the structure and basic fundamentals of the TDC are taken from previous works on the basis of which the redesign of PID controller and new calibration approach are done.

2.1 PID Controller

PID controller, an abbreviation for Proportional-Integral-Derivative controller, is a feedback control system widely used in various engineering applications and in industries to regulate processes and maintain desired setpoints by adjusting the control inputs to systems. The PID controller continuously calculates an error signal as the difference between the desired setpoint and the measured process variable. Then, it uses this error signal to adjust the control input in a way that reduces the error and brings the system closer to the setpoint. An example of a control system with a parallel PID controller is shown in figure 2.1.

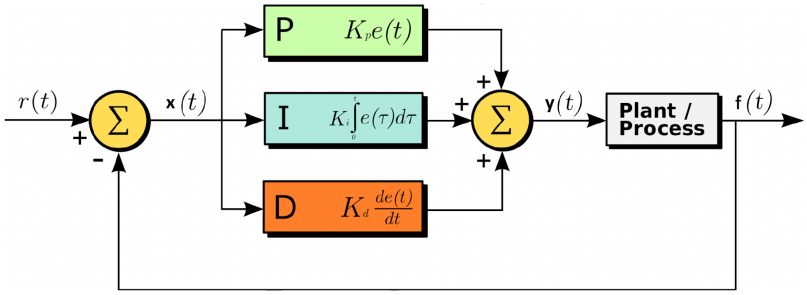

FIGURE 2.1: PID Controller

A parallel PID controller is much more preferred than a PID controller where the proportional, integral and derivative components are connected in series. The parallel structure allows a complete decoupling of the proportional, integral and derivative parts whereas in the series structure, a modification in the gain of one component affects the action of other components. Also, by switching off the proportional, integral and derivative parts of the controller individually at runtime, other types of

controllers like P, PI, PD can be realised. By tuning each components individually, a wide range of behaviour can also be achieved in case of a parallel configuration.

Some properties of the individual components of a PID controller are the following:

- The proportional term is directly proportional to the error value. Gain of the proportional part determines the sensitivity of the controller to the error. A larger gain results in a stronger response but may lead to oscillations and instability, while a smaller gain results in a weaker response and slower settling time.
- The integral term considers the accumulation of past errors over time and helps in eliminating steady-state errors in the system. Integral gain determines the controller's response to the accumulated error. A larger gain increases the integral action and helps in eliminating steady-state errors, but it can also lead to overshooting and instability if set too high.
- The derivative term anticipates future errors by considering the rate of change of the error. It helps in damping oscillations and reducing overshoot. Derivative gain determines the controller's response to the rate of change of the error. A larger gain increases the derivative action and helps in damping oscillations, but it can also make the system more sensitive to noise.

The PID controller shown in figure 2.1 uses continuous time integrator and differentiator and operates on continuous time signals. By using the control properties of all the three components of the PID controller, the continuous time transfer function[9] can be given by the following equation.

$$y(t) = P * x(t) + \int_0^t x(t)\ dt + D * \frac{dx(t)}{dt} \tag{2.1}$$

By taking the Laplace transformation of the above equation, the continuous time transfer function can be written as:

$$Y(S) = P * X(S) + \frac{I}{s} * X(S) + D * s * X(S) \tag{2.2}$$

By substituting

$$C(S) = \frac{Y(S)}{X(S)} \tag{2.3}$$

$$C(S) = P + \frac{I}{s} + D * s \tag{2.4}$$

By using backward Euler approximation[10] to derive a discrete time system[11] from the continuous time transfer function, s can be written as:

$$s = \frac{z - 1}{z * T_s} \tag{2.5}$$

substituting this value of s in equation 2.4

$$\begin{aligned} C(z) &= P + \frac{I * T_s * z}{z-1} + D * \frac{z-1}{z * T_s} \\ &= P + I * T_s * \frac{z}{z-1} + \frac{D}{T_s} * \frac{z-1}{z} \\ &= \frac{(P + I * T_s + \frac{D}{T_s}) * z^2 + (-P - 2\frac{D}{T_s}) * z + \frac{D}{T_s}}{z^2 - z} \end{aligned} \tag{2.6}$$

For a given scenario in the discrete time domain, the sampling period T_s is assumed to be constant. Substituting $K_p = P$, $K_i = I * T_s$, $K_d = \frac{D}{T_s}$ in equation 2.6

$$\begin{aligned} Y(z)(z^2 - z) = &(K_p + K_i + K_d) * X(z) * z^2 \\ &+ (-K_p - 2 * K_d) * X(z) * z \\ &+ K_d * X(z) \end{aligned} \tag{2.7}$$

The inverse z-Transform gains the difference equation for the PID controller which can be given by

$$\begin{aligned} y[n] = &y[n-1] + x[n] * (K_p + K_i + K_d) \\ &+ x[n-1] * (-K_p - 2 * K_d) \\ &+ x[n-2] * K_d \end{aligned} \tag{2.8}$$

In this PID controller description, the integrator state is contained within the previous output value. The state of the controller is contained with the registers y[n - 1], x[n - 1] and x[n - 2]. The coefficients only have to be recomputed once the sampling time or one of the gains change. Equation 2.8 can be rewritten as:

$$\begin{aligned} y[n] &= y[0] + K_p * \left[\sum_{k=0}^{n} x_k - \sum_{k=0}^{n-1} x_k \right] \\ &\quad + K_i * \sum_{k=0}^{n} x_k \\ &\quad + K_d * \left[\sum_{k=0}^{n} x_k - 2 * \sum_{k=0}^{n-1} x_k + \sum_{k=0}^{n-2} x_k \right] \\ &= y[0] + K_p * x[n] + K_i * \sum_{k=0}^{n} x_k + K_d * \big(x[n] - x[n-1]\big) \end{aligned} \tag{2.9}$$

Substitution of the following in equation 2.9 gives rise to equation 2.11.

$$\begin{aligned} y[0] &= 0 \\ i[n] &= \sum_{k=0}^{n} x_k = i[n-1] + x[n] \\ d[n] &= x[n] - x[n-1] \end{aligned} \tag{2.10}$$

$$y[n] = K_p * x[n] + K_i * i[n] + K_d * d[n] \tag{2.11}$$

In equation 2.9, proportional, integral and derivative components are split up in separate summands and the initial condition is y[0] = 0. Since the integral part is separated, it can be written in recursive form and has its own state register i[n]. To calculate the next time step, only two registers are required here, i[n - 1] and x[n - 1]. Three multiplications and four additions are required here for the computation.

The gains of the proportional, integral and derivative components- K_p, K_i and K_d could be represented as signed integers, signed fixed points, signed floating points etc. For integers and fixed point numbers, the logic implementation of the calculations are fairly easy- only adders and multipliers are needed. Compared to integers, fractional numbers can be represented by fixed points. However, for both integers and fixed point numbers, the range of numbers that can be represented are limited whereas floating point numbers can cover larger dynamic ranges of numbers. They can also accomodate calculations with numbers that are multiple orders of magnitude apart at relatively small bit widths as compared to integers and fixed point numbers. So, a large range for tuning of the gain coefficients can be provided in a better manner if the gain coefficients are represented in floating point format.

2.2 Floating Point Representation

Floating-point arithmetic is a method used in computing to represent and perform calculations with real numbers. It's designed to approximate a wide range of real numbers with a fixed number of binary bits. In floating-point arithmetic, real numbers are represented as a combination of a sign, a significand or mantissa, and an exponent. The general form is as follows:

$$number = sign * mantissa * base^{exponent} \tag{2.12}$$

The "sign" bit defines the sign of the number and whether the number is positive or negative. The "mantissa" contains the fractional part of the number, and the "exponent" determines the magnitude of the number relative to the base. The "base" is typically 2 for most computer systems but sometimes, a base of 10 is also used. The most widely used standard for floating point representation is IEEE 754[5]. This standard defines formats for single precision (32-bit) and double precision (64-bit) floating point numbers. It also specifies rules for arithmetic operations and rounding behavior to ensure consistent results. In IEEE 754, floating point numbers are normalized, meaning the most significant bit of the mantissa is always 1, except for special cases like zero and denormalized numbers. Also, the exponent is always 2 for IEEE 754. Denormalized numbers allow for representing extremely small values close to zero that would otherwise be too small to represent normally. The range and precision of floating point numbers depend on the number of bits allocated to the mantissa and exponent.

For IEEE 754, equation 2.12 can be written as

$$Z = (-1)^s * (1 + m) * 2^e \tag{2.13}$$

2.3 Devised Floating Point Representation for PID

As addition and multiplication in IEEE floating point format is relatively complex, for ease and convenience, a custom floating point representation is devised for the calculations related to the PID controller. According to this custom representation, a number Z can be written as

$$Z = m * 2^e \tag{2.14}$$

Here, m is a 2's complement number. So, the number representation doesn't require a separete sign bit. In multiplication, instead of resulting mantissa normalization, the bit width of the resulting mantissa is grown to accomodate the result in the following manner:

$$Z_3 = Z_1 * Z_2 = m_1 * m_2 * 2^{e_1+e_2} = m_3 * 2^{e_3} \tag{2.15}$$

where

$$\underset{\longleftrightarrow}{m_3} = \underset{\longleftrightarrow}{m_1} + \underset{\longleftrightarrow}{m_2} \tag{2.16}$$

and

$$\underset{\longleftrightarrow}{e_3} = max(\underset{\longleftrightarrow}{e_1}, \underset{\longleftrightarrow}{e_2}) + 1 \tag{2.17}$$

Using this approach, the floating point multiplication can be represented as one integer addition and one integer multiplication. When adding two numbers, following approach is used:

$$Z_3 = Z_1 + Z_2 = m_1 * 2^{e_1} + m_2 * 2^{e_2} \tag{2.18}$$

$$e_3 = min(e_1, e_2) \tag{2.19}$$

$$\begin{aligned}
Z_1 &= m_1 * 2^{e_1-e_3} * 2^{e_3} = \hat{m_1} * 2^{e_3} \quad ; \quad \underset{\longleftrightarrow}{\hat{m_1}} = \underset{\longleftrightarrow}{m_1} + (e_1 - e_3) \\
Z_2 &= m_2 * 2^{e_2-e_3} * 2^{e_3} = \hat{m_2} * 2^{e_3} \quad ; \quad \underset{\longleftrightarrow}{\hat{m_2}} = \underset{\longleftrightarrow}{m_2} + (e_2 - e_3) \\
Z_1 + Z_2 = Z_3 &= (\hat{m_1} + \hat{m_1}) * 2^{e_3} = m_3 * 2^{e_3} \quad ; \quad \underset{\longleftrightarrow}{m_3} = \underset{\longleftrightarrow}{\hat{m_1}} + \underset{\longleftrightarrow}{\hat{m_2}} + 1
\end{aligned} \tag{2.20}$$

Both numbers are brought to a common exponent e_3 in this calculation. The smaller of the two summands' exponents is chosen as the common exponent so that no information is lost in the process. In order to compensate for the matching of the exponent, the mantissa of the summand with the larger exponent is left shifted and grown accordingly.

2.4 Applying the devised representation to PID

There should be provision to supply negative exponents to the PID controller. For a given bit width, the set of representable exponents for a 2's complement number

is predetermined. For example, a 4 bit exponent in 2's complement can represent 16 exponents: -2^3,...,2^3 - 1. But if 16 exponent from -1,...,14 need to be represented, the same 4 bit 2's complement exponent will not be large enough. For this reason, for each coefficient, the minimum exponent value e_{min} is defined by a parameter. Instead of supplying full exponents as inputs to the PID controller, biased exponents Δe are used. The actual exponent is then calculated as $e = e_{\min} + \Delta e$. Considering the example, to represent -1,...,14, $e_{\min} = -1$ with a 4 bit $0 \leq \Delta e \leq 15$ can be taken into account. This gives $-1 \leq e = e_{\min} + \Delta e \leq 14$. This Δe is an unsigned number which can simplify the calculations and the associated logic design.

Because of the series of calculations involved, the number representation grow continuously. However, the mantissa of the output of the PID controller has a fixed bit width. So, rounding is necessary. The final result of the PID computation is y[n] and it is rounded to $\tilde{y}[n]$.

$$\tilde{y}[n] = m_{\tilde{y}} * 2^{e_{\tilde{y}}} \tag{2.21}$$

and

$$\tilde{y}[n] \approx y[n] \tag{2.22}$$

Parameter	Symbol	Meaning
WidthX	$\underset{\longleftrightarrow}{x[n]}$	Bit width of the error signal
WidthY	$\underset{\longleftrightarrow}{m_{\tilde{y}}[n]}$	Bit width of the mantissa of the rounded PID output value
WidthAcc	$\underset{\longleftrightarrow}{i[n]}$	Bit width of the accumulator inside the integrator
WidthK_p	$\underset{\longleftrightarrow}{m_{\text{K}_\text{p}}}$	Bit widths of the mantissae of the coefficients
WidthK_i	$\underset{\longleftrightarrow}{m_{\text{K}_\text{i}}}$	
WidthK_d	$\underset{\longleftrightarrow}{m_{\text{K}_\text{d}}}$	
$\text{WidthDExpK}_\text{p}$	$\underset{\longleftrightarrow}{\Delta e_{\text{K}_\text{p}}}$	Bit widths of the biased exponents of the coefficients
$\text{WidthDExpK}_\text{i}$	$\underset{\longleftrightarrow}{\Delta e_{\text{K}_\text{i}}}$	
$\text{WidthDExpK}_\text{d}$	$\underset{\longleftrightarrow}{\Delta e_{\text{K}_\text{d}}}$	
WidthDExpY	$\underset{\longleftrightarrow}{\Delta e_{\text{y}}}$	Bit width of the biased exponent of PID output
ExpMinK_p	$\text{e}_{\text{K}_\text{p},\text{min}}$	Minimum exponents of the coefficients
ExpMinK_i	$\text{e}_{\text{K}_\text{i},\text{min}}$	
ExpMinK_d	$\text{e}_{\text{K}_\text{d},\text{min}}$	
ExpMaxK_p	$\text{e}_{\text{K}_\text{p},\text{max}}$	Maximum exponents of the coefficients
ExpMaxK_i	$\text{e}_{\text{K}_\text{i},\text{max}}$	
ExpMaxK_d	$\text{e}_{\text{K}_\text{d},\text{max}}$	
ExpMinY	$\text{e}_{\tilde{y},\text{min}}$	Minimum exponent of the rounded PID output
ExpMaxY	$\text{e}_{\tilde{y},\text{max}}$	Maximum exponent of the rounded PID output

TABLE 2.1: Parameters of The PID Controller

Signal	Width	Symbol	I/O	Signed	Meaning
x	WidthX	x[n]	I	Yes	Current input to PID (error value)
y	WidthY	$m_{\tilde{y}}$	O	Yes	Mantissa of the rounded output value of PID
k_p	WidthK_p	m_{K_p}	I	Yes	Mantissae of P, I and D coefficients
k_i	WidthK_i	m_{K_i}			
k_d	WidthK_d	m_{K_d}			
dexp_k_p	WidthDExpK_p	Δe_{K_p}	I	No	Biased exponents of P, I and D coefficients
dexp_k_i	WidthDExpK_i	Δe_{K_i}			
dexp_k_d	WidthDExpK_d	Δe_{K_d}			
dexp_y	WidthDExpY	Δe_y	I	No	Biased exponent of output y[n]
sat	1		O		Active high signal indicating saturation of Integrator

TABLE 2.2: Inputs and Outputs of The PID Controller

2.4.1 Derivation of the Floating Point Equations

The coefficient inputs to the PID controller are the following:

$$\begin{aligned} K_p &= m_{K_p} * 2^{e_{K_p}} = m_{K_p} * 2^{e_{K_p,\min}+\Delta e_{K_p}} \\ K_i &= m_{K_i} * 2^{e_{K_i}} = m_{K_i} * 2^{e_{K_i,\min}+\Delta e_{K_i}} \\ K_d &= m_{K_d} * 2^{e_{K_d}} = m_{K_d} * 2^{e_{K_d,\min}+\Delta e_{K_d}} \end{aligned} \tag{2.23}$$

Substituting equation 2.23 in equation 2.11

$$\begin{aligned} y[n] = m_y * 2^{e_y} &= K_p * x[n] + K_i * i[n] + K_d * d[n] \\ &= \underbrace{x[n] * m_{K_p}}_{m_p} * 2^{e_{K_p}} + \underbrace{i[n] * m_{K_i}}_{m_i} * 2^{e_{K_i}} + \underbrace{d[n] * m_{K_d}}_{m_d} * 2^{e_{K_d}} \end{aligned} \tag{2.24}$$

Where

$$\begin{aligned} \underset{\longleftrightarrow}{m_p} &= \underset{\longleftrightarrow}{x[n]} + \underset{\longleftrightarrow}{m_{K_p}} \\ \underset{\longleftrightarrow}{m_i} &= \underset{\longleftrightarrow}{i[n]} + \underset{\longleftrightarrow}{m_{K_i}} \\ \underset{\longleftrightarrow}{m_d} &= \underset{\longleftrightarrow}{d[n]} + \underset{\longleftrightarrow}{m_{K_d}} = \underset{\longleftrightarrow}{x[n]} + 1 + \underset{\longleftrightarrow}{m_{K_d}} \end{aligned} \tag{2.25}$$

For addition of proportional, integral and derivative components, a common exponent is established. Since the mantissae are grown according to the exponents, choosing the smallest possible exponent facilitates no loss in information.

$$e_y = min(e_{K_p,\min}, e_{K_i,\min}, e_{K_d,\min}) \tag{2.26}$$

$$y[n] = m_y * 2^{e_y} = \Big(\underbrace{m_p * 2^{e_{K_p} - e_y}}_{\hat{m_p}} + \underbrace{m_i * 2^{e_{K_i} - e_y}}_{\hat{m_i}} + \underbrace{m_d * 2^{e_{K_d} - e_y}}_{\hat{m_d}} \Big) * 2^{e_y} \tag{2.27}$$

Substituting e_{K_p}, e_{K_i} and e_{K_d} with their biased exponent representation from earlier:

$$\begin{aligned} y[n] &= m_y * 2^{e_y} \\ &= \Big(\underbrace{m_p * 2^{e_{K_p,\min} - e_y + \Delta e_{K_p}}}_{\hat{m_p}} + \underbrace{m_i * 2^{e_{K_i,\min} - e_y + \Delta e_{K_i}}}_{\hat{m_i}} + \underbrace{m_d * 2^{e_{K_d,\min} - e_y + \Delta e_{K_d}}}_{\hat{m_d}} \Big) * 2^{e_y} \end{aligned} \tag{2.28}$$

Where the mantissa of the output of the PID controller m_y can be written as

$$m_y = \hat{m_p} + \hat{m_i} + \hat{m_d} \tag{2.29}$$

As explained earlier, after computing the output of the PID controller y[n], it is rounded to $\tilde{y}[n]$. From equation 2.22, the final rounded output of the PID controller can be given by

$$\begin{aligned} &\tilde{y}[n] = y[n] \\ \Rightarrow &m_{\tilde{y}} * 2^{e_{\tilde{y}}} = m_y * 2^{e_y} \\ \Rightarrow &m_{\tilde{y}} * 2^{e_{\tilde{y}}} = m_y * 2^{e_y - e_{\tilde{y}}} * 2^{e_{\tilde{y}}} \\ \Rightarrow &m_{\tilde{y}} = m_y * 2^{e_y - e_{\tilde{y}}} \end{aligned} \tag{2.30}$$

2.5 Time-to-Digital Converter

A Time-to-Digital Converter is a type of electronic circuit used to measure the time interval between two events and convert it to a digital representation. It is used in applications like ToF, radar systems, experiments involving laser ranging, measurements in atomic and high energy physics etc. The TDC works by quantizing the time interval between events into digital pulses or codes. There are different types of TDC architectures like Vernier TDC, digital TDC, Time-to-Code Converter TDC etc. Among these, digital TDCs are used vastly in integrated circuits.

The quantization of time interval in digital TDC is achieved by a reference clock. The TDC contains a clock generator and a counter. The principle is to measure the number of clock cycles between the time interval. The time interval is bounded by *start* and *stop* signals. When the first event occurs, the start signal goes high and the counter starts counting the clock cycles and when the second event is detected by the assertion of the *stop* signal, the counter stops. The counter value corresponds to the time interval between the two events and is represented in digital code. Here, the resolution of the TDC is limited by the frequency of the refernce clock. However, quantization errors might occur in this type of measurement because of the asynchronicity between the *start* and *stop* signal and the reference clock.

Before moving on to the structure of the TDC, it is necessary to understand the delay-line based TDC. By dividing the period of the reference clock into smaller time segments, the resolution of the TDC can be increased. This can be achieved by using a chain of identical delay elements called delay-lines which produce phase shifted duplicates of the reference clock. Figure 2.2 shows the basic structure of a delay-line based TDC. Here, the reference clock signal is considered as the *start* signal of the timing event. It is delayed by the delay elements and supplied as inputs to the flip-flops present after each delay element. The *stop* signal is used as the clock input for the flip-flips of the delay-line elements. When the *stop* signal arrives, a logical HIGH level is stored in all the flip-flops for which the *start* signal is already present. All the other flip-flops where the *start* signal has not yet arrived store a logical LOW value. The outputs of the flip-flop taken altogether create a thermometer code whose HIGH-LOW transitions indicate how many delay elements the *start* signal passes until the *stop* signal arrives.

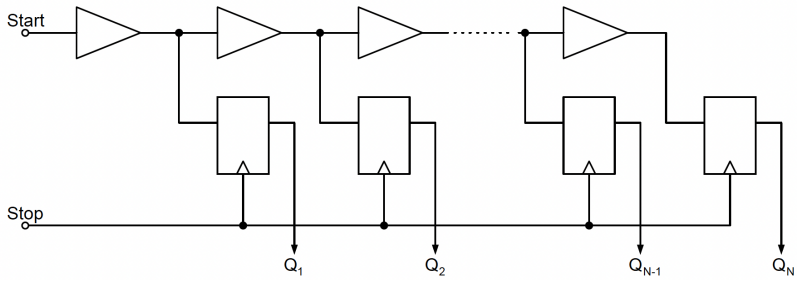

FIGURE 2.2: Basic Delay-Line based TDC

2.5.1 Structure of the TDC

The primary goal of the delay-line of the TDC used in this design is to delay the incoming clock signal by exactly one clock period. To achieve this over a wide range of temperature and operatinig conditions, there is a provision to control the delay of the delay-line. To change the delay of the delay elements, the supply voltage of the inverters inserted before the delay-line is adjusted till the delay of the dely-line corresponds to one clock period of the reference clock. In this application, there are 180 elements in the delay-line. Additional elements placed at the beginning and end of the delay-line don't contribute to the overall delay. The structure of the voltage controlled delay-line is shown is figure 2.3.

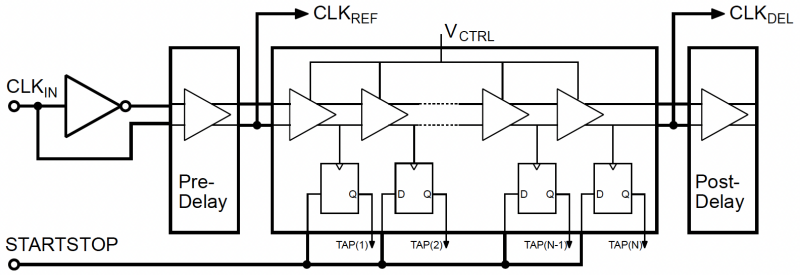

FIGURE 2.3: Voltage Controlled Delay-Line

The individual delay elements of the delay-line consist of a number of components in addition to the differential inverters. The structure of a single delay element is shown in figure 2.4.

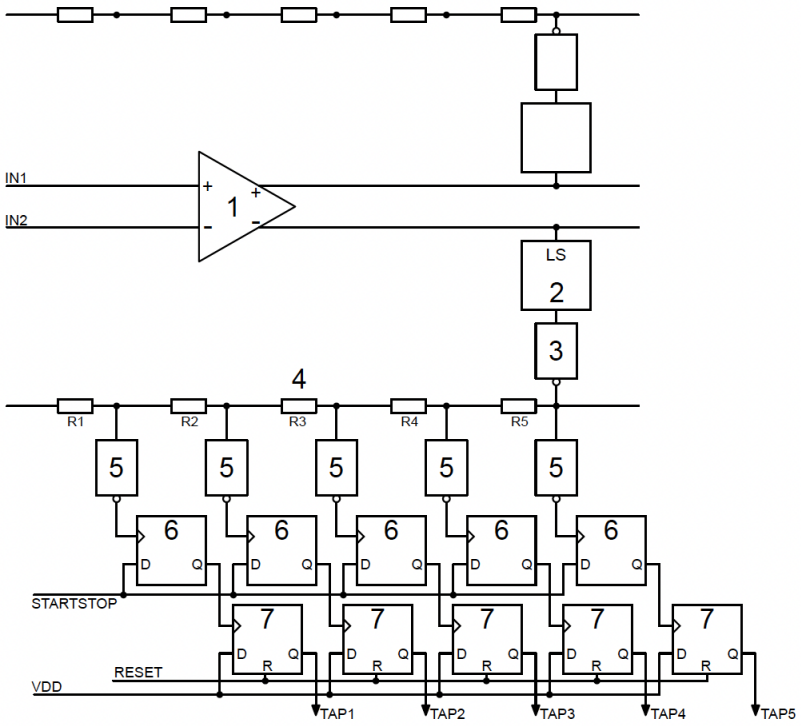

FIGURE 2.4: Structure of a Single Delay Element

The control voltage V_{CTRL} is used as the supply voltage for the inverter chain in the delay-line. The output voltage of the inverters (1) are limited to the value of the control voltage. To process the outputs of the delay-line by circuits supplied by 3.3V, the outputs of the inverters need to be raised beyond the control voltage level. For this purpose, level shifters (2) are used. A strong buffer stage (3) comprised of series connected inverters are used to drive the interpolation resistors (4) and the output buffers (5) associated with them. The interpolation reistors are used to improve the resolution.

Each output of the interpolation register, after passing through the buffer stage, is connected to the clock input of a flip-flop. These flip-flops are termed as delay-line flip-flops. The *STARTSTOP* signal, created by the *start* and *stop* signal, is fed as input data to the flip-flops. When the clock signal passes through the delay-line, till the time the *STARTSTOP* signal is active, the output of each flip-flop through which the clock signal passes is switched to a logic HIGH state. This generates the thermometer code which represents the time for which the *STARTSTOP* signal was active, which is nothing but the measured time interval. After one clock period, these delay-line flip-flops are reset with each rising edge of the clock. To store the thermometer code beyond the period of one clock cycle, another set of flip-flops named as output flip-flops are used. The supply voltage is connected as input to these flip flops and the outputs from the delay-line flops are connected to the clock inputs.

Figure 2.5 shows the entire structure of the TDC consisting of different components. The functionalities of these components are described briefly.

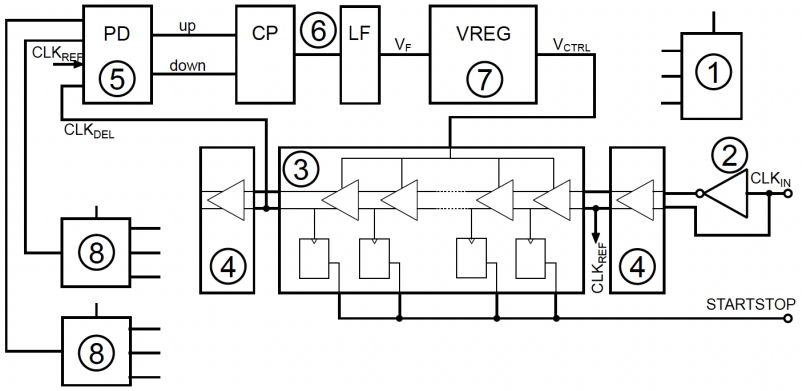

FIGURE 2.5: Structure of the TDC

A clock signal with a frequency of 40 MHz is used as the input to the TDC, named by CLK_{IN}. To generate an inverted input signal for the differential delay-line, it is passed through a series of inverters (2). This clock signal, after passing through the delay-line (3) generates the delayed clock CLK_{DEL}. For load balancing at the input and output of the delay elements, some dummy elements (4) are placed at the beginning and end of the delay-line. These elements don't contribute to the overall delay of the delay-line. The reference clock CLK_{REF} is picked up after passing through the dummy element as shown in the figure.

The reference clock and the delayed clock signals are compared with each other in terms of their phase position by the help of a phase detector (5). Depending on the leading clock edge, the phase detector generates *UP* or *DOWN* signal which is fed to a charge pump with loop filter (6). Based on the input, the charge pump supplies positive or negative current to the loop filter which in turn increases or decreases the voltage V_F which is input to the linear regulator. The output from the voltage regulator is V_{CTRL} which supplies and in turn controls the delay-line.

2.5.2 Concept of Double Sampling and Blind Spot

When the clock signal propagates through the delay-line, the flip-flops gets sampled on the rising edge of the clock. Based on whether the delay-line is too slow or too fast, the sampling of the flip-flops gets affected. This might lead to a double sampling or a blind spot phenomenon in the TDC. Figure 2.6 and figure 2.7 showing the examples of double sampling and blind spot effects, respectively are presented here for better understanding.

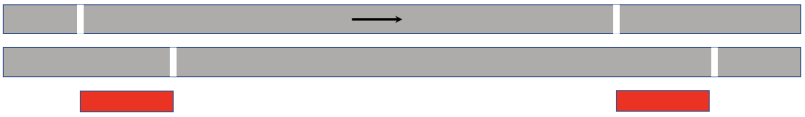

FIGURE 2.6: Example Showing Double Sampling Effect

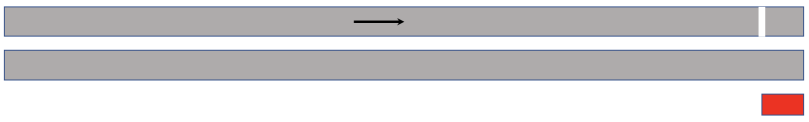

FIGURE 2.7: Example Showing Blind Spot Effect

In figure 2.6 and figure 2.7, the colors grey, white and red represent the delay-line, rising edge of the clock signal that propagates through the delay-line and the number of flip-flops that get sampled to logic HIGH state during the presence of the *STARTSTOP* pulse, respectively.

When the delay-line is too slow, the rising edge of one clock cycle might be still present in the delay-line when the rising edge of the next clock cycle enters. This leads to the sampling of the flip-flops twice because of the presence of two rising edges of the clock signal. It is observed that during the presence of one rising edge in the delay-line, the number of flip-flops triggered are marginally lower than the ideal value but during the presence of two rising edges, the number is twice as high. This is termed as double sampling effect. But when the delay-line is too fast, the rising edge of one clock cycle might have left or is just about to leave the delay-line before the rising edge of the next clock cycle enters. So, the number of flip-flops sampled are notably less than the ideal value. This effect is termed as blind spot.

2.6 I2C Protocol

Inter Integrated Circuit (I2C) is a serial communication protocol used to transfer data between integrated circuits or chips within a device. It has become widely adopted since its development in the 1908s due to its simplicity and versatility. The protocol allows multiple devices to communicate with each other over a common bus using just two signal lines- SCL (Serial Clock Line) and SDA (Serial Data Line). It supports various data rates such as standard mode, fast mode, high spe ed mode etc.

I2C uses a master-slave architecture. The communication is initiated by the master device which controls the bus and generates the serial clock signal. This signal is also used to synchronize data transmission between the master and the slave devices. The serial data line is used to send data in both directions. The SDA line is bidirectional which means that the master and slave devices can both send data on the line. The communication is half duplex where only a master or a slave device can send data on the bus at a time. A master device starts and stops the communication on the bus which removes the potential problem of bus contention. Also, the communication happens via a unique address on the bus which allows multiple master and multiple slave devices on the bus.

2.6.1 I2C Interface

To communicate over the bus, a slave device needs to be addressed by the master. Each device on the I2C bus has a specific device address to distinguish themselves from other devices which are on the same bus. Many devices require configuration after startup. This is done by the bus masters accessing the internal register maps of the devices. The internal register maps of the slave devices have unique addresses. Depending on the application, a device can have one or multiple registers where data is stored, read from or written to.

The SCL and SDA lines must be connected to the power supply through pull-up resistors. The size and value of the pull-up resistor depends on the amount of capacitance on the I2C bus lines. Data transfer can be initiated when the bus is idle

and the bus is considered idle if both the SCL and SDA lines are high after a STOP condition. The general process for a master to access a slave device consists of a number of steps. Before moving on to the data transfer between a master and a slave device, it is necessary to know the associated concepts.

START and STOP condition

Communication with a slave device is initiated by the master sending a START condition and terminated by the master sending a STOP condition. A HIGH to LOW transition on the SDA line while the SCL line is HIGH defines a START condition. Similarly, a LOW to HIGH transition on the SDA line while the SCL line is HIGH defines a STOP condition.

Repeated START condition

A repeated START condition is similar to the START condition and is used when there is back to back data transfer between the master and the slave. In other words, when the master wants to start a new communication but doesn't want to release the bus, it generates a repeated START condition instead of a START condition. It might look identical to the START condition but actually it is different because it happens before a STOP condition when the bus is not idle. This repeated START condition is really useful in case of a system where there are multiple masters. For example, If one master is done with the communication and let the bus go idle by asserting the STOP condition, another master might take control over the bus for communicating with the slave device(s) and the first master has to wait till the second master finishes the communication.

Acknowledge (ACK) and Not Acknowledge (NACK)

During data transfer, each byte of data is followed by one bit of ACK from the receiver. The receiver can be either the master or the slave. For the receiver, the ACK bit is a way of telling the transmitter that the data byte was successfully received and another byte might be sent. Before the receiver can send an ACK, the transmitter must release the SDA line. To send an ACK bit, the receiver pulls down the SDA line. When the SDA line remains high during the ACK/NACK related clock period, it is interpreted as a NACK. The NACK condition can be generated if the receiver doesn't understand the command that it gets from the master or if the receiver can't receive any more data bytes or if the receiver is busy and performing some other functions, thereby not ready to start communication with the transmitter.

Writing to a Slave Device

To write to a slave device of the I2C bus, the master sends a START condition on the bus with the slave's device address. Usually, this device address is of 7 bits. It is followed by a R/$\overline{W}$ bit signifying a data read or data write operation. For read or write operation bit 1 or 0 is sent, respectively. In this case, the device address followed by a bit 0 is sent, indicating write operation. After the slave sends the ACK bit, the master sends the register address of the register it wants to write to. The slaves sends the ACK bit again, letting the master know it is ready to receive the

next data byte. Then, the master starts sending the data to be written to the register. After this data byte, the slave sends the ACK signal again and the master terminates the transmission with a STOP condtion. Figure 2.8 shows an example of I2C write operation to a register of a slave device.

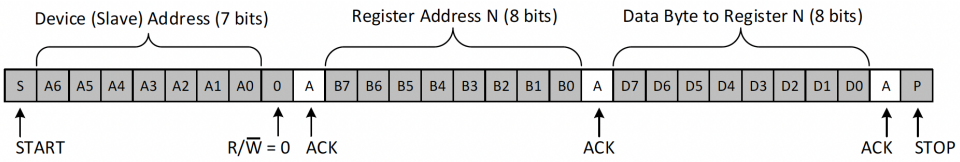

FIGURE 2.8: Example of I2C Write Operation to a Slave Device Register

Reading From a Slave Device

Reading from a slave device is similar to the writing but with a few extra steps. The master starts off by sending a START condition followed by the device address of the slave device and a bit 0 indicating which specfic device it wants to perform the read operation from. Upon receiving the ACK from the slave, the master lets the slave know which register it wants to read from by sending the address of that particular register. Once the slave acknowledges this register address, the master sends the START condition again, followed by the slave address with the R/$\overline{W}$ bit set to 1 indicating read operation. After getting acknowledgement from the slave, the master releases the SDA line but continues to supply clock to the slave, so that the slave can transmit data. Now, the master becomes the master-receiver and the slave becomes the slave-transmitter. At the end of every data byte, the master sends an ACK to the slave, letting the slave know it is ready for more data. Once the master receives the number of data bytes it is expecting, it signals the slave to pause the communication and release the bus by sending a NACK signal. Then, the master generates the STOP condition in the end.

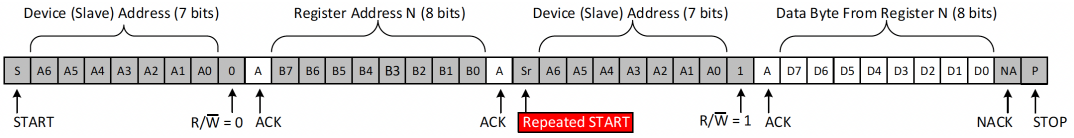

FIGURE 2.9: Example of I2C Read Operation from a Slave Device Register

Chapter 3

Design Approach

3.1 History

This work is a continuation of "Panoptes delay asymmetry compensation logic". The delay asymmetry compensation logic has the goal of increasing the accuracy of distance measurement to a value smaller than one centimeter which is achieved by integrating a TDC with temporal resolution below 100ps. As described in the delay asymmetry compensation logic document[3], an indirect approach is followed to measure T_d by measuring time-current areas or charges. It is done by applying two window functions to the photocurrent at the receiver and the result can be integrated as shown in figure 3.1. Windowing the light pulse is achieved by using a shutter while the integration of the time-current area is accomplished by the accumulation of charges in the pixel behind the shutter which is an electronic switch. The charges are collected by shifting the shutter pulse relative to the laser pulse.

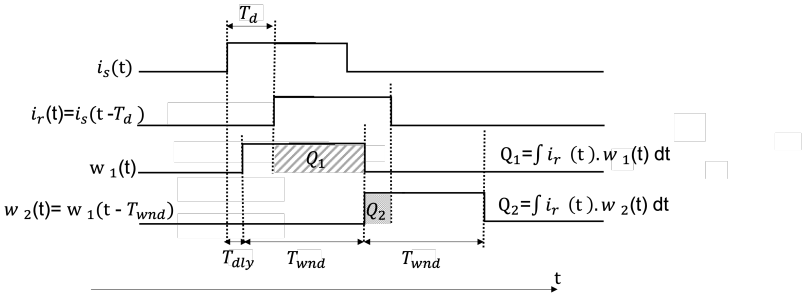

FIGURE 3.1: Windowing of Received Light for Indirect Measurement

The relationship between T_d, T_{dly} and the charges acculumated in the window functions is given by the following equation:

$$T_d - T_{dly} = T_{wnd} \cdot \frac{Q1}{Q1 + Q2} \tag{3.1}$$

From this above equation, T_d can be measured if the time delay T_{dly} is known which is basically the delay between the laser emitting light and the shutter opening. So, the primary focus of the delay asymmetry compensation logic is to calculate the T_{dly}. Theoretically, if the light pulse was emitted exactly at the same time when shutter opens, T_{dly} would have been 0 and would have been eliminated from the above equation. Even if the drivers for the shutter and the light source get driven at the same time, their outputs don't react concurrently. In the actual circuit, the light source consumes a lot of power and the drivers associated with it are large in comparison with the drivers for the shutter. So, this T_{dly} is the result of these drivers and changes with different operating conditions. So, the delay asymmetry compensation

logic contains both a measurement circuit which calculates the T_{dly} and a compensation circuit which increases or decreases the T_{dly} in order to bring it to a set point or reference value. The time difference between the shutter opening and the laser emitting light is fed to a TDC. The TDC converts this time difference to digital output signals which is then measured by the digital logic and the difference between the measurement and the reference value, known as the error value, is calculated. This error value is fed to a PID compensator which tries to minimize it. This PID compensator output is fed to a variable delay which in turn increases or decreases the T_{dly}.

A block diagram of the delay asymmetry compensation logic is shown in figure 3.2. Some of the control signals are omitted from the figure for clarity and comprehensibility.

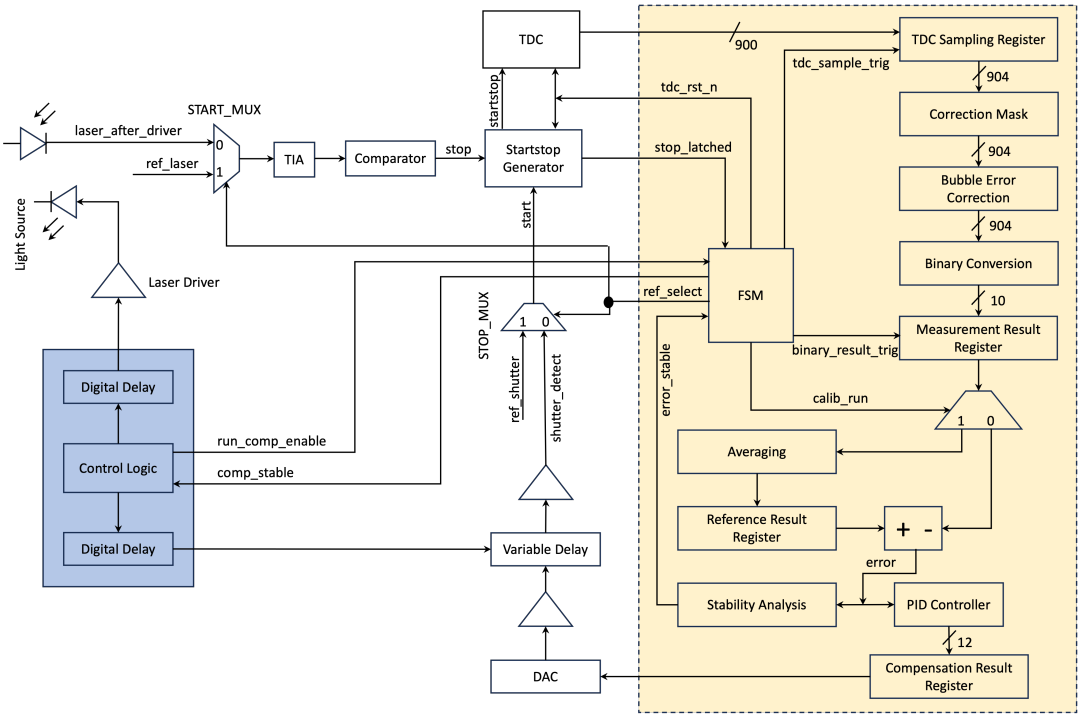

FIGURE 3.2: Block Diagram of Delay Asymmetry Compensation Logic

The control logic for the camera developed by Elmos Semiconductors triggers the shutter opening and light emission. It contains digital delays for both the signals which can be used to adjust the relative timing of these two signals down to 1ns resolution. The light emission trigger signal is amplified off-chip and fed to the LED. A photodiode placed in the proximity of the LED captures the light as soon as it's emitted. The output of the photodiode is passed through a transimpedance amplifier and then through a comparator circuit which produces the stop signal for the startstop generator. The start signal for the startstop generator is produced by the variable delay.

The startstop generator runs at a clock frequency of 80 MHz. It combines the start and stop signals into a single startstop signal, the pulse width of which must be the difference between the rising edges of start and stop. It also generates *stop_latched*

signal which is used by the main FSM of the digital part to recognize the completion of a measurement. Since, the digital part runs at a clock frequency of 20 MHz, to avoid metastability because of clock domain crossing, *stop_latched* is passed through multiple flip-flops for clock synchronization.

The startstop signal is then fed as an input to the TDC. As described previously, the TDC measures the time interval between the start and stop pulses and produces a digital code representation of that time interval as output. This digital output of 900 bits is fed into the digital part for further processing.

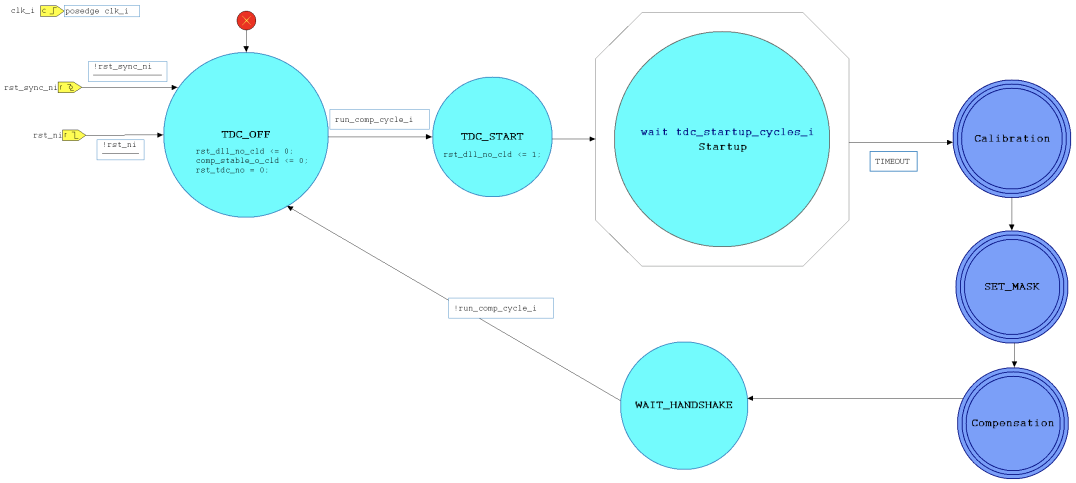

FIGURE 3.3: Main FSM of the Control Loop

The delay compensation FSM is the main control unit which carries out all the steps like calibration, setting of mask bits, compensation etc. by keeping a balanced coordination among all the other modules of the digital part. The diagram of the FSM is shown in figure 3.3. It generates all the necessary control signals for the operation of other modules. When powering up the camera, a change in temperature is detected or a certain time period has elapsed. It is assumed that T_{dly} might have changed from its last known value and requires recompensation. At that point, the camera control logic, shown by the light blue block in figure 3.2, initiates the delay compensation procedure by asserting the signal *run_comp_enable* to the control loop which is shown in light yellow block in figure 3.2. The FSM powers up the TDC and waits for it to settle. After the counter expires indicating the end of the startup cycle period, the system enters the calibration state. During this phase, the system is calibrated by taking a set of reference pulses into consideration. The reference pulses are generated by the edge generator module which is described in section 3.13.1. During the calibration process, the FSM generates the output signal *ref_select* which is at a logic HIGH state and given as an input to the multiplexers START_MUX and STOP_MUX as shown in figure 3.2. Therefore, the outputs of the multiplexers are the reference pulses. The reference pulse *ref_laser* from the STOP_MUX is passed through the TIA and the comparator and the *stop* signal generated by the comparator is fed as one of the inputs to the startstop generator. Similarly, the reference pulse *ref_shutter* is passed through the START_MUX and provided as the *start* signal to the startstop generator. Along with the *startstop* signal, the startstop generator also generated the signal *stop_latched* which is given as an input to the main FSM. After the

assertion of the *stop_latched* signal, the FSM waits for some time and then generates the output signal *tdc_sample_trig* which is an input to the TDC sampling register.

As described earlier, the *startstop* signal generated by the startstop generator is passed through the TDC which generates the digital code representation of the time interval defined by the startstop signal. This digital code is fed to the TDC sampling register which samples this code and synchronizes it to the 20MHz clock based on the *tdc_sample_trig* signal from the FSM. Then, the digital code is passed through the correction mask module. During the calibration process, the correction mask module doesn't generate any mask bits and provides the digital code output of the TDC sampling register as input to the bubble error correction module. The bubble error correction mechanism is described in detail in "Panoptes delay asymmetry compensation logic" document[3]. Then, the digital code is checked for the bubble error correction. The bubble errors are eliminated, if there are any. Then, the number of 1s in the digital code are calculated. As described in section 2.5 of chapter 2, the longer the pulse width of the time interval is, the more flip-flops of the TDC change state from 0 to 1. So, the count of number of 1s represent the measurement value. In the meantime, the FSM generates the *binary_result_trig* signal which assigns the measurement value to the measurement result register. The measurement results are averaged over multiple measurements of the reference pulses and thereby, the set point of the compensation loop for the current operating condition is established. Then, the calibration process gets completed and the *calib_done* output signal is asserted by the main FSM. Also, the *ref_select* signal is assigned to a logic LOW state. The state diagram showing the calibration procedure is presented in figure 3.5.

After the calibration process is complete, the system goes into the SET_MASK state. This state generates the control signals necessary for the generation of the mask bits in the correction mask module. The state diagram showing the generation of these control signals is presented in figure 3.4. These mask bits are added to the digital code to mitigate the double sampling or blind spot effect observed in the TDC. It can be observed from figure 3.4 that if there is no such phenomenon in the TDC, then the system exits the SET_MASK composite state after passing through the INIT_MASK state. This transition has a higher priority than the other transition from the INIT_MASK state. The concepts of double sampling and blind spot are explained in section 2.5.2 of chapter 2.

After that, the system goes into the compensation process. During the compensation process, the *ref_select* signal is deasserted by the FSM. Therefore, the signals *laser_after_driver* and *shutter_detect* are passed through the START_MUX and STOP_MUX, respectively. After the *stop_latched* signal is asserted by the startstop generator, the digital code output of the TDC is sampled in the TDC sampling register based on the *tdc_sample_trig* control signal. Then, the mask bits generated in the SET_MASK state are added to the sampled digital code. The digital code is corrected if there are any bubble errors. After that, the measurement value is assigned to the measurement result register which is controlled by the *binary_result_trig* signal as explained earlier. The difference between the reference value, assigned to the reference result register during calibration process, and the measurement value is calculated and this error signal is passed through the PID controller. Based on the value of this error signal, the PID controller carries out a series of calculations as per the equations explained in section 2.4.1 of chapter 2 and generates the rounded output. This

digital output is passed through a DAC and the analog output of the DAC is used to control the variable delay which increases or decreases the T_{dly} and tries to bring it back to the reference value. Then, the compensation loop repeats itself and a series of measurements are carried out one after the other until the error value is the lowest and can't be minimized any further. At that point, the stability analysis module asserts the *error_stable* output signal to the FSM. In response to that, the FSM generates the *comp_stable* signal and notifies the camera control logic regarding the completion of the compensation process. The TDC is switched off and the delay compensation logic goes into sleep mode. The state diagram showing the compensation procedure is presented in figure 3.6.

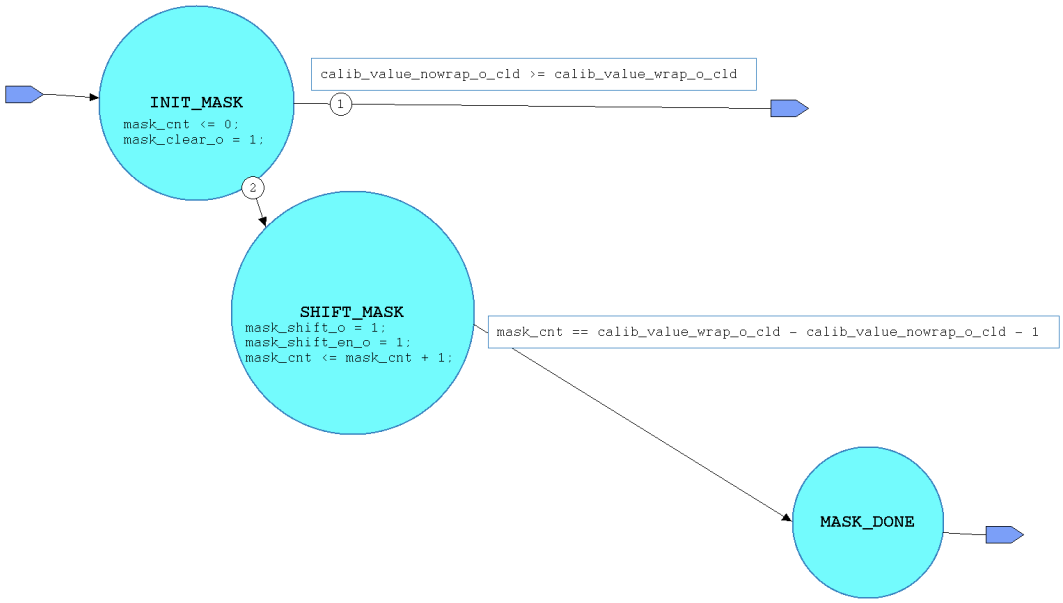

FIGURE 3.4: Generation of Control Signals for the Mask Bits

Note: Except for the synchronous reset (*rst_sync_ni*) and asynchronous reset signals (*rst_ni*), all other signals used in this design are active high. All the design and verification modules are created using Verilog[12][13] and SystemVerilog[14].

3.2 Digital Debugging Features

A part of the scope of this work is to develop the debugging features for the delay asymmetry compensation logic which help in assuring the desired functionality of the digital part of the system and debugging it by moving through the system step by step using the debug mode. The basic idea behind this is to provide an interactive environment between the user and the system. The user can communicate with the system via I2C bus. It is expected that the system operates normally during the standard mode of operation but when the user wants to check whether something is working as per the expectation or not, that can be achieved by asserting the debug mode. For the system operation in debug mode, a dedicated debug state was added to both the calibration and compensation parts of the main FSM. The debug mode can be asserted by asserting the *debug_mode* signal which is accomplished by writing 1 to a particular bit of an I2C register. When the debug mode is asserted, the system halts in the debug state. At that point, the user can decide to be in that state and

examine some particular aspects or functionalities of the design or can decide to get out of the debug state and debug the next measurement data by writing 1 to another bitfield of the same I2C register which in turn asserts the *debug_continue* signal.

3.2.1 List of debugging features

1. Addition of TDC sampling register.
    - In normal mode, TDC raw data is forwarded to the correction mask module and gets processed in the digital part for the measurement of the T_{dly}.
    - In debug mode, the register shifts its contents by 8-bits which is forwarded to the I2C control unit. The shifting only occurs after each byte of TDC data is read by the user by using the I2C register dedicated for the TDC raw data. The I2C control unit generates a signal which is responsible for the shifting of the TDC sampling register. Here, the shifting of the TDC raw data is synchronized with the I2C readout command.
2. Modification of the I2C control unit and generation of signal for the shifting of TDC sampling register such that the raw data can be read out after each byte handover between the digital part and the I2C control unit.
3. Reading out of the measurement result register.
4. Reading out of the reference result register.
5. Reading out of the stability flag register.
6. Reading out of the double sampling/blind spot compensation result.

Reading out of measurement result register, reference result register, stability flag register and double sampliing/blind spot compensation results are implemented in I2C registers explained in section 3.8. Registers reg_00 and reg_01 refer to measurement result register, reg_04 and reg_05 refer to reference result register, reg_08 represents stability flag register and reg_2A & reg_2B & reg_2C & reg_2D & reg_2E represent mask registers for double sampling/blind spot compensation result.

3.3 Addition of DEBUG State in the Main FSM

The principle and working of the main FSM is described earlier in a nutshell. The two most important states of the main FSM are "calibration" and "compensation". These are composite states and have many other states inside them. A pictoral representation of calibration and compensation states are shown in figure 3.5 and figure 3.6, respectively. As described earlier, the system first gets calibrated according to the set point value and then the compensation loop runs. The compensation process tries to increase or decrease T_{dly} to bring it to the set point value by taking measurements continously one after the other until a steady state is reached.

The inclusion of debugging features demands a dedicated **DEBUG** state, both in calibration and compensation composite states. As seen from figure 3.5 and figure 3.6, when *stop_latched* signal goes high, indicating completion of a measurement, *tdc_sample_trig* gets generated and the TDC output contents are sampled. After the number of 1s in the TDC binary representation get calculated, the system can go to

the CALIBRATION_RESET state or COMPENSATE state either directly or through the DEBUG state. The paths through the DEBUG states are given less priority because the system is intended to be in standard mode of operation most of the times. When *debug_mode* signal gets asserted, the system goes to the DEBUG state and halts there. The only way to get out of that state is the deassertion of *debug_mode* signal specifying the standard mode of operation or the assertion of *debug_continue* signal specifying the continuity of debug mode. Therefore, during the next measurement, the system again comes and halts in the DEBUG state until the conditions to get out of that state are satisfied.

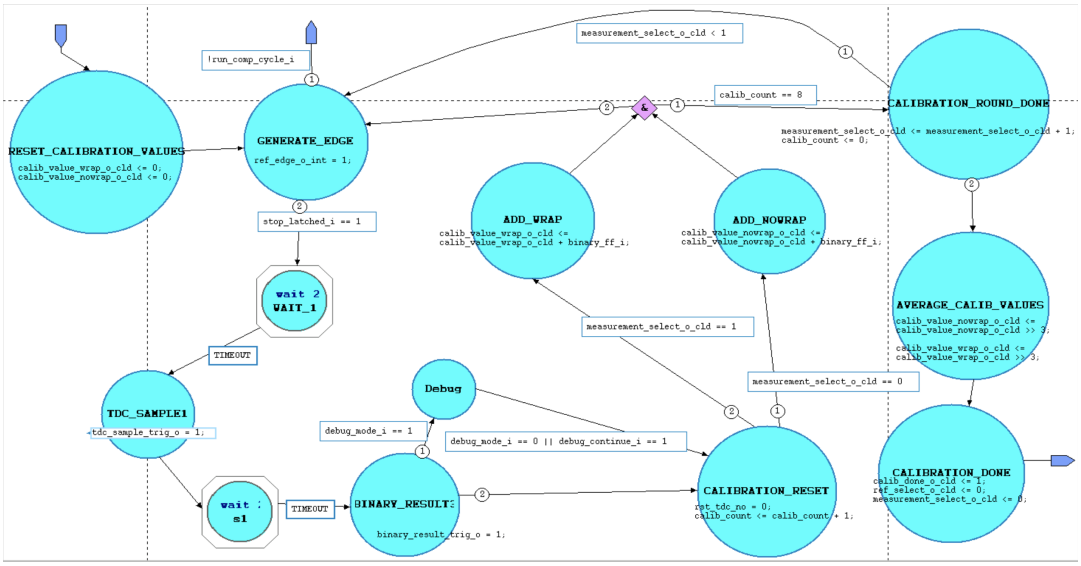

FIGURE 3.5: Inclusion of **Debug** State in Calibration

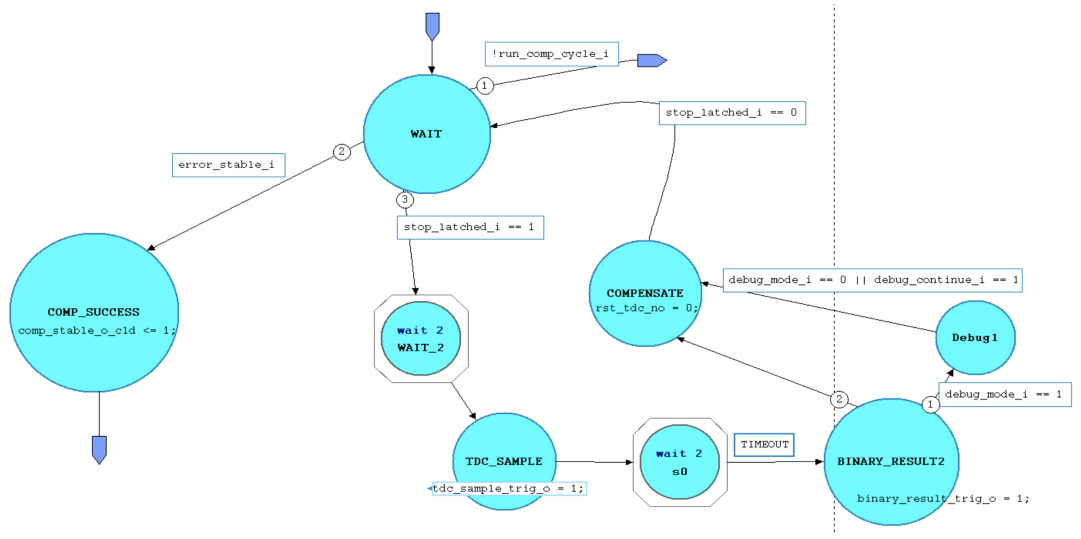

FIGURE 3.6: Inclusion of **Debug** State in Compensation

3.4 Generation of *debug_continue_edge*

This subsection describes how the system continues to be in debug mode even though it gets out of the DEBUG state. How the system gets out of the DEBUG state

by asserting *debug_continue* signal is described in section 3.3. By having a look at figure 3.7, it is clear that the *debug_continue* signal which is an input to the FSM is connected with *debug_continue_edge*, a signal that is generated on the top level of the digital part.

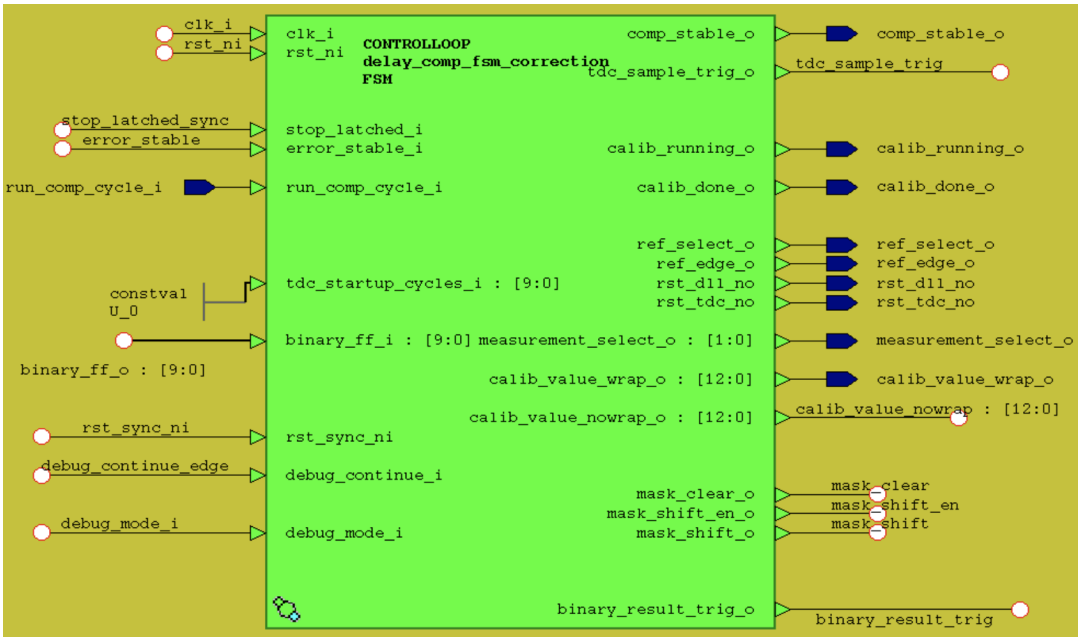

FIGURE 3.7: Block Diagram of the Main FSM

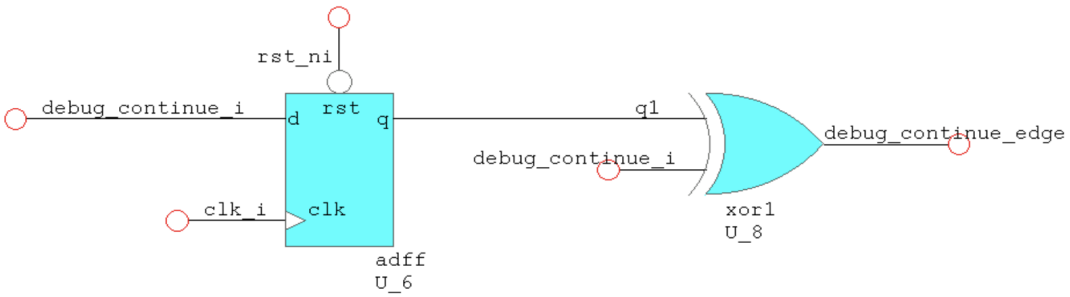

FIGURE 3.8: Generation of *debug_continue_edge* Signal

The user asserts debug mode by writing 1 to the bitfield of **reg_1D** register of the I2C control unit. This register also has another bitfield dedicated for the continuation of debug mode. When that bitfield is written with 1, *debug_continue* signal gets asserted which is an input from the I2C control unit to the digital part. This should not be confused with the *debug_continue* signal of the main FSM because the main FSM signal for the continuation of debug mode is actually *debug_continue_edge* which gets generated from the *debug_continue* signal asserted by the user. In figure 3.8, the *debug_continue* signal is passed through a flip-flop and then a two input XOR gate whose inputs are the *debug_continue* signal itself and the output of the flip-flop. The output of the XOR gate is assigned as *debug_continue_edge*. The *debug_continue* signal is stored in the flip-flop and becomes available at the output of the flip-flop on the next rising edge of the clock. After *debug_continue* signal gets asserted by the user, the XOR gate gets both of its inputs as 1 and produces 0 as output on the next

rising edge of the clock of the digital part. Similarly, when *debug_continue* signal gets deasserted, the XOR gate gets both of its inputs as 0 and produces 1 as output on the next rising edge of the clock. Basically, this whole circuit works as a circuit that toggles the *debug_continue* signal. Simulation results justifying this description are shown in figure 3.9 and figure 3.10.

FIGURE 3.9: Simulation of *debug_continue_edge* - 1

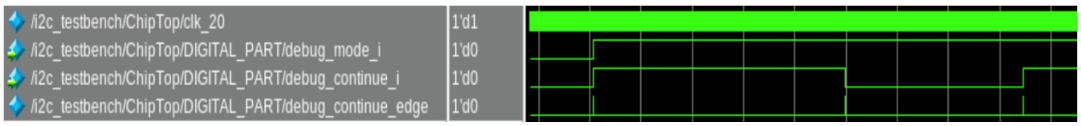

FIGURE 3.10: Simulation of *debug_continue_edge* - 2

In summary, for the system operation in debug mode, the user has to write 1 to the **debug_mode** and **debug_continue** bitfields of the dedicated I2C register. The system goes into the DEBUG state and gets out of that state, initially. But for the next measurement, the system halts at the DEBUG state as *debug_continue_edge* signal has the value 0 because of the toggling circuit. To get out of that state, the user has to write 0 to the debug_continue bitfield of the I2C register. Everytime the user wants to get the debug data for the next measurement, a change of value from the last value of *debug_continue* signal is required. So, the corresponding bitfield of the I2C register needs to be changed by keeping the *debug_mode* at constant 1. When *debug_mode* is changed to 0, the system exits debug mode and works normally in standard mode.

3.5 TDC Sampling Register

As the name defines, the primary objective of the TDC sampling register is to sample the digital output produced by the TDC. Like other modules and components of the digital part, the TDC sampling register is also clocked with the system clock of the digital part, i.e. 20 MHz. This sampling register is designed to behave like a normal register consisting of 900 flip-flops in standard mode of operation. In debug mode, it behaves as a shift register which rotates to the right in each clock cycle. To make the operation easier and convenient, the length of the output of TDC sampling register is designed to be in multiples of 8. This is done by appending 4 bits of 0 before the TDC raw output of 900 bits. This output is also provided as an input to the I2C control unit which gets stored in the **reg_FF_tdc** register. In standard mode of operation, the main FSM generates a control signal *tdc_sample_trig* which is fed as one of the inputs to the TDC sampling register. On assertion of this control signal, the sampling register samples and forwards the TDC raw output data to the correction mask module for further processing. The sampling register also uses another control signal as input known as *rotate_enable* which is used to rotate the contents of the shift register by 8 bits to the right. The *rotate_enable* signal is actually a pulse which remains high for only one clock cycle to ensure shifting of the contents of the sampling register once for one read operation of the **reg_FF_tdc** register.

In summary, when the I2C register dedicated for the TDC raw data is read in the debug mode, the sampling register in the digital part shifts its contents 8 bits to the right and the next TDC raw data byte is available at the I2C register. Implementation of this byte handover between the digital part and the I2C control unit is one of the most important features. The block diagram of TDC sampling register is shown in figure 3.11.

The code snippet regarding the functionality of the TDC sampling block is shown below.

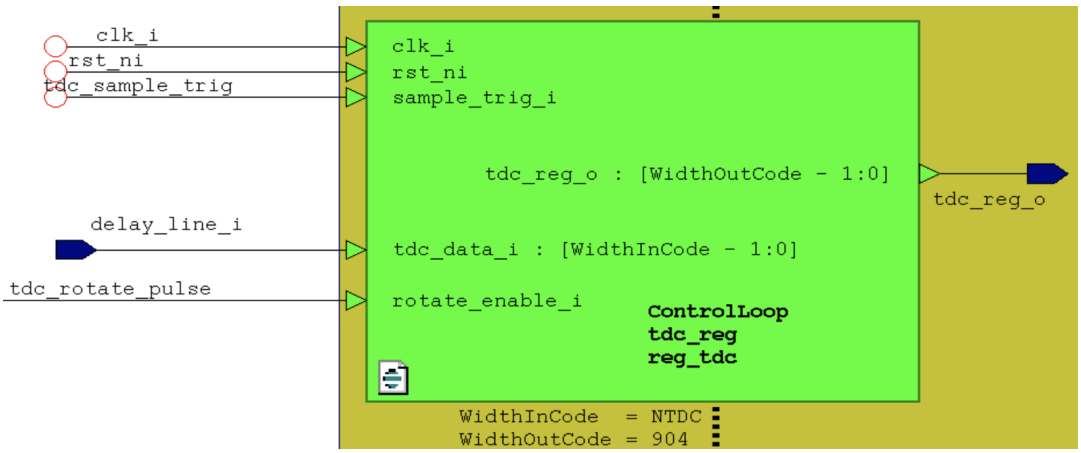

FIGURE 3.11: Block Diagram of TDC Sampling Register

```
always @(posedge clk_i, negedge rst_ni) begin
  if (!rst_ni) begin
    tdc_reg_o <= 0;
  end else begin
    if (sample_trig_i) begin
      tdc_reg_o <= {4'h0, tdc_data_i};
    end else begin
      if (rotate_enable_i) begin
        tdc_reg_o <= {tdc_reg_o[RotationBits-1:0],
                      tdc_reg_o[WidthOutCode-1:RotationBits]};
      end
    end
  end
end
```

3.6 Changes in I2C Control Unit

The primary change which is done in the I2C control unit is the generation of the control signal for the byte handover between the I2C control unit and the digital part. This control signal is given as an input to the digital part from the I2C control unit. In the I2C register map, described in subsection 3.8, a particular register is defined for storing the TDC raw data- **reg_FF_TDC**. The purpose of the control signal is to make the next set of TDC raw data available from the digital part for the next read operation of this register. During debug mode, whenever the user reads the contents of the **reg_FF_tdc** register to get information regarding TDC raw data, the I2C control unit generates the control signal, notifying the readout of the said I2C

register. Since the I2C clock is much slower than the clock of the digital part, to ensure the byte transfer between the I2C control unit and the digital part, this control signal is delayed by one I2C clock cycle to produce *tdc_shift_ctrl_delayed* signal and fed as an input to the digital part. The code snippet for the generation and the delaying of the control signal is shown below.

```
always @(posedge clk_i, negedge rst_ni) begin
  if (rst_ni == 1'b0) begin
    tdc_shift_control <= 1'b0;
  end else begin
    tdc_shift_control <= 1'b0;

    if (data_active_i==1'b1 && read_i && reg_address==8'hFF)
    begin
      tdc_shift_control <= 1'b1;
    end
  end
end

always @(posedge clk_i, negedge rst_ni) begin
  if (rst_ni == 1'b0) begin
    tdc_shift_ctrl_delayed_o <= 1'b0;
  end else begin
    tdc_shift_ctrl_delayed_o <= tdc_shift_control;
  end
end
```

Another major addition to the I2C control unit is the inclusion of additional registers for the implementation of the debug features. The entire I2C register map is defined in section 3.8.

3.7 Generation of *tdc_rotate_pulse*

As described in section 3.6, the *tdc_shift_ctrl_delayed* signal which is an input to the digital part from the I2C control unit, spans over multiple clock cycles of the clock of the digital part. To ensure that the contents of TDC sampling register get shifted by a byte only once for each read operation, a pulse is needed whenever a read operation is performed on **reg_FF_tdc** register of I2C control unit. So, *tdc_shift_ctrl_delayed* signal is passed through a simple state machine and the state machine produces a pulse known as *tdc_rotate_pulse* which is fed as an input to the sampling register. The state machine and the connection between it and the sampling register are shown in figure 3.12 and figure 3.13, respectively. A pictoral representation of the simulation of *tdc_rotate_pulse* generation is shown in figure 3.14.

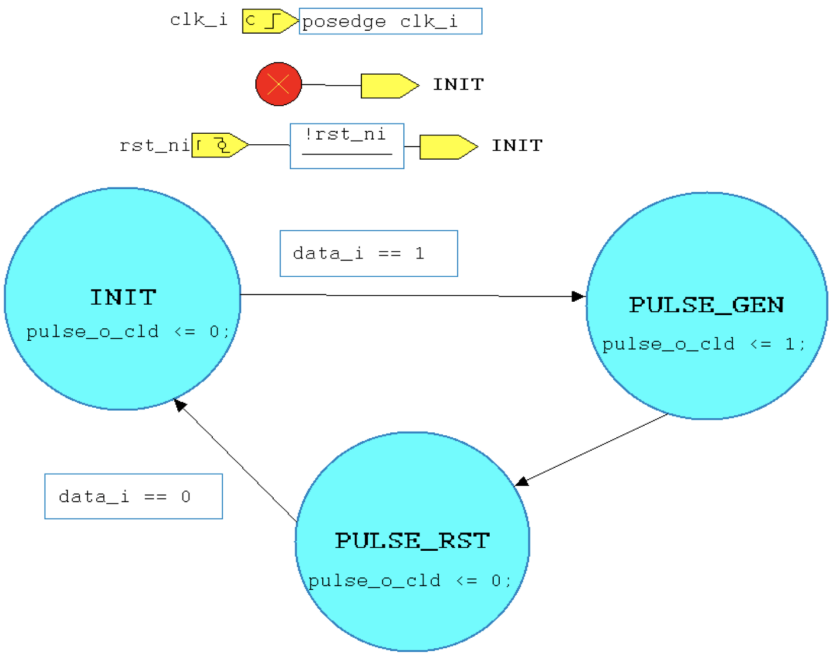

FIGURE 3.12: State Machine Generating Rotation Pulse for Sampling Register

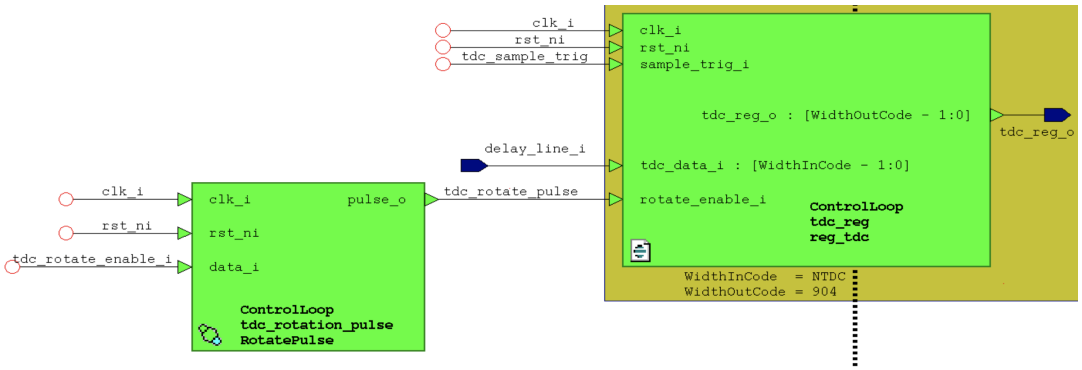

FIGURE 3.13: Connection Between State Machine and Sampling Register

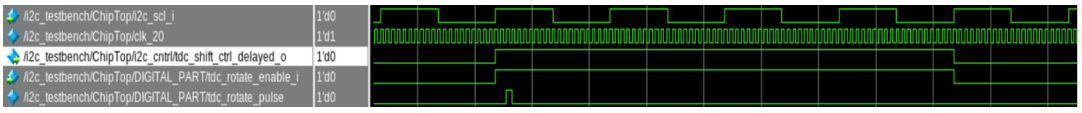

FIGURE 3.14: Generation of *tdc_rotate_pulse*

3.8 I2C Register Map

The register map provides an interactive environment between the user and the system, in this case the testchip. With the help of these registers, the user can configure the chip and communicate with it.

Each target device on the I2C bus has an associated I2C address. The device address of the Panoptes testchip is 0x55, i.e. 1010101. When the user wants to communicate with the testchip, it uses this address to send or receive data in the following I2C frames. For read and write operation, 1 and 0 are appended to the end of this address, respectively. The basic concepts of I2C protocol is described in chapter 2. The registers used here consist of 8 bit fields. The tables from 3.1 to 3.29 contain all the registers and their bitfields used for communication between the user and the Panoptes testchip. The default or reset value of all the registers is 0x00. Registers from address 0x00 to 0x08 & 0xFF are read-only registers and from 0x11 to 0x1D are write only registers from the user point of view.

Register Name	Address
reg_00	0x00
reg_01	0x01
reg_02	0x02
reg_03	0x03
reg_04	0x04
reg_05	0x05
reg_06	0x06
reg_07	0x07
reg_08	0x08
reg_2A	0x2A
reg_2B	0x2B
reg_2C	0x2C
reg_2D	0x2D
reg_2E	0x2E
reg_FF_tdc	0xFF
reg_11	0x11
reg_12	0x12
reg_13	0x13
reg_14	0x14
reg_15	0x15
reg_16	0x16
reg_17	0x17
reg_18	0x18
reg_19	0x19
reg_1A	0x1A
reg_1B	0x1B
reg_1C	0x1C
reg_1D	0x1D

TABLE 3.1: I2C Registers for Panoptes Testchip

The individual bitfields of each registers are presents in below tables.

Bit	Name	Access	Description
7:2	-	R	-
0:1	binary_ff_i [9:8]	R	Measurement result register

TABLE 3.2: Bit-fields of reg_00

Bit	Name	Access	Description
7:0	binary_ff_i [7:0]	R	Measurement result register

TABLE 3.3: Bit-fields of reg_01

Bit	Name	Access	Description
7	calib_running_i	R	Calibration running
6	calib_done_i	R	Calibration done
5	-	R	-
4:0	calib_value_wrap_i [12:8]	R	Wrapping measurement value

TABLE 3.4: Bit-fields of reg_02

Bit	Name	Access	Description
7:0	calib_value_wrap_i [7:0]	R	Wrapping measurement value

TABLE 3.5: Bit-fields of reg_03

Bit	Name	Access	Description
7:5	-	R	-
4:0	calib_value_nowrap_i [12:8]	R	Reference result register

TABLE 3.6: Bit-fields of reg_04

Bit	Name	Access	Description
7:0	calib_value_nowrap_i [7:0]	R	Reference result register

TABLE 3.7: Bit-fields of reg_05

Bit	Name	Access	Description
7:5	-	R	-
4:0	pid_y_i [12:8]	R	Output of PID

TABLE 3.8: Bit-fields of reg_06

Bit	Name	Access	Description
7:0	pid_y_i [7:0]	R	Output of PID module

TABLE 3.9: Bit-fields of reg_07

Bit	Name	Access	Description
7:1	-	R	-
0	comp_stable_i	R	Stability flag for compensation

TABLE 3.10: Bit-fields of reg_08

Bit	Name	Access	Description
7:0	mask_i [39:32]	R	Mask bits for TDC reg

TABLE 3.11: Bit-fields of reg_2A

Bit	Name	Access	Description
7:0	mask_i [31:24]	R	Mask bits for TDC reg

TABLE 3.12: Bit-fields of reg_2B

Bit	Name	Access	Description
7:0	mask_i [23:16]	R	Mask bits for TDC reg

TABLE 3.13: Bit-fields of reg_2C

Bit	Name	Access	Description
7:0	mask_i [15:8]	R	Mask bits for TDC reg

TABLE 3.14: Bit-fields of reg_2D

Bit	Name	Access	Description
7:0	mask_i [7:0]	R	Mask bits for TDC reg

TABLE 3.15: Bit-fields of reg_2E

Bit	Name	Access	Description
7:0	tdc_reg_i [7:0]	R	TDC output data byte

TABLE 3.16: Bit-fields of reg_FF_tdc

Bit	Name	Access	Description
7:5	-	W	-
4	sync_rst_o	W	Unused
3	clk80_int_sel_o	W	Unused
2	clk40_int_sel_o	W	Unused
1	error_i2c_sel_o	W	Unused
0	delay_control_i2c_sel_o	W	Unused

TABLE 3.17: Bit-fields of reg_11

Bit	Name	Access	Description
7:4	-	W	-
3:0	delay_control_i2c_o [11:8]	W	Unused

TABLE 3.18: Bit-fields of reg_12

Bit	Name	Access	Description
7:0	delay_control_i2c_o [7:0]	W	Unused

TABLE 3.19: Bit-fields of reg_13

Bit	Name	Access	Description
7:3	-	W	-
2:0	error_i2c_o [10:8]	W	Unused

TABLE 3.20: Bit-fields of reg_14

Bit	Name	Access	Description
7:0	error_i2c_o [7:0]	W	Unused

TABLE 3.21: Bit-fields of reg_15

Bit	Name	Access	Description
7:5	dexp_kp_o	W	Biased exponent of P part of PID
4:0	-	W	-

TABLE 3.22: Bit-fields of reg_16

Bit	Name	Access	Description
7:0	kp_o	W	Mantissa of P part of PID

TABLE 3.23: Bit-fields of reg_17

Bit	Name	Access	Description
7:5	dexp_ki_o	W	Biased exponent of I part of PID
4:0	-	W	-

TABLE 3.24: Bit-fields of reg_18

Bit	Name	Access	Description
7:0	ki_o	W	Mantissa of I part of PID

TABLE 3.25: Bit-fields of reg_19

Bit	Name	Access	Description
7:5	dexp_kd_o	W	Biased exponent of D part of PID
4:0	-	W	-

TABLE 3.26: Bit-fields of reg_1A

Bit	Name	Access	Description
7:0	kd_o	W	Mantissa of D part of PID

TABLE 3.27: Bit-fields of reg_1B

Bit	Name	Access	Description
7:5	dexp_y_o	W	Biased exponent of output of PID
4:0	-	W	-

TABLE 3.28: Bit-fields of reg_1C

Bit	Name	Access	Description
7	debug_mode_o	W	Mode of operation- 1: Debug, 0: Standard
6	debug_continue_o	W	Debug mode continuation

TABLE 3.29: Bit-fields of reg_1D

3.9 Stability Analysis

The delay asymmetry compensation logic works by regulating the variable delay to reach a set point value. The purpose of the stability analysis block is to determine whether the system has reached an acceptable steady state or not. If the system reaches an acceptable steady state, then the stability analysis module generates a flag that triggers the FSM. The FSM indicates the control logic regarding the completion of the compensation procedure by asserting *comp_stable* signal.

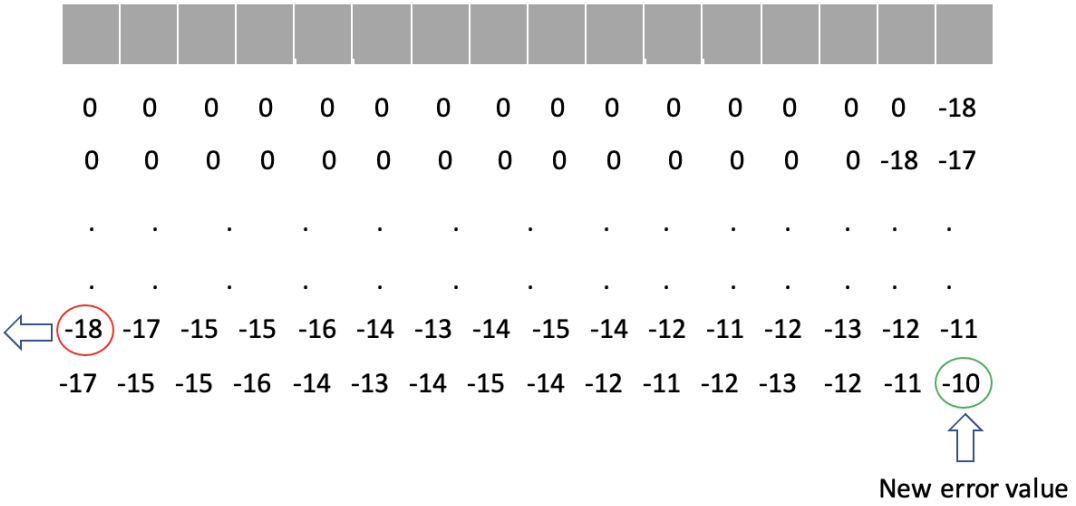

FIGURE 3.15: Concept of Moving Average Window Function

The stability analysis module basically works on the principle of a moving average filter. Moving average calculation is based on the analysis of data points by creating a series of averages of different selections of the complete data set. After each measurement, the error signal is calculated and fed into the stability analysis module as input. The absolute values of the error signals are put inside a shift register the width of which is 16 bits. When a new measurement is carried out and the corresponding error signal gets generated, the data in the shift register gets shifted to the left so that the new error signal could be accommodated at the LSB of the shift register as shown in figure 3.15.

Once the whole register is filled with data, the average is calculated. Every time a new measurement is carried out, the new error signal comes in and the average

changes. A particular average value is set as a parameter, on reaching which the stability analysis module generates the stability flag which is fed as an input to the FSM. A block diagram of the stability analysis module is shown in figure 3.16.

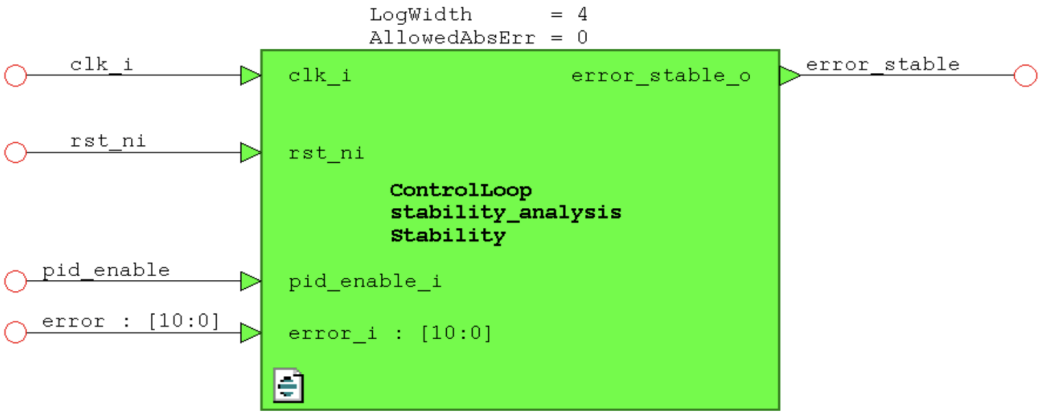

FIGURE 3.16: Block Diagram of Stability Analysis Module

3.10 Redesign of the PID Controller

During the compensation procedure, as new measurements are carried out, the measurement results get stored in the measurement result register. The content of the reference result register is treated as the set point for the compensation. So, for each measurement, the difference between the reference result register and the measurement result register is calculated and is known as the error signal. In other words, the error signal is the difference between the set point value and the binary representation of the time period measured by the TDC. This difference can be positive or negative. So, this input to the PID controller is a signed integer whose bit width depends on the number of TDC output flip-flops. The goal of the PID controller is to minimize this error signal as much as possible so that the system would converge to the set point value. The PID compensator takes the error signal as input along with the proportional, integral, and derivative gain components and produces a signed output. This output is stored in the compensation result register and converted to analog output which in turn controls the variable delay element. The output of the variable delay element controls the opening of the shutter, making the entire design a closed loop system. The derivation of the mathematical formulae and basic description of the PID controller are already given in section 2.4.1 of chapter 2.

As mentioned in table 2.2 in chapter 2, the proportional, integral and derivative gain inputs to the PID controller are K_p, K_i and K_d, respectively. Instead of providing only one input for each of the proportional, integral and derivative parts of the PID controller, two inputs are provided, i.e. one for the mantissa and another for the exponent as explained in section 2.4. In other words, for mantissas m_{K_p}, m_{K_i} and m_{K_d}, the inputs provided are k_p, k_i and k_d, respectively. Similarly, for the biased exponents Δe_{K_p}, Δe_{K_i} and Δe_{K_d}, the inputs provided are dexp_k_p, dexp_k_i and dexp_k_d, respectively.

The PID controller which was previously designed as part of the "Panoptes delay asymmetry compensation logic" had some flaws. The most important flaw which

necessitates the redesign of the controller was the timing problem. Previously, there were timing violations on the **pid_intermediate_reg** which is used for the summation of the outputs of the P, I and D parts of the controller. In the previous design, there was a barrel shifter which was used to do the calculations for the output values of the P, I and D parts of the PID controller and that result was given to the PID state machine for summation, the result of which was the final output of the PID controller after rounding. The barrel shifter took the result of the calculation between the mantissa and the minimum exponent for each of the P, I and D parts of the controller as one input and the biased input as another input. The first input was shifted to the left by the amount represented by the second input according to the equations derived in section 2.4.1 and the result was supplied as an output to the PID state machine. This output result for each of the P, I and D parts was added in the PID state machine and then rounded for the generation of the final PID output. This barrel shifter was a combinational circuit and these set of calculations gave rise to timing problem, creating a critical path in the PID intermediate register which was used to store the sum of the final calculation of the P, I and D parts of the controller.

In the new approach, the calculations of the old barrel shifter are done in a sequential fashion to avoid the timing problem by making the critical path shorter. In this approach, the concept and the inputs and outputs of the shifter module remain the same but the shifting is done only once in a clock cycle and the design follows the synchronous design approach.

This shifter is designed in such a way that it can be used across different designs where any left shift operation between two operands is needed in a synchronous manner. Therefore, it also adheres to the modular and reusable design practice.

To make this modular design adaptable to the Panoptes PID controller design, some changes are made in the PID state machine. In the state machine, the calculation between the mantissa and the minimum exponent is done for each of the P, I and D parts of the controller according to the equations derived in section 2.4.1. The state machine keeps the calulation moving by first doing the computations for the P part and then adding the results from the shifter module for the P part, then doing the similar operations for the I and the D parts and finally generating the output which is the summation of the computation outputs from all the states of the controller. Additional empty states like s0, s2 and s3 are introduced between the P & I, I & D and D & OUTPUT states so that proper handshaking between the state machine and the shifter module can be achieved which is really essential for erroneous free calculations.

The changed PID state machine and the new shifter module are shown in figure 3.17 and figure 3.18, respectively.

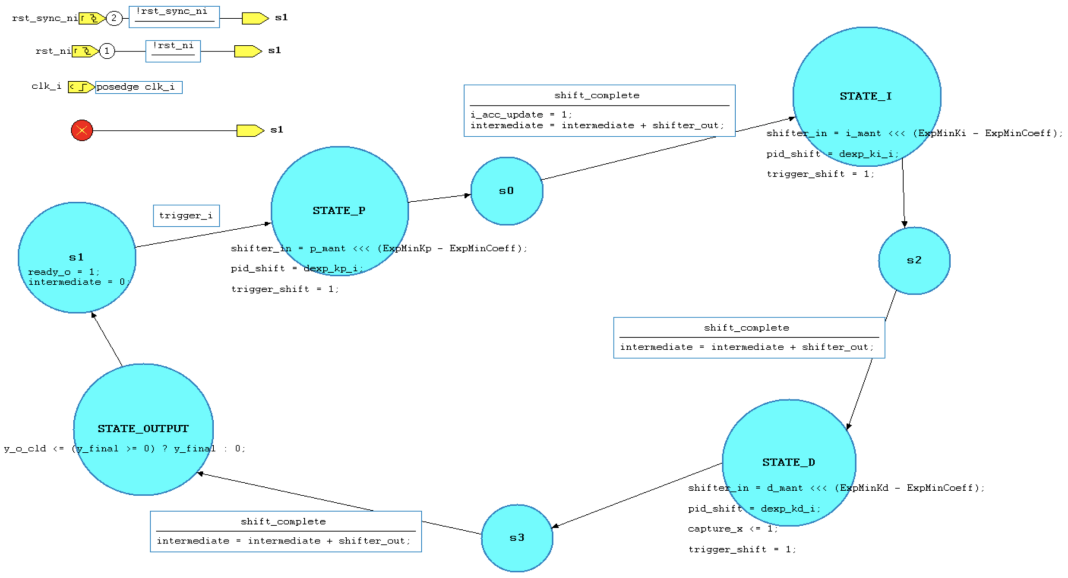

FIGURE 3.17: Changed PID State Machine

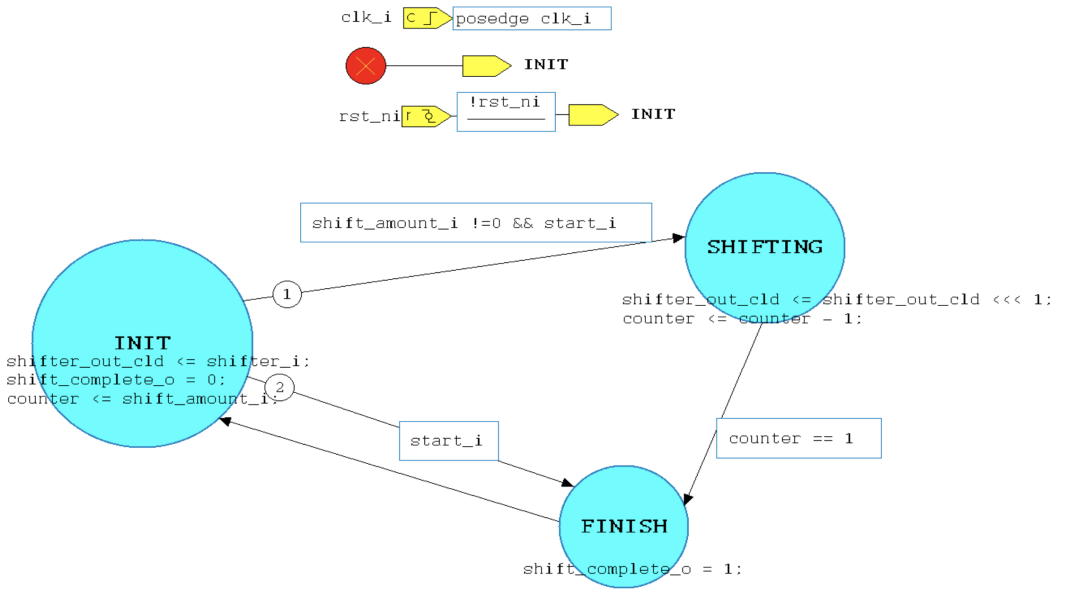

FIGURE 3.18: Shifter Module

3.11 Addition of Bank of Flip-Flops After Error Signal

Previously, it's mentioned that there were timing violations in the PID module. After the addition of the new shifter module, some of the violations are mitigated but not all. It's observed that the path from the generation of the error signal to the shifter module inside the PID controller is too long. The error signal generation is combinational. To make the critical path shorter, a bank of flip-flops is introduced which mitigates the setup violations upto a great extent. The error signal is passed through the bank of flip-flops and the output of the flip-flops is provided as input to the PID controller module. From figure 3.19, it shouldn't be confused as one flip-flop, rather it is a bank of flops used from the "Sequential" section of the "ModuleWare" component in HDL Designer.

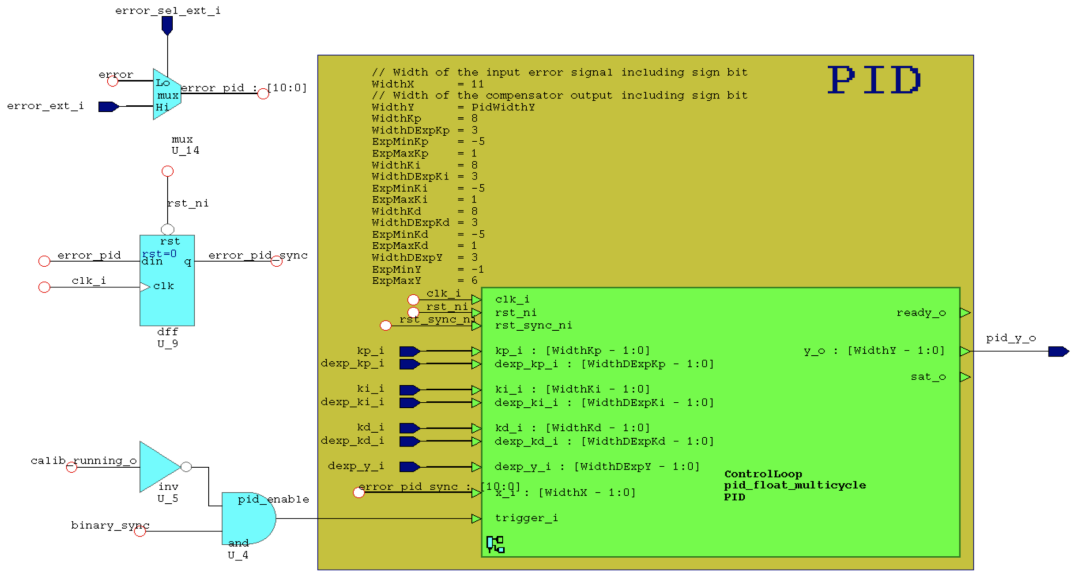

FIGURE 3.19: Bank of Flip Flops After Error Signal Generation

3.12 Addition of Synchronous Reset to Digital Part

As it is explained earlier in this report, the digital part is controlled via I2C bus protocol. But the I2C control module gets the I2C bus protocol inputs from outside the testchip along with the asynchronous reset signal. The asynchronous reset signal is considered as the system reset or global reset input to the testchip which resets all other blocks along with the digital part. To reset only the digital part, a synchronous reset is provided from the I2C control unit to the digital part and can be controlled by the user via an I2C register. The fifth bitfield of the **reg_11** register is used for the same. The synchronous reset is provided as an input to the main FSM and the PID module of the digital part. The FSM generates the *rst_tdc_n* signal as an output. When the fifth bitfield of **reg_11** is written with a value 0, the synchronous reset is asserted which in turn asserts the *rst_tdc_n* reset, therefore ensuring the reset states in TDC model, startstop generator and edge generator modules. Also, the PID state machine goes into reset state, and thereby clearing the contents of the PID accumulator and the PID intermediate register which are responsible for the final PID output. In a nutshell, the main FSM is in reset state, the startstop pulse and the TDC don't generate any output. Therefore, there is nothing to measure.

This synchronous reset is the most useful during power on and debug mode. After the release of the global asynchronous reset, the synchronous reset can be asserted and then the PID parameters can be set by the user by writing to the dedicated I2C registers. Once all the parameters are in place, the synchronous reset can be released leading to a normal system operation. Synchronous reset can also be used when only the digital part needs to be reset, but not all the other blocks.

3.13 New Calibration Approach for the Digital Part

As described earlier, the stop signal for the startstop generator is produced after the output of the photodiode is passed through a transimpedance amplifier (TIA) and a comparator. Upon capturing the light just after its emission, the photodiode

converts it into current. The TIA helps the current generated by the photodiode amplify and convert it to an output voltage. This output of the TIA is fed through a comparator circuit to generate the stop signal. In this application, both the TIA and the comparator circuits are non-inverting amplifiers with unity gains. The previous design didn't include the effect of TIA and comparator circuits. Similarly, the previous design didn't include the effects of the predelay element which is present just before the TDC delay-line as stated in subsection 2.5.1. But in actual practice, the delay associated with the presence of all these components, both individually and combined, affect the design and performance of the calibration logic immensely. Because of these delays, the arrival time of the startstop pulse at the TDC varies. As described earlier in chapter 2, the TDC outputs are very sensitive to the arrival time of the startstop pulse with respect to the clock of TDC which in turn affects the measurement results to a great extent.

It is evident that the compensation loop runs by taking calibration of the TDC as a baseline. Calibration is done by taking wrapping and non-wrapping measurements into consideration. These wrapping and non-wrapping measurements are done based on the arrival time of the startstop pulse at the TDC with respect to the clock. More explanation regarding wrapping and non-wrapping measurements is provided in subsection 3.13.2. Different sets of reference edges are considered for the generation of the startstop pulse. These reference edges occur at different time instances of the TDC clock. Addition of new elements like predelay, TIA and comparator in the design calls for a new calibration approach. To put it briefly, the combination of all the reference edges needs to be evaluated against the delays associated with all these elements to see which range of values of these delay combinations produce the desired TDC output behaviour. The edge combinations also need to be simulated with different parameters of the TDC model used in the design.

3.13.1 Edge Generator

In the previous design approach, the reference edges for calibration procedure were generated by the startstop generator. But the introduction of the TIA and comparator into the design requires a separate module for generation of the reference edges. This is because of the fact that the reference edges have to pass through the TIA and comparator whose delays affect the performance of the design. During calibration, the start and stop signals are generated by the reference edges as explained earlier. Since the startstop generator is clocked with a frequency of 80 MHz, these reference edges need to be generated according to this clock. To figure out the effects of the arrival time of the startstop pulse at the TDC, multiple reference edges need to be generated and simulated against the delays associated with the predelay, TIA and comparator models. Figure 3.20 shows the different edges taken into consideration. Since the full scale output of TDC is 25ns, the edges *edge_1*, *edge_2*, *edge_3* and *edge_4* are of primary interest. After that, the cycle gets repeated. That's why *edge_5* and *edge_6* can be considered as the repeatitions of *edge_1* and *edge_2*. In this document, wherever *edge_5* and *edge_6* are used, those should be understood as *edge_1* and *edge_2* of the next cycle of the 40 MHz clock. Since, a set point value of 12.5ns is chosen for the calibration of the digital part, the edges which are separated by a time interval of 12.5ns are of primary interest. So, the combination of *edge_1* & *edge_3*, *edge_2* & *edge_4*, *edge_3* & *edge_5* and *edge_4* & *edge_6* are considered primarily for the simulations with respect to the delays. The code snippet regarding the generation of the reference edges is given below.

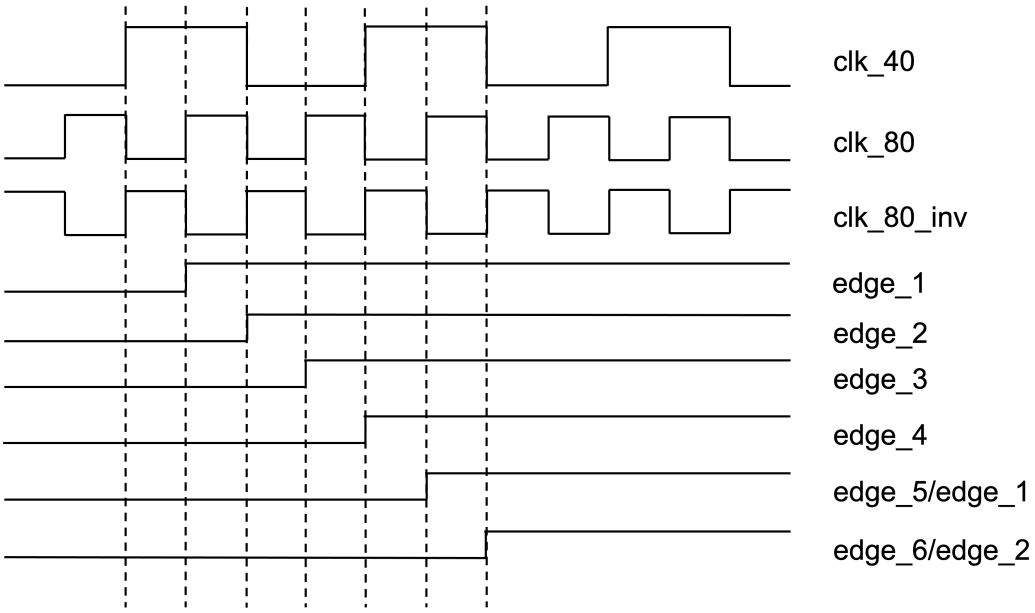

FIGURE 3.20: Generation of Reference Edges

```
always_ff@(posedge clk_80_i,negedge rst_ni,negedge rst_tdc_ni)
  begin
   if (!rst_ni || !rst_tdc_ni) begin
     edge_1_o <= 1'b0;
     edge_3_o <= 1'b0;
     edge_5_o <= 1'b0;
   end else begin
     if (edge_gen_ctrl_i) begin
       if (clk_40_i) begin
         edge_1_o <= 1'b1;
       end

       edge_3_o <= edge_1_o;
       edge_5_o <= edge_3_o;
     end
   end
 end

 always_ff@(posedge clk_80_inv_i,negedge rst_ni,negedge rst_tdc_ni)
  begin
   if (!rst_ni || !rst_tdc_ni) begin
     edge_2_o <= 1'b0;
     edge_4_o <= 1'b0;
     edge_6_o <= 1'b0;
   end else begin
     if (edge_1_o) begin
       edge_2_o <= 1'b1;
       edge_4_o <= edge_2_o;
       edge_6_o <= edge_4_o;
     end
   end
  end
```

The block diagram of the edge generator module is shown in figure 3.21. The *edge_gen_ctrl* input to the edge generator module comes from the digital part. From figure 3.5, the GENERATE_EDGE state of calibration generates the *ref_edge* signal indicating the start of the calibration process. This signal from the digital part is taken as an output and fed to the edge generator module as *edge_ctrl* as shown in figure 3.21.

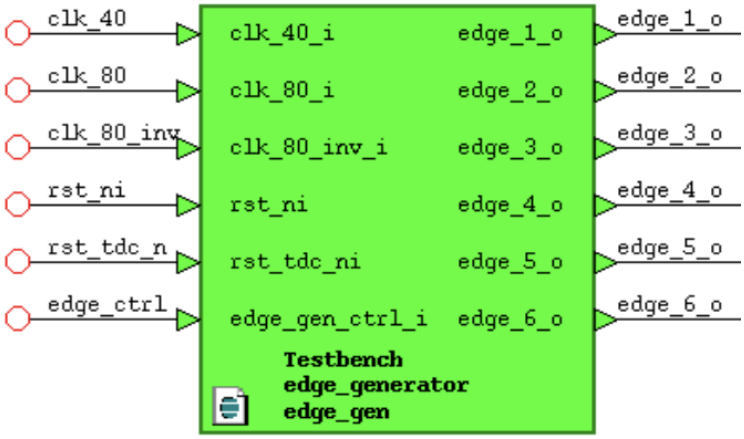

FIGURE 3.21: Block Diagram of Edge Generator Module

3.13.2 Addition of the Predelay element before the TDC

As explained in chapter 2, there are some dummy delay elements placed before the delay-line of the TDC which don't contrinute to the overall delay of the delay-line. The delay associated with these elements has significant impact over the digital calibration. That delay needed to be simulated with the reference edges to gain an idea regarding the effect of it on the measurement outputs of the TDC. A model of the delay element called predelay is created and the TDC clock is passed through that delay element which signifies that the clock to the TDC is not ideal, rather it is delayed by a non-trivial amount of time.

A simulation environment consisting of a stimulus module, the predelay element and the model of the TDC is shown in figure 3.22. The main idea behind this simulation is to find out the effect of different times, at which startstop pulses are sampled with respect to TDC clock, on the measurement results. Reference edges and startop pulses generated by these reference edges are created by the stmuli module, like the edge generator module would generate them in an actual simulation environment. Then those startstop pulses are applied to the predelay element. In the actual simulation environment, the input to the predelay element is the 40 MHz clock and the output from it is the delayed clock which is supplied as input clock to the TDC. There, startstop pulses arrive with respect to the delayed TDC clock. In this simulation, the TDC clock is kept ideal whereas the startstop pulses are delayed to show the sampling of startstop pulses with respect to different phase offsets[7]. TDC outputs for startstop pulses generated by different reference edges are captured in the stimuli module and are written to separate text files. Then, the contents of the text files are plotted in MATLAB[15] to analyze the effects of predelay elements on the measurement results considering different reference edges.

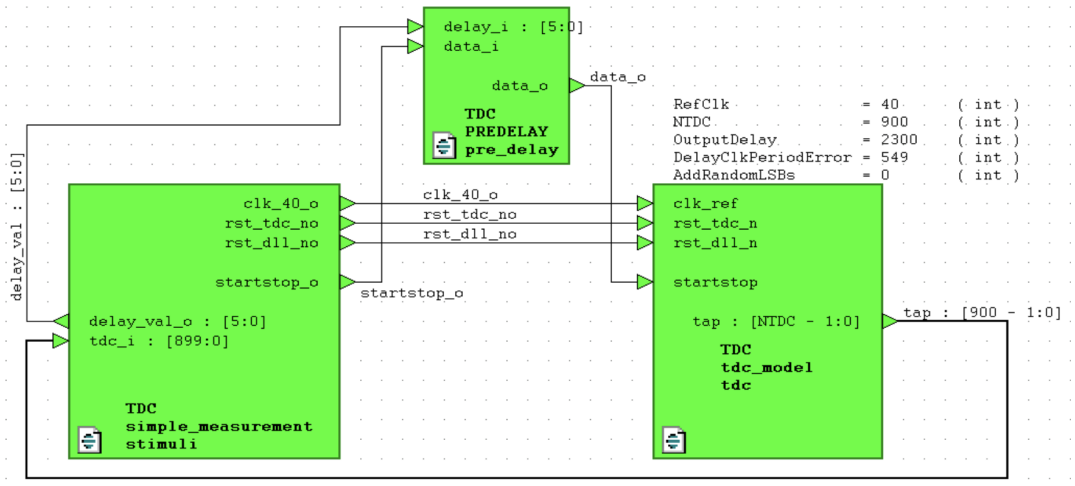

FIGURE 3.22: Simulation Environment of Predelay Element

Before moving on to simulation results, it is essential to understand the concept of wrapping and non-wrapping measurements. From figure 3.21, it can be seen that the startstop pulse generated by *edge_1* and *edge_3* remains within the boundary of one clock period of 40 MHz clock but the pulse generated by *edge_2* and *edge_4* attains zero value just when one clock period of 40 MHz clock is about to end and the the next clock period is about to start. In other words, the negative edge of the startstop pulse generated by *edge_2* and *edge_4* aligns with the negative edge of the 40 MHz clock. But this is an ideal case. For the simulations here, the startstop pulses are delayed by the predelay element. So, the pulse generated by *edge_2* and *edge_4* would wrap around the positive edge of the 40 MHz clock. Similarly, the pulse generated by *edge_3* and *edge_5* would also wrap around. TDC measurements carried out by considering these pulses are known as wrapping measurements. The measurements carried out by taking the pulse generated by *edge_1* and *edge_3* into consideration are known as non-wrapping measurements. Wrapping measurements justify the concept of double sampling effect described in subsection 2.5.2.

The plots of TDC outputs with respect to the delay of the predelay element for startstop pulses generated by different edges are shown in figure 3.23. As explained earlier, the simulation results from the above simulation environment are imported into MATLAB for the generation of these plots. In each plot, the delay of the pre-delay element is swept from 0ns to 5ns in 100ps intervals and the startstop pulse is sampled in every interval. Plot- 1 corresponds to a non-wrapping measurement. So, the output varies within 440 and 441. For the wrapping measurement in plot- 2, the output was initially at a low value of around 440 but as time progresses, the output value increases rapidly and then oscillates between 459 and 460. In the next plot, the output keeps varying between 459 and 460. The increase in the output value from around 440 to around 460 is because of the double sampling effect seen in the TDC. In the final plot, the output value falls down from around 460 and remains at around 440. The MATLAB code for this plot is given in appendix A.

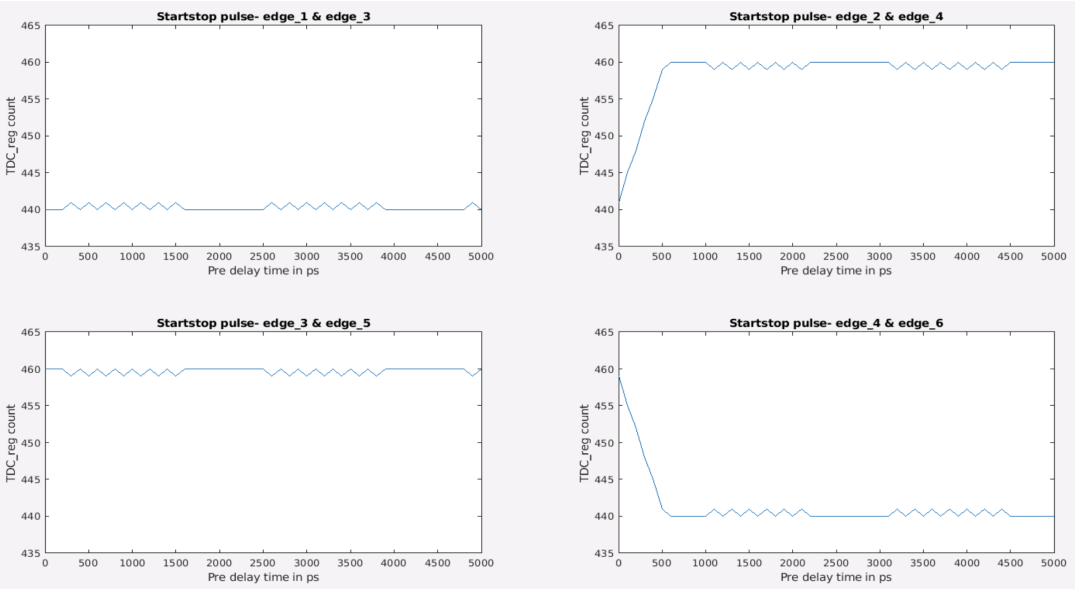

FIGURE 3.23: Simuation of Reference Edges with Predelay Element

3.13.3 Models of TIA and Comparator

Since the TIA and the comparator are analog design blocks. Digital models of these elements are created to simulate the effect of delays associated with these elements on digital calibration. TIA and comparator circuits are high gain amplifiers. So, when the photodiode captures the light emitted from the laser souce, it generates current which is passed through the TIA. The TIA converts this current into a voltage level which is then passed through the comparator circuit. The comparator circuit compares the output voltage of the TIA with a reference voltage. If the input voltage is higher than the reference voltage, then positive saturation voltage is generated as output which is known as the stop signal. The code from the model of TIA and comparator is shown below.

```
‘resetall
‘timescale 1ns/10ps
module non_inverting_amplifier #(
  parameter real DelayNs = 0
)(
    input logic sig_i,
    output logic sig_o
);

  initial begin
    sig_o = 0;
  end

  always @* begin
    sig_o <= #(DelayNs)sig_i;
  end

endmodule
```

3.13.4 Reference Selector

This module selects the reference edges for measurements in the calibration process. After the CALIBRATION_RESET state in figure 3.5, the non-wrapping or wrapping measurement is carried out depending on the *measurement_select* signal. The reset value of that signal is 0, which means that whenever the system starts the calibration process, it always gets started with non-wrapping measurements. After eight consecutive measurements, the results are averaged and stored in the measurment result register. Then, *measurement_select* is incremented and wrapping measurements are carried out.

For wrapping and non wrapping measurements, the startstop pulses need to be generated by different sets of reference edges as explained earlier. The *measurement_select* output from the digital part drives the *sel_measurement* input of the reference selctor module which selects the appropriate reference edges. The code snippet is given below and the block diagram of the reference selector module is shown in figure 3.24.

```
always_comb begin
  case (sel_measurement_i)
    2'b00: begin
      ref_shutter_o = edge_1_i;
      ref_laser_o = edge_3_i;
    end
    2'b01: begin
      ref_shutter_o = edge_3_i;
      ref_laser_o = edge_5_i;
    end
    default: begin
      ref_shutter_o = edge_1_i;
      ref_laser_o = edge_3_i;
    end
  endcase
end
```

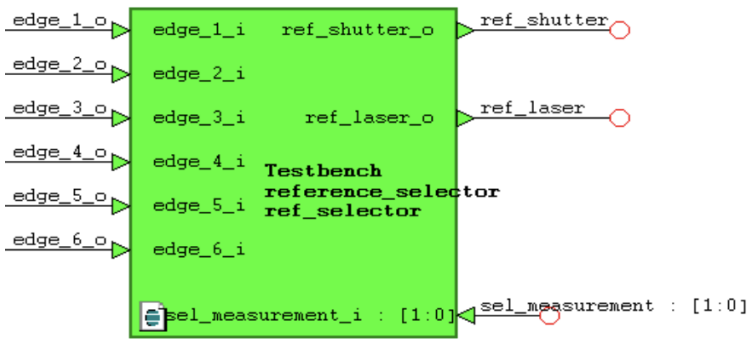

FIGURE 3.24: Block Diagram of Reference Selector Module

3.13.5 Effect of Delays of TIA, Comparator and Predelay on Reference Edges

Simulating reference edges only with predelay element is not enough. To get a deeper understanding of how the TDC outputs behave with different reference edge combinations, the startstop pulses generated by different edge combinations need to

be simulated with the models of the TIA, comparator and predelay altogether. The simulation environment is shown in figure 3.25.

In this simulation environment, the stimulus block generates all necessary signals which would have been generated by the digital part in actual operation. Apart from these signals, it also generates a delay signal which is provided as inputs to the TIA and comparator modules. So, the basic goal of this simulation is to vary the delays of the TIA and comparator models and see the effect of those delays on the TDC outputs. The delay of the predelay element is kept at a value of 3ns. This delay is also varied and many simulations are run to see the combined effect of the predelay, TIA, comparator delay values on the outputs of the TDC model. It can be noticed that the output of the predelay element is fed to the clock of the TDC model as explained before.

The PLL and the power_on_reset modules generate all the clock signals and reset signal necessary for all blocks. The edge generator module generates reference edges which are provided as inputs to the reference selector module. Depending on the value of *sel_measurement* signal from the stimulus module, the reference selector selects the edges. The *ref_shutter* signal is applied at the *start* input of the startstop generator and the *ref_shutter* signal is provided as input to the TIA. The output of the TIA is connected to the input of the comparator module and the comparator generates the *stop* signal which is provided to the startstop generator. By taking *start* and *stop* inputs, the startstop generator generates the startstop pulse and it is provided to the TDC for measurment. The output from the TDC is provided as input to the stimulus block. It captures the measurment results and writes those into a text file.

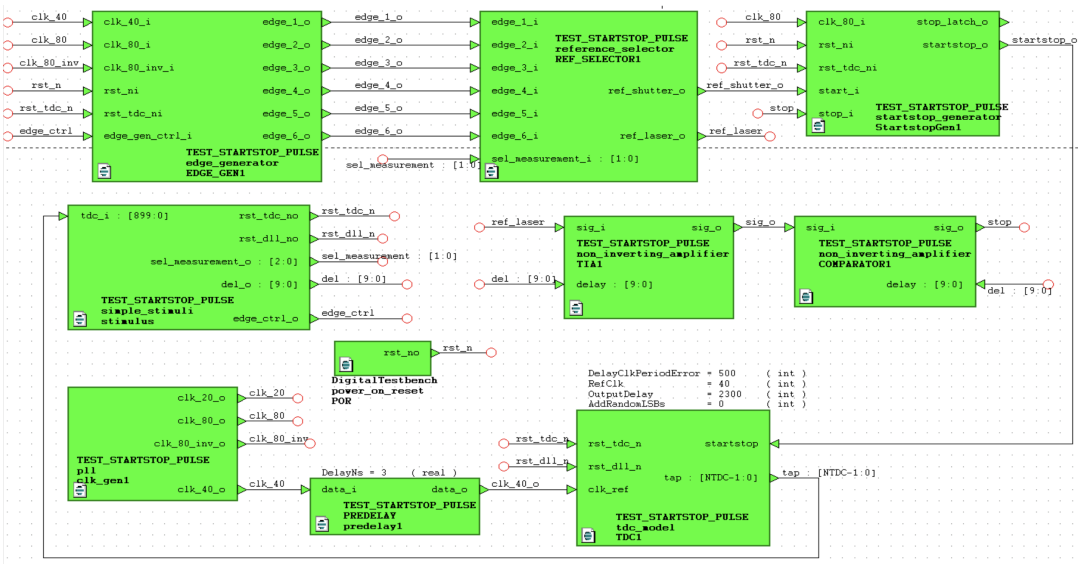

FIGURE 3.25: Simulation Environment Containing All Delay Elements

To plot the measurment results in a more convenient way, the above approach is tweaked a bit. Since, the TIA and comparator blocks are just models of the actual circuits and it is intended to observe the effect of their dalys on the measurement results, a single block having the combined delays of both of these elements can be considered. So, instead of having two separate blocks and two separate delay inputs

to each of these blocks, only one block is considered which gets the combined delays of the TIA and comparator models as one input from the stimulus block. After the stimulus block captures the measurement results, 3D plots are generated in MATLAB where the measurement results are plotted against the delay of the predelay element on one axis and the combined delays of the TIA and comparator on another axis.

The plots for the measurement results of the startstop pulses generated by two specific reference edges were not enough to select the reference edges and decide the range of delay values for the TIA, comparator and predelay elements. To visualize the results more perspicuously, the difference plots for startstop pulses created by two sets of reference edges were plotted. It's evident that the range of the delay values corresponding to the flat area can be taken into consideration. Some difference plots and their respective explanations are presented below.

In figure 3.26, it can be seen that the flat area doesn't start from 0ns for both the axes- delay of predelay and the combined delay of the TIA and comparator. For example, the flat area starts from around 2ns value for the combined delays of TIA and comparator when the predelay value is around 0ns but it starts from around 6ns and lasts upto some value when the predelay value is around 3ns. So, different values of delays of the predelay element put minimum constraints on the combined delays of TIA and comparator which is not good for practical implementations.

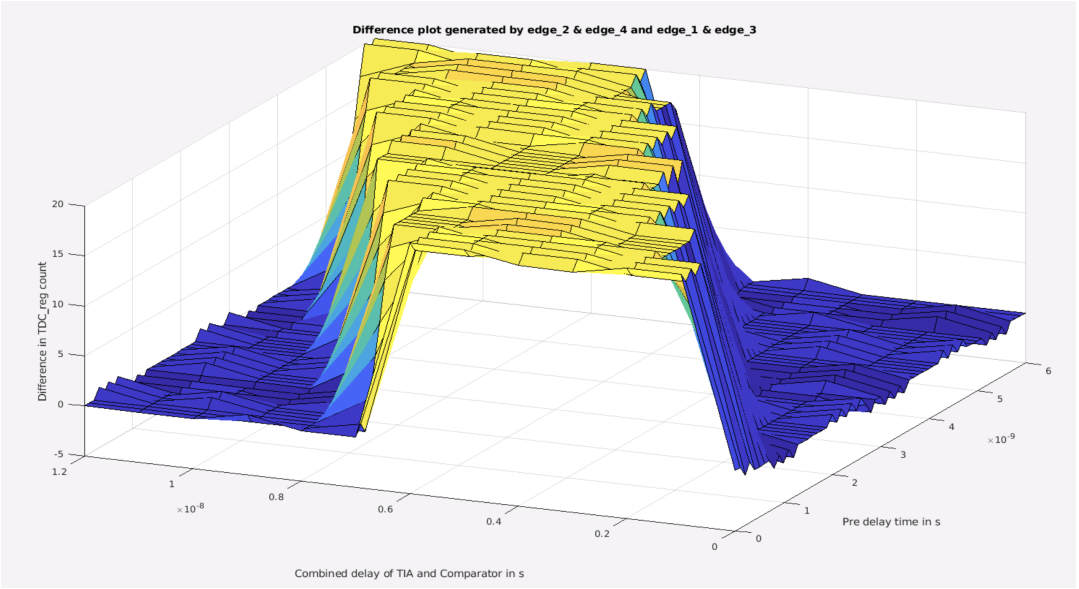

FIGURE 3.26: Difference Plot for *edge_1* & *edge_3* and *edge_2* & *edge_4*

In figure 3.27, the flat area starts from 0ns for both the axes. But it can be seen that the differences in the measurement results for the startstop pulses generated by *edge_3* & *edge_5* and *edge_4* & *edge_2* are always around 0. But there is a need for non-zero difference in the measurement results to show the double sampling behaviour of the TDC. So, these set of reference edges are also not suitable for the new calibration approach.

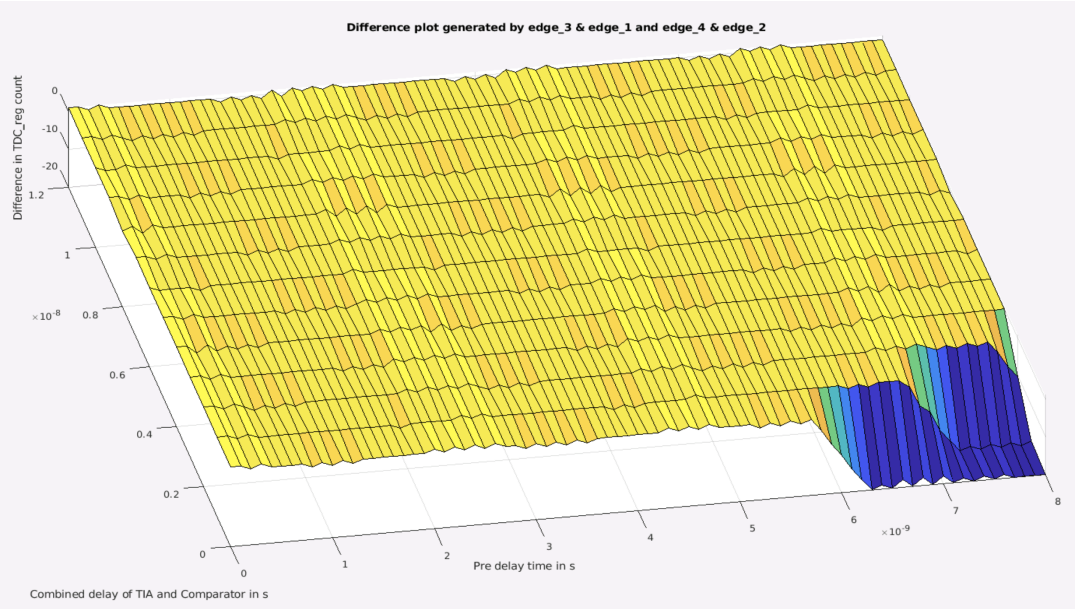

FIGURE 3.27: Difference Plot for *edge_3* & *edge_5* and *edge_4* & *edge_2*

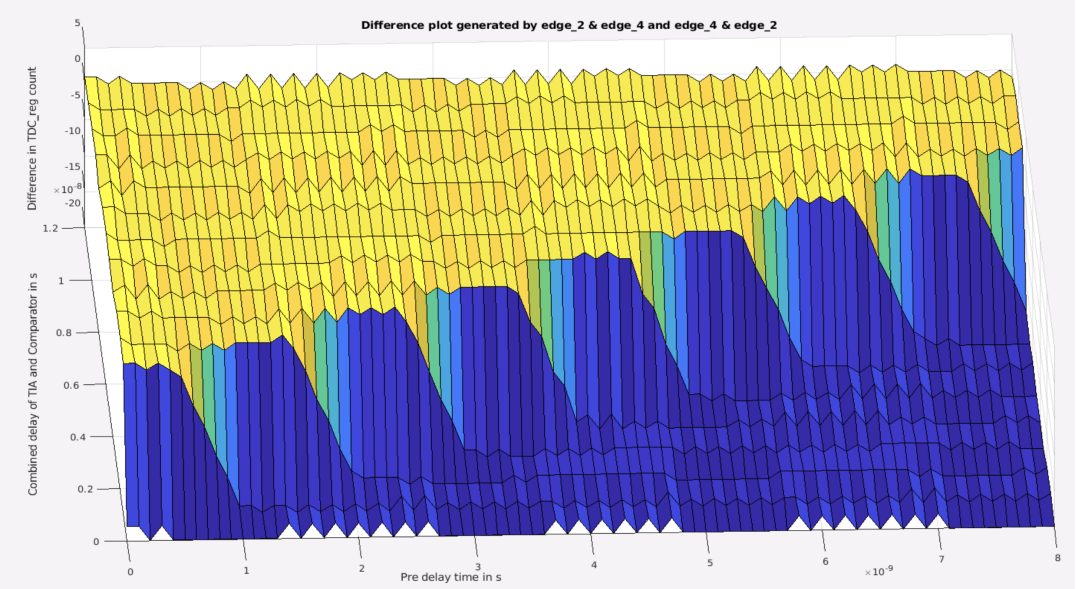

FIGURE 3.28: Difference Plot for *edge_2* & *edge_4* and *edge_4* & *edge_2*

In figure 3.28, the flat area doesn't start from 0ns for the combined delays of TIA and comparator. Also, the flat area corresponds to 0 difference level in the measurement results for the startstop pulses generated by *edge_2* & *edge_4* and *edge_4* & *edge_2*. So, these sets of edges are also discarded.

In figure 3.29, the flat area starts from 0ns for both the axes and lasts upto some value. With increase in the delay value of the predelay element upto 6ns, the value of the combined delays of TIA and comparator corresponding to the flat area also increases from 7ns to 12ns. So, this plot is the most suitable one among all other plots shown previously. From the analog simulations of the TDC, predelay and post delay elements, it is observed that the delay value for the predelay element would

be between 1.5ns to 3ns. From figure 3.30, it can be clearly seen that for a predelay value of around 3ns, the combined delay of TIA and comparator can range from 0ns to 9ns which can be realised in practice.

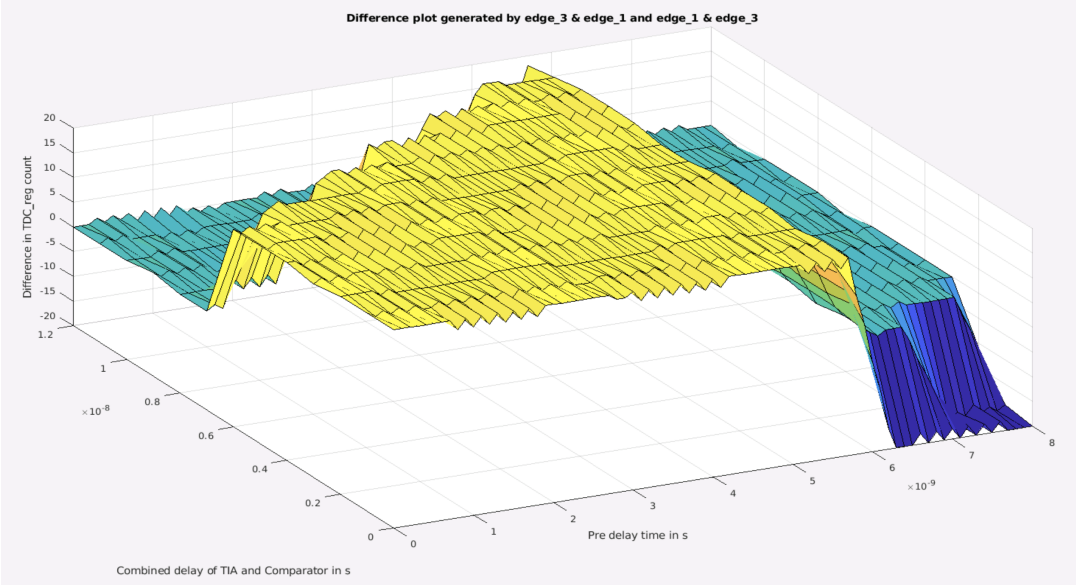

FIGURE 3.29: Difference Plot- 1 for *edge_1* & *edge_3* and *edge_3* & *edge_5*

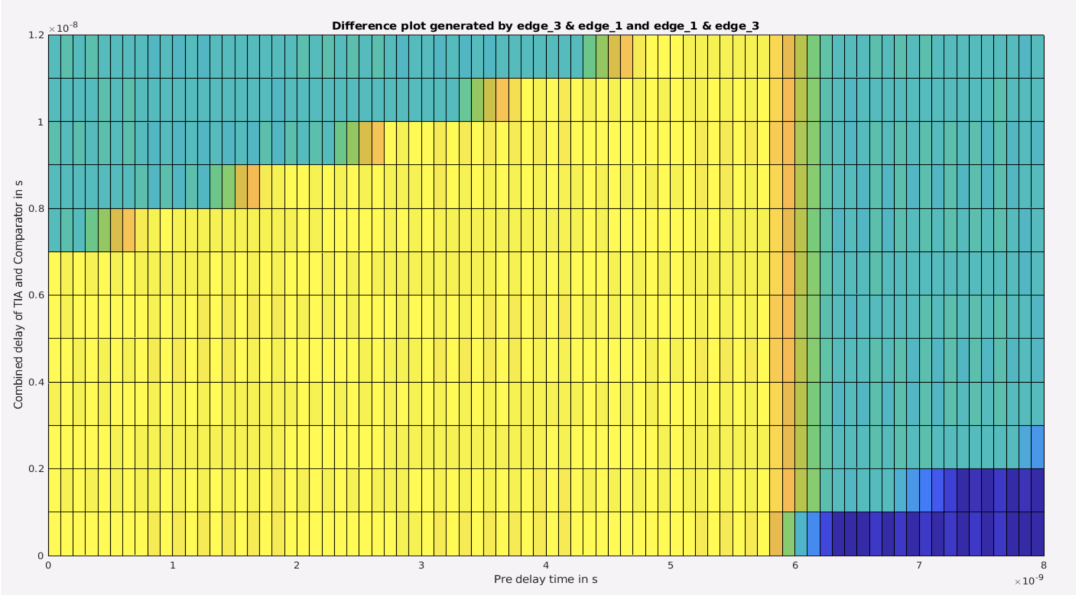

FIGURE 3.30: Difference Plot- 2 for *edge_1* & *edge_3* and *edge_3* & *edge_5*

To consolidate the selection of reference edges, some futher simulations need to be carried out for different settings of the TDC model. From figure 3.22, it can be clearly seen that the TDC model has a number of parameters. The parameters **RefClk**, **NTDC**, **OutputDelay**, **DelayClkPeriodError** and **AddRandomLSBs** are clock frequency in MHz, number of flip-flops of the TDC, delay to produce output after the application of input at the flip-flops of a delay element of the TDC, how slow

or fast the delay-line is and change in the performance of TDC due to any reason, respectively. Among these parameters, **DelayClkPeriodError** has a significant effect on the measurement results. This is in accordance with the fact that the blind spot and double sampling phenomena observed in the TDC are because of how fast or slow the delay-line is. The value of the **DelayClkPeriodError** is changed and some extra plots are generated.

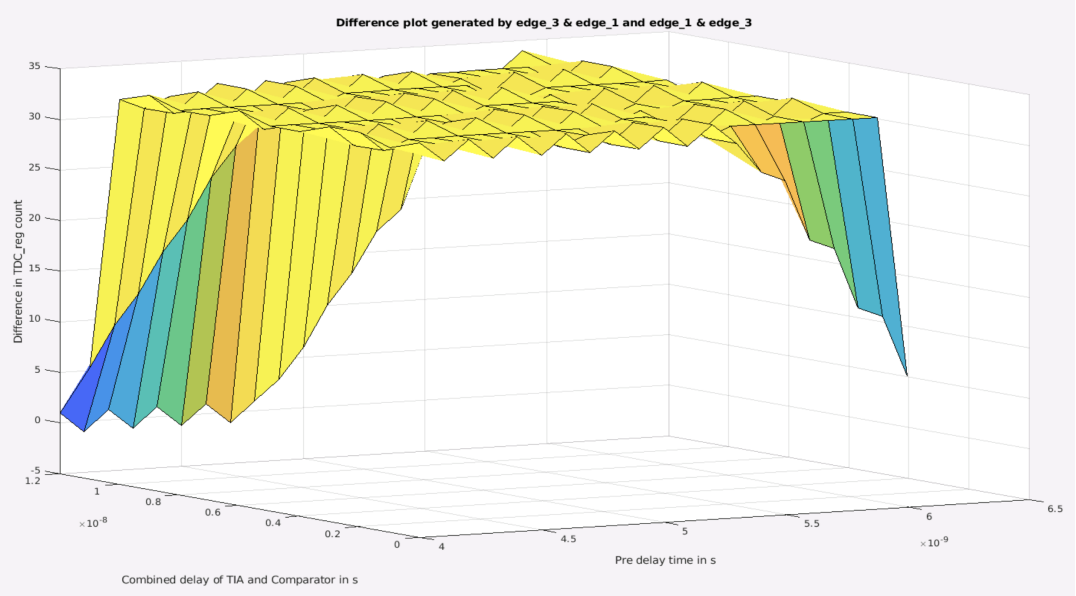

FIGURE 3.31: Difference Plot- 1 for *edge_1* & *edge_3* and *edge_3* & *edge_5* for Different TDC Parameters

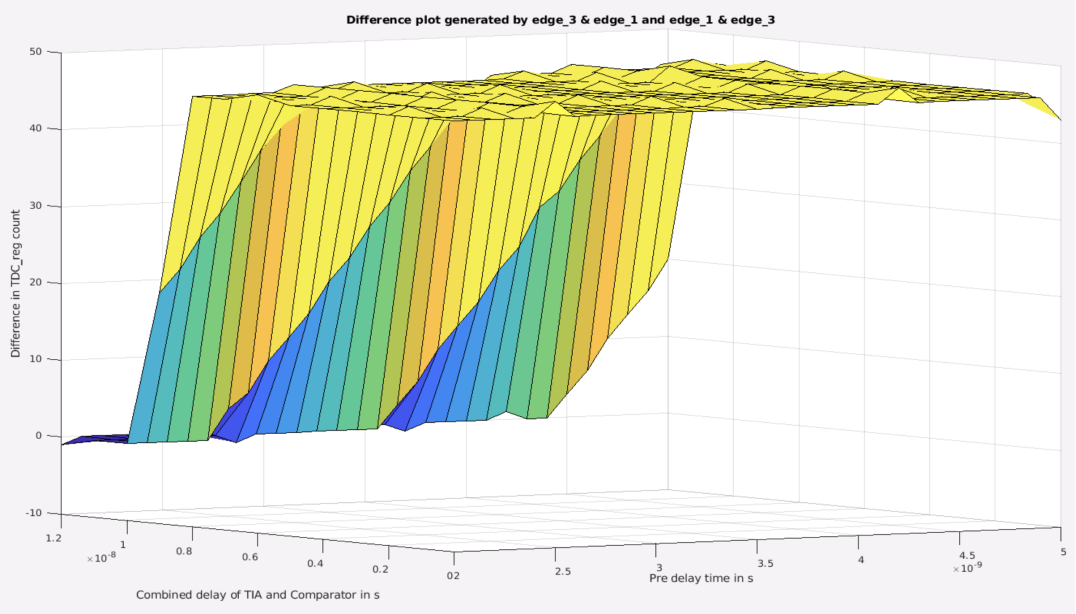

FIGURE 3.32: Difference Plot- 2 for *edge_1* & *edge_3* and *edge_3* & *edge_5* for Different TDC Parameters

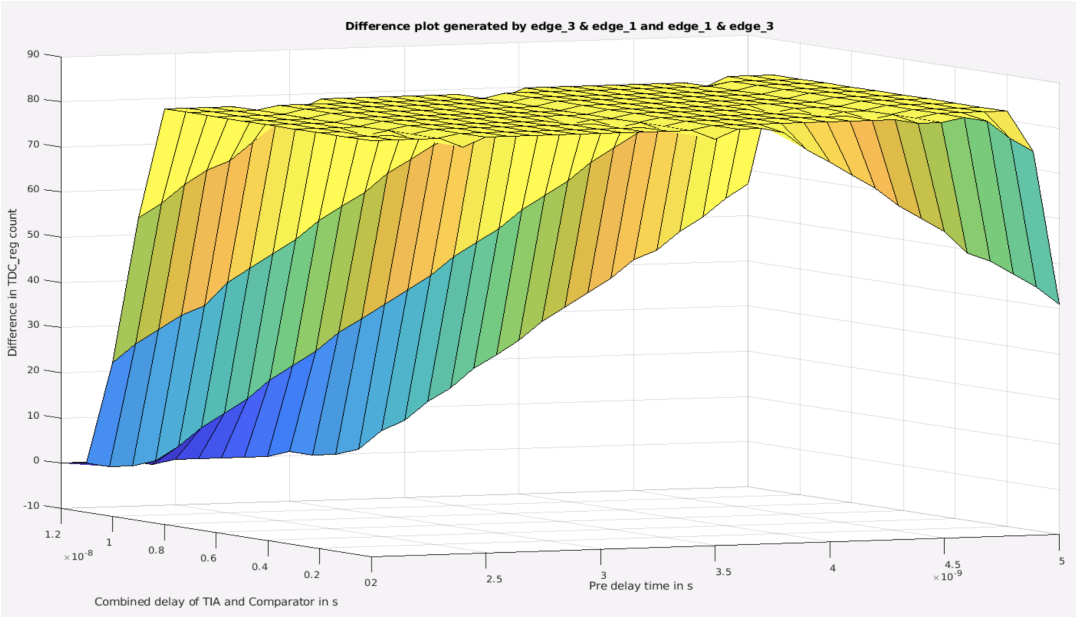

FIGURE 3.33: Difference Plot- 3 for *edge_1* & *edge_3* and *edge_3* & *edge_5* for Different TDC Parameters

It can be observed that in figure 3.31, 3.32 and 3.33, the flat regions start from 0ns for both the axes and show different levels of differences in the measurement results. So, it can be safely assumed that the reference edge sets *edge_1* & *edge_3* and *edge_3* & *edge_5* can be considered for the new calibration approach where the startsop pulses generated by *edge_1* & *edge_3* and *edge_3* & *edge_5* correspond to non-wrapping and wrapping measurements, respectively.

Note: - As described earlier, *edge_5* and *edge_1* are more or less the same reference edges as *edge_6* and *edge_2*. To avoid confusion while plotting the results, the names *edge_1* and *edge_2* are used.

Chapter 4

Verification and Implementation

4.1 Verification

4.1.1 Verification Environment for the Digital Part

After the digital part and the models of other components are ready, they are interconnected and put together in the testbench environment for simulation to ensure that the digital part works correctly and behaves as expected. The simulation is carried out by considering all the factors and parameters that could affect the performance and functionalities of the system as a whole. All the corner cases are taken into consideration during the simulation as if the system exists in actual working environment where the operating conditions can change at anytime. Figure 4.1 shows the testbench environment to verify the performance and functionalities of the digital part.

FIGURE 4.1: Simulation Environment for the Digital Part

From figure 4.1, it can be seen that the two multiplexers START_MUX and STOP_MUX are used in the simulation environment. These multiplexers are used to differentiate between the measurements done in the calibration state and those done in the compensation state as described in section of chapter 3. After startup, when the system recovers from the power on reset, reference edges are generated in the edge generator module for the calibration process. Depending on wrapping and non-wrapping measurements, different sets of reference edges are selected by the reference selector module. The control input *ref_sel* to the START_MUX and STOP_MUX

is connected to the output signal from the main FSM which signifies whether the system is in calibration or compensation process. When the system is in calibration, the START_MUX and STOP_MUX select *ref_shutter* and *ref_laser* as *start* and *stop* signals, respectively but when in compensation, the signals *shutter_detect* and *laser_after_driver* are considered.

After the calibration process is complete, the signals *calib_done* and *calib_running* are assigned to logic HIGH and logic LOW states by the system, respectively. Then, the system goes into the compensation process after mask bits are added to the 900-bit digital code. During the compensation process, the *start* and *stop* signals for the startstop generator are generated by the shutter and laser pulses. After the compensation is done, the system reaches stability and the digital part goes to sleep mode. The simulation result justifying this description is shown in figure 4.2. It is evident that the **Cursor 2** at 20425ns shows the completion of the calibration process. The **Cursor 1** at 97225ns shows the *comp_stable* signal going to a logic HIGH state, indicating the completion of the compensation process.

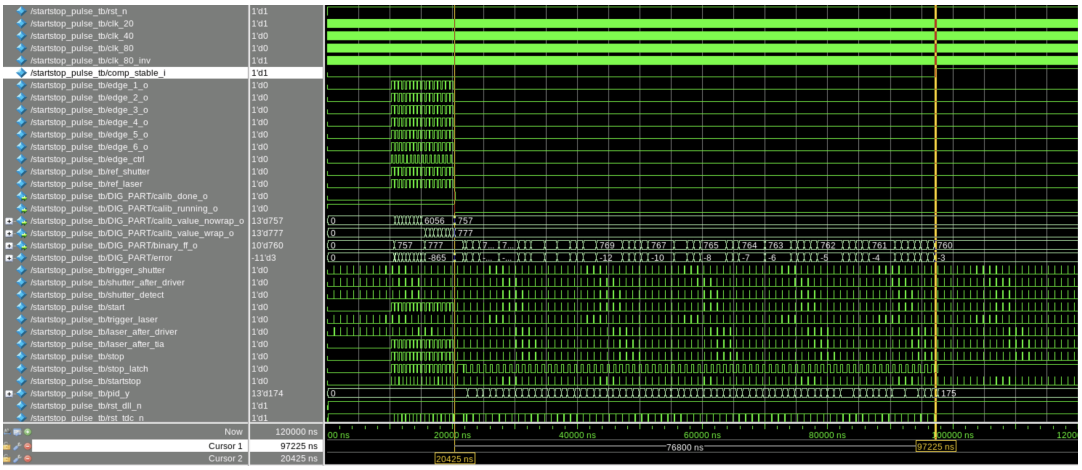

FIGURE 4.2: Simulation Results of the Digital Part

Since in an actual environment, the operating conditions might change, it is important to observe the quality of compensation for all corner cases by varying different parameters like shutter and laser driver delays, DelayClkPeriodError of TDC etc. To observe the quality of compensation, the system is forced to leave the sleep mode and the system reset is asserted from the testbench, thereby starting it again from the initial state of the main FSM. Laser and shutter pulses are applied from the testbench like the camera would generate them in actual operation. To control the shutter and laser driver delays by providing different delay values to them, *delay_shutter* and *delay_laser* signals are supplied as inputs to those models. After that, the delay between the laser emitting light and shutter opening is measured from the testbench but not using the TDC. For different delay values of shutter and laser drivers, the system is observed to be minimizing the error and bouncing back to the set point value. The same process is repeated for different values of the DelayClkPeriodError parameter of the TDC. The system is observed to be attaining stability for a wide range of positive values of DelayClkPeriodError and the variation from the set point value is within 100ps. But for the negative values beyond 250ps, it is observed that the system attains stability and the variation from the set point value is between

200-300ps. It is stated previously in chapter 3 that in order to achieve a depth resolution in the centimeter range, T_d needs to be measured with an accuracy below approximately 100ps for each pixel independently. This is possible if the value of the DelayClkPeriodError parameter of the TDC model is greater than -250ps. The simulation result justifying the quality of the compensation process is shown in figure 4.3. It can be seen that the system attains stability for different delay values of the shutter and laser drivers.

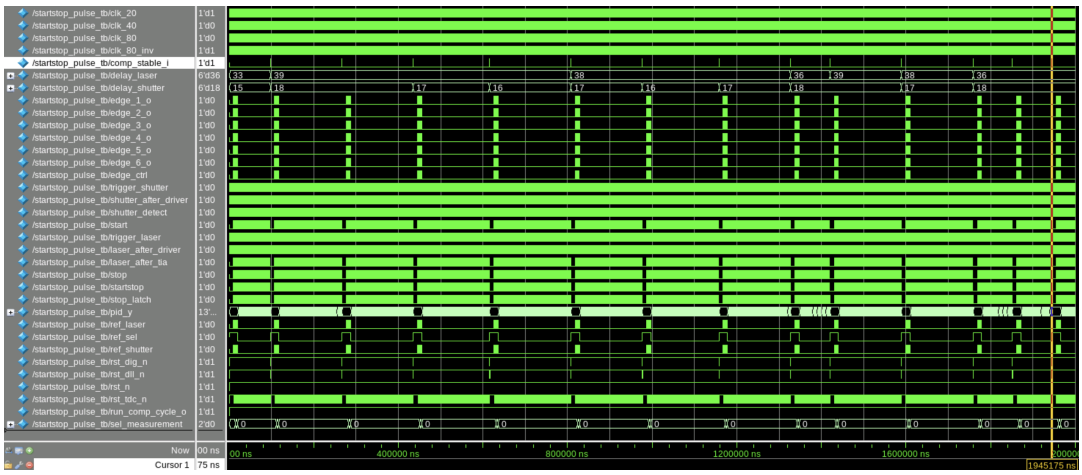

FIGURE 4.3: Simulation Showing the Quality of Compensation

4.1.2 Verification Environment for the Testchip

Figure 4.4 shows the top level simulation of the testchip. The testchip contains the models of all the components along with the digital part and the I2C control unit as shown in figure 4.5. The testchip is connected to the STIMULUS module which contains the self-checking testbench to verify that the entire system functions correctly over the I2C bus lines. In this setup, the testchip is the slave device. The STIMULUS module imitates the master device by generating the I2C clock signal for the testchip slave device. The testchip also receives the global asynchronous reset signal from the STIMULUS model for simulation purposes which in real life environment would be supplied from outside as system reset.

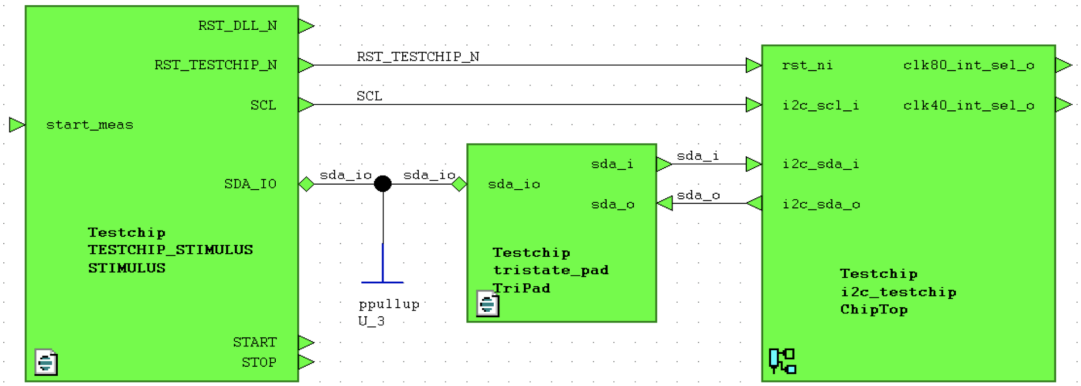

FIGURE 4.4: Top-Level Simulation of the Testchip

FIGURE 4.5: Components inside the Testchip

As described in subsection 2.6.1 of chapter 2, I2C clock and data lines are connected to the supply voltage through a pull-up resistor. To imitate that behavior, the *sda_io* line is connected to the pullup primitive. To mimic the I2C SDA line and ensure data transmission between the testchip and the STIMULUS module, the *sda_io* is a bi-directional signal here. For successful data transfer between the testchip and the STIMULUS module for verification purposes, a tristate pad model is added in between them, the code of which is the following:

```
module tristate_pad (
    input wire sda_o,
    output wire sda_i,
    inout tri1 sda_io
    );

    assign sda_io = (sda_o == 0) ?  1'b0 : 1'bz;

    assign sda_i = sda_io;

endmodule
```

The **Digital_Top** inside the testchip contains the digital part along with the I2C control unit as shown in figure 4.6.

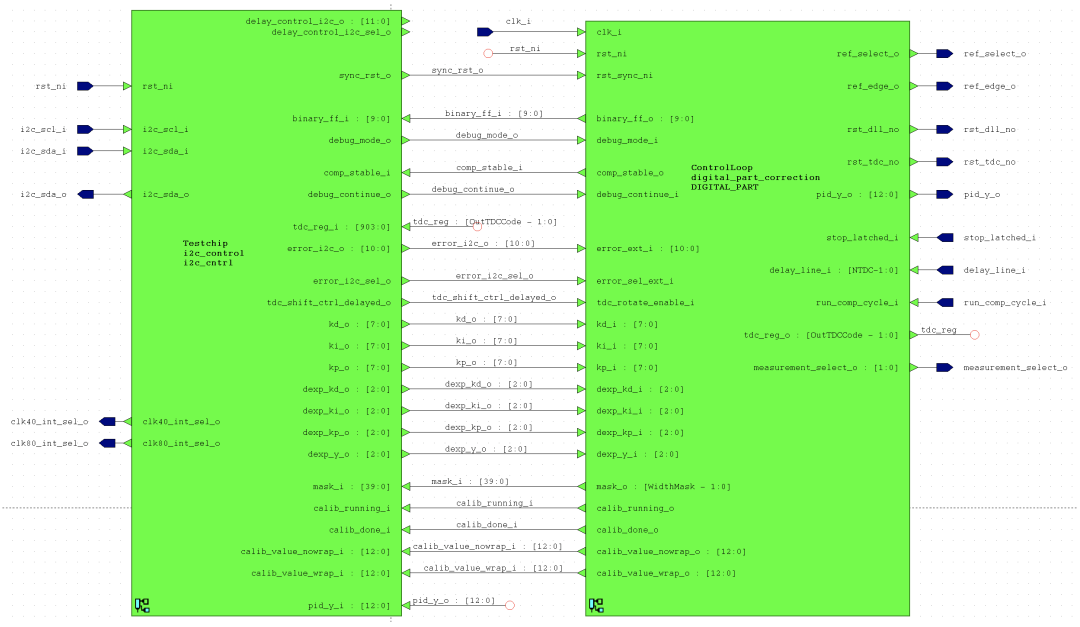

FIGURE 4.6: Digital Top level

4.1.3 Self Checking Testbench

The STIMULUS module of the top level testbench environment of the testchip contains the self-checking testbench which corresponds to the realistic test cycle for the final verification of the testchip. It simulates a complete test cycle of the actual chip without the need for running manual checks on different aspects of the chip. There are different tasks inside this self-checking testbench which ensure automated checks for different aspects of the chip and generate error messages in case of failed operations. These error messages can be viewed in the simulator log or can be written into an output file. The operations carried out via these tasks happen over the I2C bus lines. The steps involved in the implementation of this self-checking testbench are in the following order:

- Resetting the chip.
- Asserting synchronous reset.
- Setting necessary PID parameters.
- Enabling debug mode.
- Deasserting synchronous reset.
- Stepping through the calibration process using the signals dedicated for debug mode of operation.
  - Reading results of each calibration measurment.
  - Reading out the TDC register and the binary result and comparing them.
  - Comparing the output of the TDC sampling result with the pre-bubble and post-bubble correction values and checking for bubble errors and their corrections.
- Reading the calibration results, mask bits and calculating the combined delay of TIA and comparator after the calibration process is complete.

- Stepping through the compensation process by using the signals dedicated for debug mode of operation.
  - Reading results of each measurment.
  - Reading out of the TDC register and the binary result and comparing them.
  - Checking the visibility of double sampling effect.
  - Checking whether the double sampling and bubble error correction work properly or not.
  - Calculating the number of cycles to attain stability.
- Calculating the output of the PID after stability is reached.
- Disbaling debug mode and letting the system run freely.

The tasks which facilitates smooth verification and code readability of the self-checking testbench are described briefly here.

- **task write_bit**: This task is used by the master STIMULUS module to write the individual bits of the slave testchip device address, register address and to send NACK signal in case of completion of an I2C read transaction.
- **task read_bit**: This task is used by the master to read the ACK signal coming from the slave and also to read the individual bits of the content of a register in case of an I2C read transaction.
- **task set_regaddr**: This task is used to set the address of the register to be read. It uses **write_bit** and **read_bit** tasks. The input argument to this task is the address of the respective register.
- **task write_reg**: This task is used to write to a particular register. It also uses **write_bit** and **read_bit** tasks. The input arguments to this task are the address of the register to be written to and the data value which the user wants to write.
- **task read_reg_value**: This task is used to read the contents of the register, the address of which is set by the **task set_regaddr** task. The output argument of this task is the value of the respective register.
- **task set_PID_params**: This task is used to set all the parameters necessary for the PID operation. All these PID parameters are set by writing to the dedicated registers for PID operation.
- **task measurement_result**: This task is used to read the value of the measurement result register.
- **task reference_result**: This task is used to read the value of the reference result register.
- **task mask_readout**: This task is used to read the mask bits for double sampling/blind spot compensation.
- **task read_calib_info**: This task is used to read the status of the calibration process by reading the flags which represent whether calibration is done or still running.

- **task read_PID_output**: This task is used to read the PID output value.
- **task read_stability_info**: This task is used to read the stability flag which indicates whether the system has attained stability or not.
- **task change_debug_continue**: This task is used to toggle the debug signal responsible for the continuation of the system in debug mode.
- **task double_sampling_check**: This task is used to monitor if there is any double sampling in the thermometer code.
- **task read_tdc_reg_value**: This task is used to read the entire conents of the TDC sampling register output.
- **task tdc_binary_val_count**: This task is used to evaluate the binary representation of the thermometer code.
- **task calibration_state_debug**: With the help of debug mode, this task is used to step through the calibration process and measure all the aspects of each measurement by using all of the above mentioned tasks.
- **task result_after_calibration**: This task is used to read the measurement result, reference result and the mask bits after the calibration process is finished.
- **task compensation_state_debug**: With the help of debug mode, this task is used to step through the compensation process and measure all the aspects of each measurement by using all of the above mentioned tasks.

The simulaion result for the testchip in the normal mode of operation is shown in figure 4.7. The system is observed to be attaining stability at 386525ns. For the testchip operation in normal mode, the system reset and the synchronous reset are asserted first. Then, the necessary PID parameters are set. Then, the value 8'h00 is written to the **reg_1D** register and the synchronous reset is deasserted. The portion of the code of the self-checking testbench after the deassertion of the synchronous reset is commented out for the system operation in normal mode.

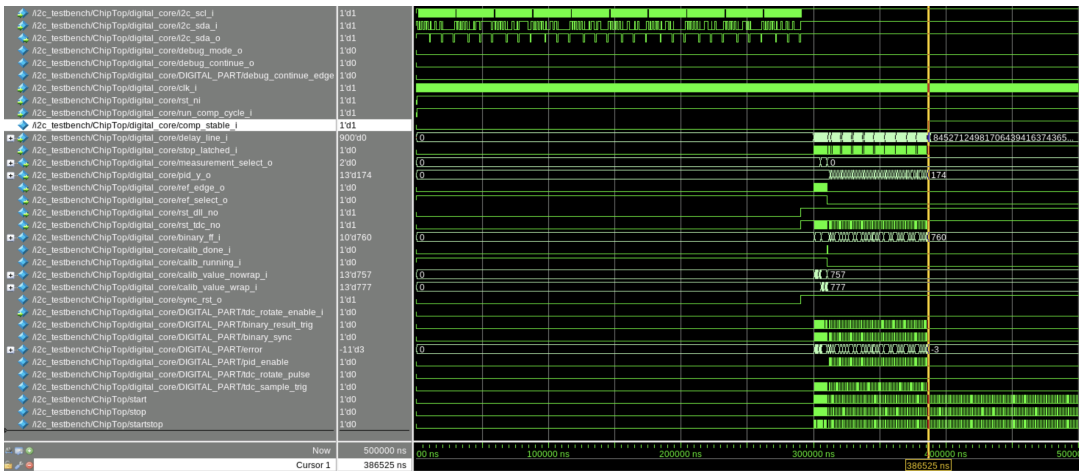

FIGURE 4.7: Simulation of the Testchip in Normal Mode of Operation

For the system operation in debug mode, the debug_mode and debug_continue bit-fields of the **reg_1D** register are asserted and the code segment which was commented out before is uncommented. The simulation result for the testchip operation in debug mode is shown in figure 4.8.

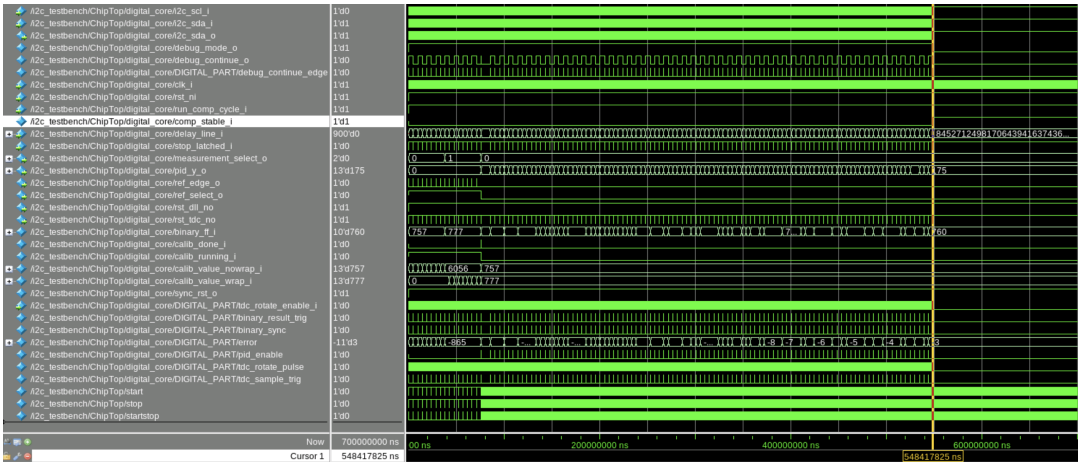

FIGURE 4.8: Simulation of the Testchip in Debug Mode of Operation

The transcript window of the simulator containing the debug outputs of the system in debug mode are presented here. Figure 4.9 shows the start of the calibration process and the debug outputs associated with it. These debug outputs are generated as the results of the tasks used for the debugging of the system which are described previously. After eight non-wrapping and eight wrapping measurements, the calibration process gets completed and the compensation process starts which can be seen in figure 4.11. The compensation loop runs until the T_d value gets settled around the set point value of 12.5ns. After the stability flag is asserted by the stability analysis module, the compensation process gets completed and the number of compensation cycles required to achieve stability and the final output of the PID controller are displayed as debug outputs which can be seen in figure 4.12.

```
# Calibration state debugging - Loop           1....
#        Measured binary value is 757
#        tdc register read out is successful
#        Count value of TDC read-out register is 757
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop           2....
#        Measured binary value is 757
#        tdc register read out is successful
#        Count value of TDC read-out register is 757
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop           3....
#        Measured binary value is 757
#        tdc register read out is successful
#        Count value of TDC read-out register is 757
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop           4....
#        Measured binary value is 757
#        tdc register read out is successful
#        Count value of TDC read-out register is 757
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop           5....
#        Measured binary value is 757
#        tdc register read out is successful
#        Count value of TDC read-out register is 757
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop           6....
#        Measured binary value is 757
#        tdc register read out is successful
#        Count value of TDC read-out register is 757
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop           7....
#        Measured binary value is 757
#        tdc register read out is successful
#        Count value of TDC read-out register is 757
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
```

FIGURE 4.9: Calibration Debug Outputs- 1

```
# Calibration state debugging - Loop           8....
#        Measured binary value is 757
#        tdc register read out is successful
#        Count value of TDC read-out register is 757
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop           9....
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop          10....
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop          11....
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop          12....
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop          13....
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop          14....
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
```

FIGURE 4.10: Calibration Debug Outputs- 2

```
# Calibration state debugging - Loop          15....
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
# Calibration state debugging - Loop          16....
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        Reading out the TDC register matched the binary result.
#        No bubble errors.
#
#
#        Calibration is complete
#
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 793
#        The reference value is 757
#        Double sampling correction mask value is 1099510579200
#        Double sampling correction applied in TDC register
#        No bubble errors.
# TIA+Comparator delay is               9.00ns
# Compensation state debugging - Loop           1....
#        Measured binary value is 777
#        tdc register read out is successful
#        Count value of TDC read-out register is 793
#        TDC register value read-out and binary result don't match.
#        Double sampling is visible in TDC register
#        No bubble errors.
#
#
# Compensation state debugging - Loop           2....
#        Measured binary value is 776
#        tdc register read out is successful
#        Count value of TDC read-out register is 792
#        TDC register value read-out and binary result don't match.
#        Double sampling is visible in TDC register
#        No bubble errors.
#
#
# Compensation state debugging - Loop           3....
#        Measured binary value is 776
#        tdc register read out is successful
#        Count value of TDC read-out register is 792
#        TDC register value read-out and binary result don't match.
#        Double sampling is visible in TDC register
#        No bubble errors.
```

FIGURE 4.11: Calibration & Compensation Debug Outputs

```
# Compensation state debugging - Loop          94....
#        Measured binary value is 761
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        TDC register value read-out and binary result don't match.
#        Double sampling is visible in TDC register
#        No bubble errors.
#
#
# Compensation state debugging - Loop          95....
#        Measured binary value is 760
#        tdc register read out is successful
#        Count value of TDC read-out register is 776
#        TDC register value read-out and binary result don't match.
#        Double sampling is visible in TDC register
#        No bubble errors.
#
#
# Compensation state debugging - Loop          96....
#        Measured binary value is 760
#        tdc register read out is successful
#        Count value of TDC read-out register is 776
#        TDC register value read-out and binary result don't match.
#        Double sampling is visible in TDC register
#        No bubble errors.
#
#
# Compensation state debugging - Loop          97....
#        Measured binary value is 761
#        tdc register read out is successful
#        Count value of TDC read-out register is 777
#        TDC register value read-out and binary result don't match.
#        Double sampling is visible in TDC register
#        No bubble errors.
#
#
# Compensation state debugging - Loop          98....
#        Measured binary value is 760
#        tdc register read out is successful
#        Count value of TDC read-out register is 776
#        TDC register value read-out and binary result don't match.
#        Double sampling is visible in TDC register
#        No bubble errors.
#
#
#        Number of compensation cycles took to achieve stability = 98
#        PID value is 175
```

FIGURE 4.12: Compensation Debug Outputs

4.2 Implementation

After the design is successfully verified, it has to undergo a series of steps before the production of the final chip. After the design successfully passes through all the steps, the GDSII file is generated and released to the fabs for manufacturing of the chip which is known as *tapeout*. The entire design flow and the individual steps are

described in detail as part of the previous work[16]. In this work presented here, the main focus is on the Engineering Change Order (ECO) and re-importing the modified netlist into IC Compiler to remove potential timing violations.

Engineering Change Order is a method to patch or modify the gate level, post synthesized version of a design. It is used to accommodate last minute design revisions and changes. When some changes occur during the later stages of the design implementation, may it be an RTL bug fix or addition of some extra portions of RTL code, it might cause problems during the signoff phase. To circumvent the idea of re-implementing the entire design, ECO comes in handy which is cost effective and saves time and effort.

During the final part of the design flow, the design is observed to have setup and hold time violations in sign-off STA. To remove those violations, a number of steps are carried out which also involve the inclusion of the script file for fixing timing violations and inclusion of additional Synopsys commands[17] in the implementation script to automate the ECO flow. A number of changes are also adopted to the usual design flow. Addition of the new script for fixing timing violations and the changes made to the existing implementation script are described later. The steps followed and the changes made in the flow in mitigating the timing violations are jotted down below.

- In the Synopsys Design Constraints (SDC) file, the clock uncertainties corresponding to setup and hold times are removed and included in a separate TCL[18] script intended for setup and hold violations fixing which is called as *signoff_pt.tcl*.
- Then, the synthesis and implementation flow are run followed by parasitic extraction by using the respective commands and scripts as described in the previous work[16].
- After that, PrimeTime shell is opened for running sign-off STA. For setup and hold time analyses, *primetime_max.tcl* and *primetime_min.tcl* scripts are run and timing violations are observed.
- To fix the timing violations, *signoff_pt.tcl* is run after it is sourced into PrimeTime shell. This script generates an output file *eco_pt.tcl* containing all the changes to be made to the existing netlist for fixing the violations. Usually, these types of files contain addition of buffers, inverter pairs, cell sizing information etc. based on the Synopsys commands used in the script.
- PrimeTime is closed and the implemented design is opened in IC Compiler again if it has been closed previously. To incorporate the changes generated by PrimeTime into the existing netlist, the modified and newly added cells contained in *eco_pt.tcl* need to be placed and routed in IC Compiler.
- To do so, *eco_pt.tcl* file is imported into IC Compiler by using the command **eco_netlist -by_tcl_file** followed by the name of the file i.e. *eco_pt.tcl*.
- After that, the ECO flow needs to be run on the design in IC Compiler to merge the contents of *eco_pt.tcl* into the existing netlist. The entire ECO flow is put inside a procedure named **flow_step_eco_signoff** and the procedure is added in the implementation script *icc_flow.tcl*. To run the ECO flow, this procedure needs to be executed in IC Compiler.

- After the ECO flow is complete, parasitic extraction procedure is carried out on the changed netlist by using the StarRC command. After parasitic extraction is complete, the tool writes relevant information into SPEF file formats which are used by PrimeTime for timing analysis.
- The timing scripts *primetime_max.tcl* and *primetime_min.tcl* are run in PrimeTime shell and it is found that all the timing violations are removed. PrimeTime writes all the timing information into SDF files.

4.2.1 Addition of New Script for Fixing Timing Violations

The newly added script *signoff_pt.tcl* contains Synopsys commands for fixing setup and hold violations present in the design. The commands used in the script are described here.

- **set_clock_uncertainty -hold 0.5 [get_clocks "clk"] :** This command is used to specify the uncertainty or skew for hold time of the specified clock network.
- **set_clock_uncertainty -setup 1 [get_clocks "clk"] :** This command is used to specify the skew for setup time of the clock network driven by *clk*.
- **fix_eco_timing -type setup -cell_type {combinational} -pba_mode exhaustive -methods {size_cell_side_load} -verbose :** This command is used for fixing the timing violations. It uses some switches followed by arguments, the functions of which are described below.
  - **-type :** Specifies the type of timing violations the tool is trying to fix, which in this case are setup violations, justified by the argument "setup".
  - **-cell_type :** Specifies the type of cells to be modified for timing fixing. The default argument is "combinational" where fixing is performed by sizing or inserting combinational logic cells in the data path.
  - **-pba_mode :** Specifies the mode of the timing analysis. The argument "exhaustive" performs an exhaustive path based analysis to determine worst case paths in the design. This is the most computation intensive and accurate mode.
  - **-methods :** Specifies one or more fixing methods. "size_cell_side_load" argument replaces cells in the timing path with logically identical cells having a different drive strength. It also replaces cells in the fanout of drivers in the path with logically identical cells to reduce parasitic load capacitance along the path. This argument can only be used for setup fixing but not hold fixing.
  - **-verbose :** Shows additional information during the process of fixing. It also generates the report which has the violations that can't be fixed if the **eco_report_unfixed_reason_max_endpoints** variable is set to a positive integer in PrimeTime shell.
- **fix_eco_timing -type setup -cell_type {sequential} -pba_mode exhaustive -methods {size_cell} -verbose :** This command is the same as above with same switches having different arguments. The arguments are described here.
  - "sequential" : Timing fixing is performed by sizing sequential cells to fix the violations at input or output pins of the cells. Only the "size_cell" fixing method is supported for this type of fixing.

  - "size_cell" : It only replaces cells in the timing path with logically identical cells having a different drive strength.

- **fix_eco_timing -type hold -buffer_list {BF BF2 BF3 BF4 BF5 BF8 DEL10 SCHMTT25} -pba_mode exhaustive -methods {size_cell insert_buffer} -verbose :** This timing fixing is carried out for fixing hold violations as specified by the argument "hold" of the switch **-type**.
  - **-buffer_list :** Specifies the list of library buffer cells. It is used along with the "insert_buffer" method. There is no default list of buffers and this switch must be specified along with the list of buffers from the library.
  - **-methods {size_cell insert_buffer} :** This command can insert a buffer from the buffer list specified above and then size it to a different buffer, even if the sized library cell is not explicitly included in the **-buffer_list** option.
- **fix_eco_timing -type setup -methods remove_buffer :** Delay cells and buffers placed by the hold fixing command might affect setup paths. To remove the redundant buffers that give rise to setup violations without worsening or introducing hold violations, "remove_buffer" method is used. In setup fixing, it can't be used with other methods and must be used by itself.
- **write_changes -format icctcl -output eco_pt.tcl :** This command writes out the modifications performed in netlist. The **-format** switch specifies the output file format and the argument "icctcl" specifies that the output file format is a tcl script for IC Compiler or IC Coompiler II. The **-output** switch writes the change list to the specified file, which here is *eco_pt.tcl*.
- **update_timing :** This command updates timing information on the current design. This command is used multiple times in the script to ensure that the timing information gets updated after every stage of timing fixing.

4.2.2 Addition of Commands in Implementation Script for ECO Flow

This subsection describes different commands that are used as part of the ECO flow in IC Compiler.

- **remove_stdcell_filler -stdcell :** This command deletes standard cells, end cap cells, tap filler cells and pad cells. To incorporate the changes into existing design netlist, these cells need to be removed first, so that there is room for accomodating the changes.
  - **-stdcell :** Used to remove the standard cell filler cells.
- **derive_pg_connection :** This command connects power, ground and tie-off pins to the specified power and ground nets.
- **place_eco_cells -eco_changed_cells :** This command is used to perform coarse placement and legalization on ECO cells in the design.
  - **-eco_changed_cells :** Places all the cells with **eco_change_status** attribute that has values **create_cell**, **insert_buffer**, **size_cell** etc.
- **legalize_placement -effort high :** This command executes detailed placement on the design.

  - **-effort :** Specifies the effort level for detailed placement, which in this case is high, justified by the argument.

- **insert_stdcell_filler -cell_without_metal "DECAP16 FSTDS8 FSTDS4 FSTDS2 FSTDS" -connect_to_power {VDD} -connect_to_ground {GND} :** This command is used to fill empty spaces in standard cell rows with filler cells.

  - **-cell_without_metal :** Specifies the list of filler cells to be used. The filler cells which don't contain metal, also known as nonmetal filler cells, must be used. The filler cells from Elmos library are specified inside the quotes.
  - **-connect_to_power :** Specifies the name of the existing power net to which the filler cells should be connected. Here, the name of the power net is VDD.
  - **-connect_to_ground :** Specifies the name of the existing ground net to which the filler cells should be connected. Here, the name of the ground net is GND.

- **route_zrt_eco -reroute modified_nets_first_then_others -max_detail_route_iterations 5 :** This command is used to perform ECO routing on the design.

  - **-reroute :** Controls which nets are rerouted. The argument used after this switch describes that the router first freezes all the fully connected nets and tries to finish the routing by modifying only the nets with open ECO changes. If still some violations remain, the router tries to reroute the fully connected nets to resolve the violations.
  - **-max_detail_route_iterations :** Specifies the maximum number of detailed routing iterations, which in this case is 5.

- **save_mw_cel :** This command saves the specified design in Milkyway format.

- **write_verilog post_icc.v :** Writes a hierarchical Verilog file for the current design, the name of which is *post_icc.v* here.

- **write_verilog -pg -force_no_output_references "WELLTAPS FSTDS16 FSTDS8 FSTDS4 FSTDS2 FSTDS" post_icc_pg.v :** This is the same command as above with different switches which are described below.

  - **-pg :** Writes power and ground nets and ports for all cell and module instances.
  - **-force_no_output_references :** Specifies the reference cell names for which the cell instances must not be written. These cell names are specified inside the quotes, separated by space characters.

- **write_verilog -no_physical_only_cells post_icc_no_phys.v :** The switch used here is **-no_physical_only_cells** which prevents writing of physical-only cell instances.

- **extract_rc -incremental :** This command executes 2.5D extraction for routes in the design.

  - **-incremental :** Performs incremental RC extraction based on the last extraction. It only works with a detailed routed design.

- **write_parasitics -no_name_mapping :** This command is used to write parasitics to a disk file for delay calculation tools.
  - **-no_name_mapping :** Specifies that the net name is used directly in the file. This option is valid only when the parasitics are written in SPEF format. Actual net names are used in the SPEF file when this switch is specified.

4.2.3 Results Before the ECO Flow

During the implementation stage, a number of steps like floorplanning, placement, routing, clock tree synthesis, optimizations, timing analysis etc. are performed on the design. After the implementation flow is complete, the layout of the design gets generated. The commands **report_timing -delay min** and **report_timing -delay max** are run in the IC Compiler shell to display the timing information for the min and max corner analysis, otherwise known as the hold and setup analysis, respectively. The timing reports of the min and max corner analysis are presented here.

```
****************************************
Report : timing
        -path full
        -delay min
        -max_paths 1
Design : Digital_Top
Scenario(s): minmax
Version: Q-2019.12-SP5-1
Date   : Sun Aug 27 17:59:37 2023
****************************************

 * Some/all delay information is back-annotated.
Wire Load Model Mode: Inactive.
        Scenario            : minmax
        Parasitic source    : LPE
        Parasitic mode      : RealRC
        Extraction mode     : MULTI_CORNER
        Extraction derating : -40/150

Information: Percent of Arnoldi-based delays = 32.16% on scenario
                                                          minmax

  Startpoint: i2c_cntrl/U_6/U_0/reg_1D/data_o_reg[6]
              (rising edge-triggered flip-flop clocked by i2c_clk')
  Endpoint: DIGITAL_PART/mw_U_6reg_cval_reg
            (rising edge-triggered flip-flop clocked by clk)
  Scenario: minmax
  Path Group: clk
  Path Type: min

  Point                                  Incr       Path      Voltage
  ---------------------------------------------------------------------
  clock i2c_clk' (rise edge)           500.00     500.00
  clock network delay (propagated)       1.25     501.25
```

```
i2c_cntrl/U_6/U_0/reg_1D/
   data_o_reg[6]/C (DFSRLQ)          0.00     501.25 r    3.60
i2c_cntrl/U_6/U_0/reg_1D/
   data_o_reg[6]/Q (DFSRLQ)          2.39     503.63 f    3.60
DIGITAL_PART/mw_U_6reg_cval_reg/
                      D (DFSRLQ)     0.00 &   503.63 f    3.60
data arrival time                             503.63

clock clk (rise edge)               500.00     500.00
clock network delay (propagated)      2.27     502.27
DIGITAL_PART/mw_U_6reg_cval_reg/
                      C (DFSRLQ)      0.00     502.27 r
library hold time                    -0.55     501.73
data required time                             501.73
-------------------------------------------------------------------
data required time                             501.73
data arrival time                             -503.63
-------------------------------------------------------------------
slack (MET)                                      1.90


Startpoint: i2c_cntrl/U_6/Protocolunit/counter/counter_reg[0]
            (rising edge-triggered flip-flop clocked by i2c_clk)
Endpoint: i2c_cntrl/U_6/Protocolunit/counter/counter_reg[0]
          (rising edge-triggered flip-flop clocked by i2c_clk)
Scenario: minmax
Path Group: i2c_clk
Path Type: min

Point                                    Incr       Path      Voltage
-------------------------------------------------------------------
clock i2c_clk (rise edge)                0.00       0.00
clock network delay (propagated)         1.25       1.25
i2c_cntrl/U_6/Protocolunit/counter/
          counter_reg[0]/C (DFSHSL)
                                         0.00       1.25 r    3.60
i2c_cntrl/U_6/Protocolunit/counter/
          counter_reg[0]/Q (DFSHSL)
                                         1.16       2.41 r    3.60
U2153/O (NAND2XL)                        0.44 &     2.85 f    3.60
i2c_cntrl/U_6/Protocolunit/counter/
          counter_reg[0]/D (DFSHSL)
                                         0.00 &     2.85 f    3.60
data arrival time                                   2.85

clock i2c_clk (rise edge)                0.00       0.00
clock network delay (propagated)         1.25       1.25
i2c_cntrl/U_6/Protocolunit/counter/
          counter_reg[0]/C (DFSHSL)
                                         0.00       1.25 r
library hold time                       -0.42       0.83
```

```
  data required time                                  0.83
  ------------------------------------------------------------------
  data required time                                  0.83
  data arrival time                                  -2.85
  ------------------------------------------------------------------
  slack (MET)                                         2.02

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : Digital_Top
Scenario(s): minmax
Version: Q-2019.12-SP5-1
Date   : Sun Aug 27 17:55:31 2023
****************************************

 * Some/all delay information is back-annotated.
Wire Load Model Mode: Inactive.
        Scenario            : minmax
        Parasitic source    : LPE
        Parasitic mode      : RealRC
        Extraction mode     : MULTI_CORNER
        Extraction derating : -40/150

Information: Percent of Arnoldi-based delays = 32.16% on scenario
                                                        minmax

  Startpoint: DIGITAL_PART/reg_tdc/tdc_reg_o_reg[225]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: DIGITAL_PART/binary_ff_o_reg[8]
            (rising edge-triggered flip-flop clocked by clk)
  Scenario: minmax
  Path Group: clk
  Path Type: max

  Point                                  Incr       Path      Voltage
  ------------------------------------------------------------------
  clock clk (rise edge)                  0.00       0.00
  clock network delay (propagated)       2.76       2.76
  DIGITAL_PART/reg_tdc/
     tdc_reg_o_reg[225]/C (EDFMHRLQ)
                                         0.00       2.76 r    3.00
  DIGITAL_PART/reg_tdc/
     tdc_reg_o_reg[225]/Q (EDFMHRLQ)
                                         1.61       4.37 r    3.00
  U629/O (INV)                           1.14 &     5.51 f    3.00
  U631/O (NOR2XL)                        1.48 &     6.99 r    3.00
  U676/O (AND21OXL)                      2.00 &     8.99 f    3.00
  U1003/CO (FADD)                        1.51 &    10.50 f    3.00
```

```
U1000/S (FADD)                          2.40 &    12.90 r    3.00
U1218/S (FADD)                          2.51 &    15.41 f    3.00
U1475/S (FADD)                          2.44 &    17.85 r    3.00
U1570/S (FADD)                          2.70 &    20.55 f    3.00
U1628/S (FADD)                          2.53 &    23.08 r    3.00
U1673/CO (FADD)                         2.32 &    25.40 r    3.00
U1671/CO (FADD)                         0.97 &    26.37 r    3.00
U1662/S (FADD)                          2.44 &    28.80 f    3.00
U1670/S (FADD)                          2.41 &    31.22 r    3.00
U1688/CO (FADD)                         2.10 &    33.31 r    3.00
U1686/S (FADD)                          2.27 &    35.59 f    3.00
U2939/CO (FADD)                         1.90 &    37.49 f    3.00
U2938/CO (FADD)                         0.98 &    38.47 f    3.00
U2937/CO (FADD)                         1.01 &    39.48 f    3.00
U2936/CO (FADD)                         1.01 &    40.49 f    3.00
U2935/S (FADD)                          2.43 &    42.92 r    3.00
DIGITAL_PART/binary_ff_o_reg[8]/
                   D0 (EDFMHRLQ)        0.03 &    42.95 r    3.00
data arrival time                                 42.95

clock clk (rise edge)                  50.00      50.00
clock network delay (propagated)        2.65      52.65
DIGITAL_PART/binary_ff_o_reg[8]/
                    C (EDFMHRLQ)        0.00      52.65 r
library setup time                     -1.57      51.08
data required time                                51.08
-----------------------------------------------------------------
data required time                                51.08
data arrival time                                -42.95
-----------------------------------------------------------------
slack (MET)                                       8.12


Startpoint: DIGITAL_PART/MASK/mask_reg[37]
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: i2c_cntrl/U_6/Protocolunit/shift_reg_out/shift_reg_reg[5]
          (rising edge-triggered flip-flop clocked by i2c_clk')
Scenario: minmax
Path Group: i2c_clk
Path Type: max

Point                                    Incr       Path      Voltage
-----------------------------------------------------------------
clock clk (rise edge)                  450.00     450.00
clock network delay (propagated)         2.71     452.71
DIGITAL_PART/MASK/mask_reg[37]/
                   C (EDFMHRLQ)          0.00     452.71 r    3.00
DIGITAL_PART/MASK/mask_reg[37]/
                   Q (EDFMHRLQ)          2.02     454.73 r    3.00
U6013/O (AND22OXL)                       1.34 &   456.07 f    3.00
U6014/O (INV)                            0.76 &   456.83 r    3.00
```

```
  U6015/O (AND21OXL)                      1.43 &   458.25 f    3.00
  U6017/O (OR211AXL)                      1.32 &   459.58 r    3.00
  U6023/O (AND211OXL)                     1.48 &   461.06 f    3.00
  U6024/O (NAND3XL)                       1.82 &   462.88 r    3.00
  U6025/O (AND211OXL)                     1.33 &   464.21 f    3.00
  U6026/O (OR2N2AXL)                      0.72 &   464.93 r    3.00
  i2c_cntrl/U_6/Protocolunit/
  shift_reg_out/shift_reg_reg[5]/
                   D0 (EDFMHRLQ)
                                          0.00 &   464.93 r    3.00
  data arrival time                                464.93

  clock i2c_clk' (rise edge)            500.00     500.00
  clock network delay (propagated)        1.70     501.70
  i2c_cntrl/U_6/Protocolunit/
  shift_reg_out/shift_reg_reg[5]/
                    C (EDFMHRLQ)
                                          0.00     501.70 r
  library setup time                     -1.38     500.32
  data required time                               500.32
  ---------------------------------------------------------------------
  data required time                               500.32
  data arrival time                               -464.93
  ---------------------------------------------------------------------
  slack (MET)                                       35.40
```

To display the information about the number of combinational and sequential cells used and the area occupied by them, the command **report_area** is run. The information regarding the number of cells and area is presented here.

```
Library (s) Used:
ELMOSOU35HD_ SS (File: /eda/kits/ELMOS/L035/pdk/athen/3.0.3/
                       L035_libraries/ELMOSOU35HD/db/elmos0u35hd_ss.db)
Number of ports:                  931
Number of nets:                  9242
Number of cells:                 7244
Number of combinational cells:   5688
Number of sequential cells:      1556
Number of macros/black boxes:       0
Number of buf/inv:               1998
Number of references:              66

Combinational area:       565064.816017
Buf/Inv area:              87619.969374
Noncombinational area:    465785.505920
Macro/Black Box area:          0.000000
Net Interconnect area:    undefined (No wire load specified)

Total cell area:         1030850.321938
Total area:               undefined
```

The layout of the design is shown in figure 4.13.

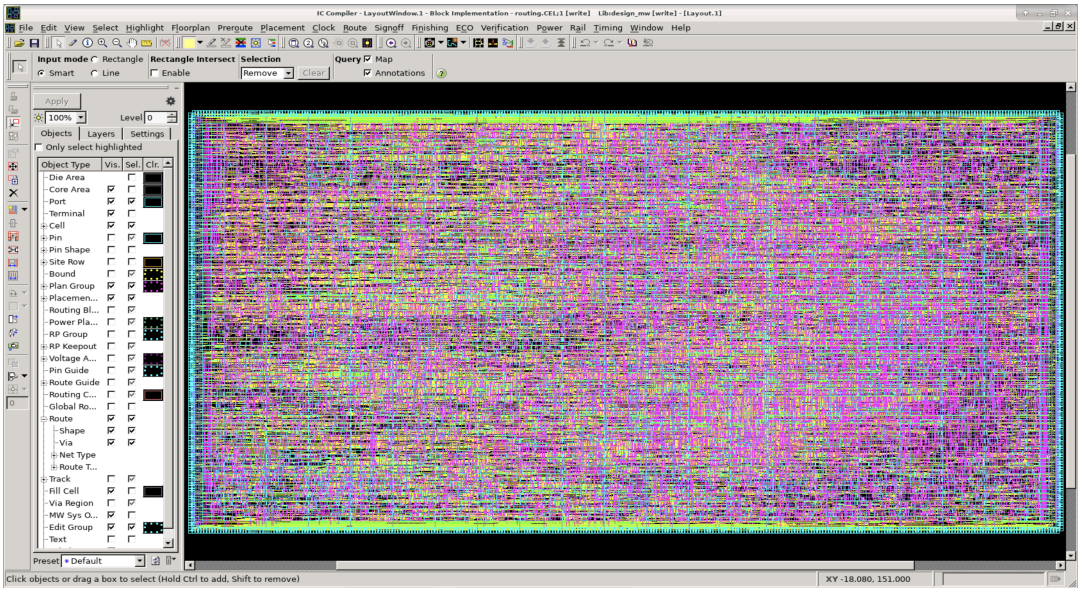

FIGURE 4.13: Layout Window of IC Compiler Before ECO

After the implementation flow is run and the layout is generated in IC Compiler, parasitic extraction is carried out on the design to create an accurate analog model of the circuit, so that detailed simulations can emulate actual digital and analog circuit responses. For calculating the delays, one should be aware of the resistances, capacitances, inductances etc. of the design network. The extraction tool used for this purpose is StarRC. After the parasitic extraction is carried out, signoff STA is carried out in PrimeTime. The Timing analysis for the min and max corners are shown in figure 4.14 and figure 4.15, respectively.

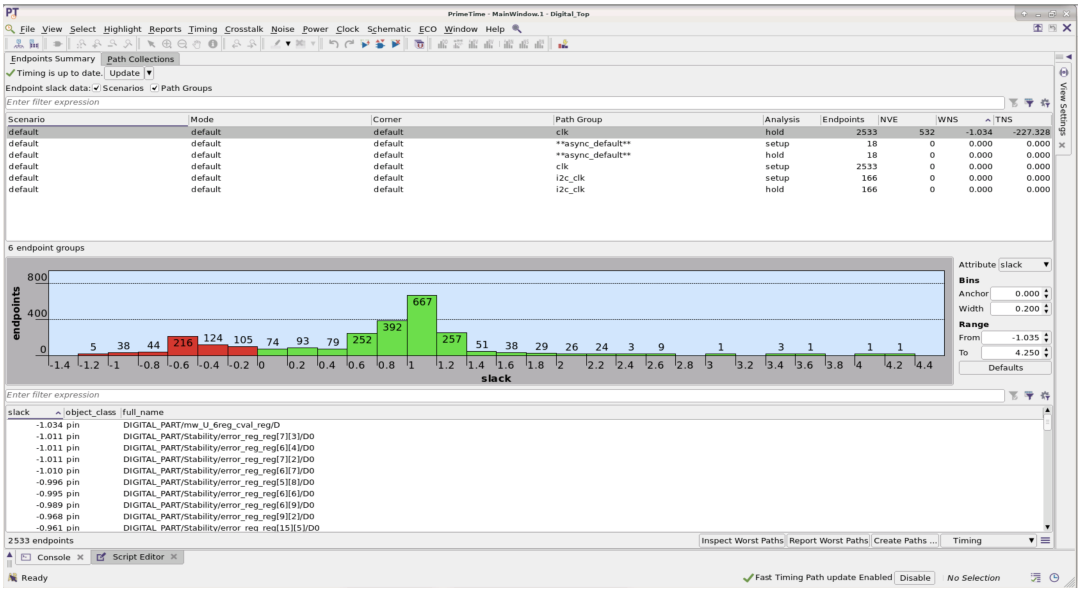

FIGURE 4.14: Timing Analysis of the Min Corner Before ECO

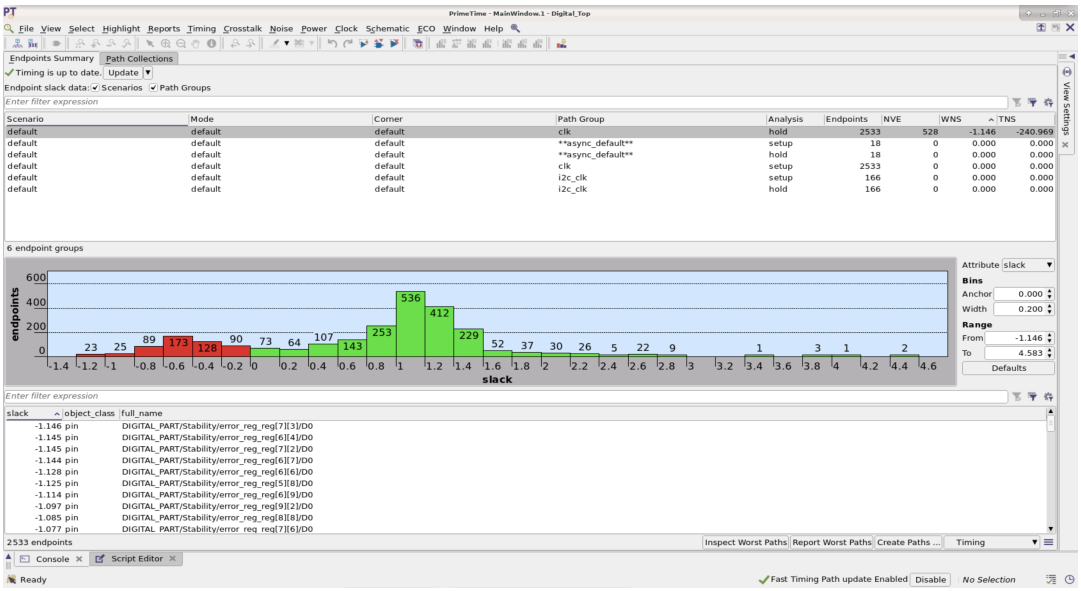

FIGURE 4.15: Timing Analysis of the Max Corner Before ECO

Since PrimeTime is an accurate signoff engine and gives the exact timing information, the command **report_timing -delay min** is run in the PrimeTime shell for the min corner analysis of the design after sourcing the *primetime_min.tcl* script. Similarly, the command **report_timing -delay max** is run in the shell for the max corner analysis after sourcing the *primetime_max.tcl* script. The results are presented here.

```
****************************************
Report : timing
-path_type full
-delay_type min
-max_paths 1
-sort_by slack
Design : Digital_Top
Version: Q-2019.12-SP1
Date   : Sun Aug 27 19:38:18 2023
****************************************


  Startpoint: i2c_cntrl/U_6/U_0/reg_1D/data_o_reg[6]
              (rising edge-triggered flip-flop clocked by i2c_clk')
  Endpoint: DIGITAL_PART/mw_U_6reg_cval_reg
              (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: min

  Point                                          Incr       Path
  -----------------------------------------------------------------------
  clock i2c_clk' (rise edge)                   500.00     500.00
  clock network delay (propagated)               0.48     500.48
  i2c_cntrl/U_6/U_0/reg_1D/data_o_reg[6]/
                                C (DFSRLQ)       0.00     500.48 r
  i2c_cntrl/U_6/U_0/reg_1D/data_o_reg[6]/
```

```
                              Q (DFSRLQ)       0.90 &   501.39 f
  DIGITAL_PART/mw_U_6reg_cval_reg/
                        D (DFSRLQ)             0.00 &   501.39 f
  data arrival time                                     501.39

  clock clk (rise edge)                       500.00     500.00
  clock network delay (propagated)              2.58     502.58
  clock reconvergence pessimism                 0.00     502.58
  DIGITAL_PART/mw_U_6reg_cval_reg/
                        C (DFSRLQ)            502.58 r
  library hold time                            -0.15     502.42
  data required time                                     502.42
  ---------------------------------------------------------------
  data required time                                     502.42
  data arrival time                                     -501.39
  ---------------------------------------------------------------
  slack (VIOLATED)                                        -1.03

****************************************
Report : timing
-path_type full
-delay_type max
-max_paths 1
-sort_by slack
Design : Digital_Top
Version: Q-2019.12-SP1
Date   : Sun Aug 27 19:34:55 2023
****************************************


  Startpoint: DIGITAL_PART/reg_tdc/tdc_reg_o_reg[225]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: DIGITAL_PART/binary_ff_o_reg[8]
              (rising edge-triggered flip-flop clocked by clk)
  Last common pin: INV5_G1B10I2/O
  Path Group: clk
  Path Type: max

  Point                                          Incr       Path
  ---------------------------------------------------------------
  clock clk (rise edge)                          0.00       0.00
  clock network delay (propagated)               2.79       2.79
  DIGITAL_PART/reg_tdc/
  tdc_reg_o_reg[225]/C (EDFMHRLQ)
                                                 0.00       2.79 r
  DIGITAL_PART/reg_tdc/
  tdc_reg_o_reg[225]/Q (EDFMHRLQ)
                                                 1.61 &     4.41 r
  U629/O (INV)                                   1.29 &     5.70 f
  U631/O (NOR2XL)                                1.62 &     7.32 r
  U676/O (AND21OXL)                              2.21 &     9.53 f
```

```
U1003/CO (FADD)                                  1.58 &    11.11 f
U1000/S (FADD)                                   2.42 &    13.53 r
U1218/S (FADD)                                   2.54 &    16.07 f
U1475/S (FADD)                                   2.47 &    18.54 r
U1570/S (FADD)                                   2.76 &    21.30 f
U1628/S (FADD)                                   2.58 &    23.88 r
U1673/CO (FADD)                                  2.36 &    26.23 r
U1671/CO (FADD)                                  0.98 &    27.22 r
U1662/S (FADD)                                   2.44 &    29.66 f
U1670/S (FADD)                                   2.43 &    32.09 r
U1688/CO (FADD)                                  2.12 &    34.21 r
U1686/S (FADD)                                   2.29 &    36.50 f
U2939/CO (FADD)                                  1.93 &    38.42 f
U2938/CO (FADD)                                  1.01 &    39.43 f
U2937/CO (FADD)                                  1.03 &    40.46 f
U2936/CO (FADD)                                  1.03 &    41.49 f
U2935/S (FADD)                                   2.65 &    44.14 r
DIGITAL_PART/binary_ff_o_reg[8]/
                   D0 (EDFMHRLQ)                 0.03 &    44.17 r
data arrival time                                          44.17

clock clk (rise edge)                           50.00      50.00
clock network delay (propagated)                 0.92      50.92
clock reconvergence pessimism                    0.11      51.03
DIGITAL_PART/binary_ff_o_reg[8]/
                    C (EDFMHRLQ)                51.03 r
library setup time                              -1.68      49.35
data required time                                         49.35
-----------------------------------------------------------------------
data required time                                         49.35
data arrival time                                         -44.17
-----------------------------------------------------------------------
slack (MET)                                                 5.19
```

It can be observed that the system has no setup violations but there are hold violations. The setup violations which existed previously are mitigated by the redesign of the PID controller and the addition of the bank of flip-flops after the generation of the error signal which are described in section 3.10 and section 3.11 of chapter 3, respectively.

4.2.4 Results After the ECO Flow

To mitigate the timing violations, the ECO flow is run on the design. It is evident that the addition of the new cells as part of the ECO flow affects the combinational area, sequential area, area occupied by the buffers and inverter pairs as well as the number of nets, cells, buffers and inverter pairs. The timing information also changes. After the newly added cells are placed and routed in IC Compiler, the respective commands are run to extract the timing and area information as described in section 4.2.3. The results are presented here.

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

```
Report : timing
        -path full
        -delay min
        -max_paths 1
Design : Digital_Top
Scenario(s): minmax
Version: Q-2019.12-SP5-1
Date   : Sun Aug 27 18:35:30 2023
****************************************

 * Some/all delay information is back-annotated.
Wire Load Model Mode: Inactive.
        Scenario            : minmax
        Parasitic source    : LPE
        Parasitic mode      : RealRC
        Extraction mode     : MULTI_CORNER
        Extraction derating : -40/150

Information: Percent of Arnoldi-based delays = 30.74% on scenario
                                                          minmax

  Startpoint: DIGITAL_PART/Stability/error_reg_reg[8][6]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: DIGITAL_PART/Stability/error_reg_reg[9][6]
            (rising edge-triggered flip-flop clocked by clk)
  Scenario: minmax
  Path Group: clk
  Path Type: min

  Point                                    Incr       Path      Voltage
  -----------------------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (propagated)         2.20       2.20
  DIGITAL_PART/Stability/
      error_reg_reg[8][6]/C (EDFMHRLQ)
                                           0.00       2.20 r    3.60
  DIGITAL_PART/Stability/
      error_reg_reg[8][6]/Q (EDFMHRLQ)
                                           1.56       3.75 f    3.60
  DIGITAL_PART/Stability/
      error_reg_reg[9][6]/D0 (EDFMHRLQ)
                                           0.00 &     3.75 f    3.60
  data arrival time                                   3.75

  clock clk (rise edge)                    0.00       0.00
  clock network delay (propagated)         2.20       2.20
  DIGITAL_PART/Stability/
      error_reg_reg[9][6]/C (EDFMHRLQ)
                                           0.00       2.20 r
  library hold time                       -0.58       1.62
  data required time                                  1.62
```

```
  ------------------------------------------------------------
  data required time                                  1.62
  data arrival time                                  -3.75
  ------------------------------------------------------------
  slack (MET)                                         2.13


  Startpoint: i2c_cntrl/U_6/Protocolunit/counter/counter_reg[0]
              (rising edge-triggered flip-flop clocked by i2c_clk)
  Endpoint: i2c_cntrl/U_6/Protocolunit/counter/counter_reg[0]
            (rising edge-triggered flip-flop clocked by i2c_clk)
  Scenario: minmax
  Path Group: i2c_clk
  Path Type: min

  Point                                    Incr       Path      Voltage
  ------------------------------------------------------------------------
  clock i2c_clk (rise edge)                0.00       0.00
  clock network delay (propagated)         1.25       1.25
  i2c_cntrl/U_6/Protocolunit/
      counter/counter_reg[0]/C (DFSHSL)
                                           0.00       1.25 r    3.60
  i2c_cntrl/U_6/Protocolunit/
      counter/counter_reg[0]/Q (DFSHSL)
                                           1.16       2.41 r    3.60
  U2153/O (NAND2XL)                        0.44 &     2.85 f    3.60
  i2c_cntrl/U_6/Protocolunit/
      counter/counter_reg[0]/D (DFSHSL)
                                           0.00 &     2.85 f    3.60
  data arrival time                                   2.85

  clock i2c_clk (rise edge)                0.00       0.00
  clock network delay (propagated)         1.25       1.25
  i2c_cntrl/U_6/Protocolunit/
      counter/counter_reg[0]/C (DFSHSL)
                                           0.00       1.25 r
  library hold time                       -0.42       0.83
  data required time                                  0.83
  ------------------------------------------------------------
  data required time                                  0.83
  data arrival time                                  -2.85
  ------------------------------------------------------------
  slack (MET)                                         2.02

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : Digital_Top
Scenario(s): minmax
```

```
Version: Q-2019.12-SP5-1
Date   : Sun Aug 27 18:33:55 2023
****************************************

 * Some/all delay information is back-annotated.
Wire Load Model Mode: Inactive.
        Scenario            : minmax
        Parasitic source    : LPE
        Parasitic mode      : RealRC
        Extraction mode     : MULTI_CORNER
        Extraction derating : -40/150

Information: Percent of Arnoldi-based delays = 30.74% on scenario
                                                          minmax

  Startpoint: DIGITAL_PART/FSM/current_state_reg[0]
              (rising edge-triggered flip-flop clocked by clk)
  Endpoint: DIGITAL_PART/reg_tdc/tdc_reg_o_reg[889]
            (rising edge-triggered flip-flop clocked by clk)
  Scenario: minmax
  Path Group: clk
  Path Type: max

  Point                                    Incr       Path      Voltage
  -----------------------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (propagated)         2.69       2.69
  DIGITAL_PART/FSM/
      current_state_reg[0]/C (DFSRLQ)      0.00       2.69 r    3.00
  DIGITAL_PART/FSM/
      current_state_reg[0]/Q (DFSRLQ)      1.89       4.58 f    3.00
  U_PTECO_HOLD_BUF437/O (BF)               2.11 &     6.69 f    3.00
  U1482/O (INV)                            2.18 &     8.87 r    3.00
  U1707/O (NAND2XL)                        2.31 &    11.18 f    3.00
  U1711/O (INV)                            1.37 &    12.55 r    3.00
  U1709/O (OR2N2AXL)                       1.02 &    13.57 r    3.00
  U1737/O (BF3)                            2.12 @    15.69 r    3.00
  U1758/O (BF2)                            2.36 @    18.05 r    3.00
  U1759/O (INV)                            0.68 @    18.72 f    3.00
  U3724/O (NAND2)                          0.48 &    19.20 r    3.00
  U1788/O (BF3)                            1.85 @    21.05 r    3.00
  U1793/O (BF2)                            3.00 @    24.05 r    3.00
  U3754/O (OR2N2AXL)                       2.52 @    26.57 f    3.00
  U_PTECO_HOLD_BUF144/O (BF8)              0.99 &    27.56 f    3.00
  U_PTECO_HOLD_BUF659/O (DEL10)           15.84 &    43.39 f    3.00
  DIGITAL_PART/reg_tdc/
      tdc_reg_o_reg[889]/D0 (EDFMHRLQ)
                                           0.00 &    43.39 f    3.00
  data arrival time                                  43.39

  clock clk (rise edge)                   50.00      50.00
```

```
clock network delay (propagated)      2.64      52.64
DIGITAL_PART/reg_tdc/
    tdc_reg_o_reg[889]/C (EDFMHRLQ)
                                      0.00      52.64 r
library setup time                   -1.34      51.31
data required time                              51.31
-----------------------------------------------------------
data required time                              51.31
data arrival time                              -43.39
-----------------------------------------------------------
slack (MET)                                     7.91


Startpoint: DIGITAL_PART/FSM/calib_value_wrap_o_cld_reg[10]
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: i2c_cntrl/U_6/Protocolunit/shift_reg_out/shift_reg_reg[2]
          (rising edge-triggered flip-flop clocked by i2c_clk')
Scenario: minmax
Path Group: i2c_clk
Path Type: max

Point                                    Incr       Path      Voltage
-----------------------------------------------------------------------
clock clk (rise edge)                  450.00     450.00
clock network delay (propagated)         2.57     452.57
DIGITAL_PART/FSM/
    calib_value_wrap_o_cld_reg[10]/C (DFSRLQ)
                                         0.00     452.57 r    3.00
DIGITAL_PART/FSM/
    calib_value_wrap_o_cld_reg[10]/Q (DFSRLQ)
                                         2.07     454.64 r    3.00
U_PTECO_HOLD_BUF18/O (DEL10)            16.53 &   471.17 r    3.00
U5951/O (AND22OXL)                       0.92 &   472.08 f    3.00
U5953/O (NAND4XL)                        1.05 &   473.13 r    3.00
U5967/O (NOR3XL)                         1.19 &   474.32 f    3.00
U5968/O (OR2N2AXL)                       0.68 &   474.99 r    3.00
i2c_cntrl/U_6/Protocolunit/
    shift_reg_out/shift_reg_reg[2]/D0 (EDFMHRLQ)
                                         0.00 &   475.00 r    3.00
data arrival time                                 475.00

clock i2c_clk' (rise edge)             500.00     500.00
clock network delay (propagated)         1.50     501.50
i2c_cntrl/U_6/Protocolunit/
    shift_reg_out/shift_reg_reg[2]/C (EDFMHRLQ)
                                         0.00     501.50 r
library setup time                      -1.37     500.13
data required time                                500.13
-----------------------------------------------------------------------
data required time                                500.13
data arrival time                                -475.00
```

```
  ----------------------------------------------------------------
  slack (MET)                                          25.13

Library (s) Used:
ELMOSOU35HD_ SS (File: /eda/kits/ELMOS/L035/pdk/athen/3.0.3/
                       L035_libraries/ELMOSOU35HD/db/elmos0u35hd_ss.db)
Number of ports:                       931
Number of nets:                       9912
Number of cells:                      7914
Number of combinational cells:        6358
Number of sequential cells:           1556
Number of macros/black boxes:            0
Number of buf/inv:                    2668
Number of references:                   70

Combinational area:          710407.589951
Buf/Inv area:                232962.743307
Noncombinational area:       465785.505920
Macro/Black Box area:             0.000000
Net Interconnect area:       undefined (No wire load specified)

Total cell area:            1176193.095871
Total area:                  undefined
```

After the addition of the new cells into the existing netlist, the placement of the cells and the routing interconnects change. As a result, the layout also changes. The layout of the design after the addition of new cells is shown in figure 4.16.

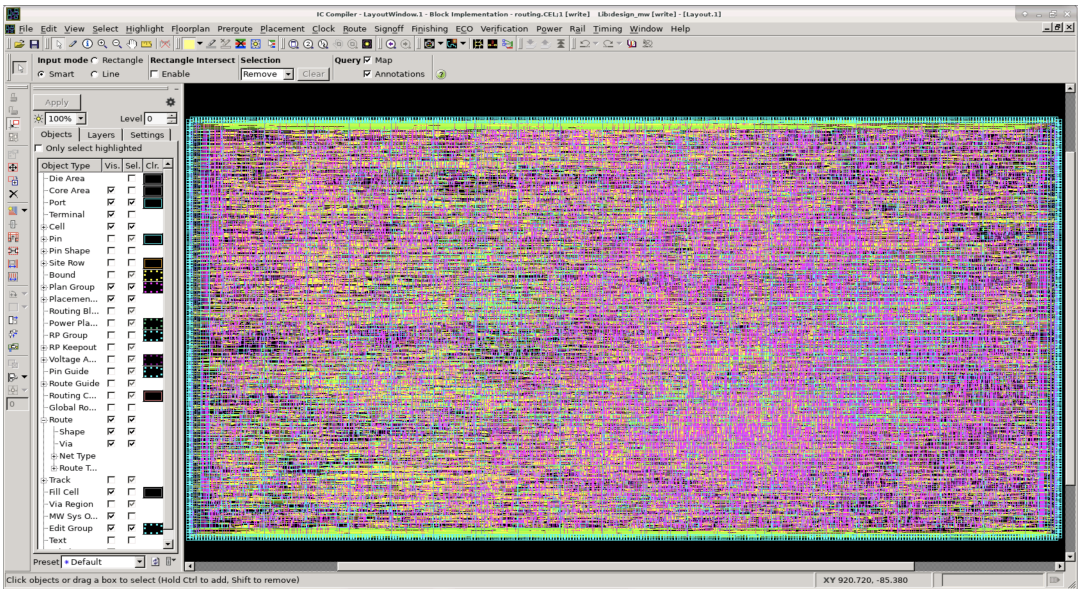

FIGURE 4.16: Layout Window of IC Compiler After ECO

After the newly added cells are placed and routed, parasitic extraction needs to be carried out on the changed netlist for accurate delay calculations. After running the parasitic extraction in StarRC, the signoff STA is carried out in PrimeTime. These steps are similar to the steps described in section 4.2.3. The signoff STA of the min

and max corners are carried out and the results are shown in figure 4.17 and figure 4.18, respectively. The timing reports are also presented. It can be seen that there are no more hold violations after the ECO flow is performed on the design.

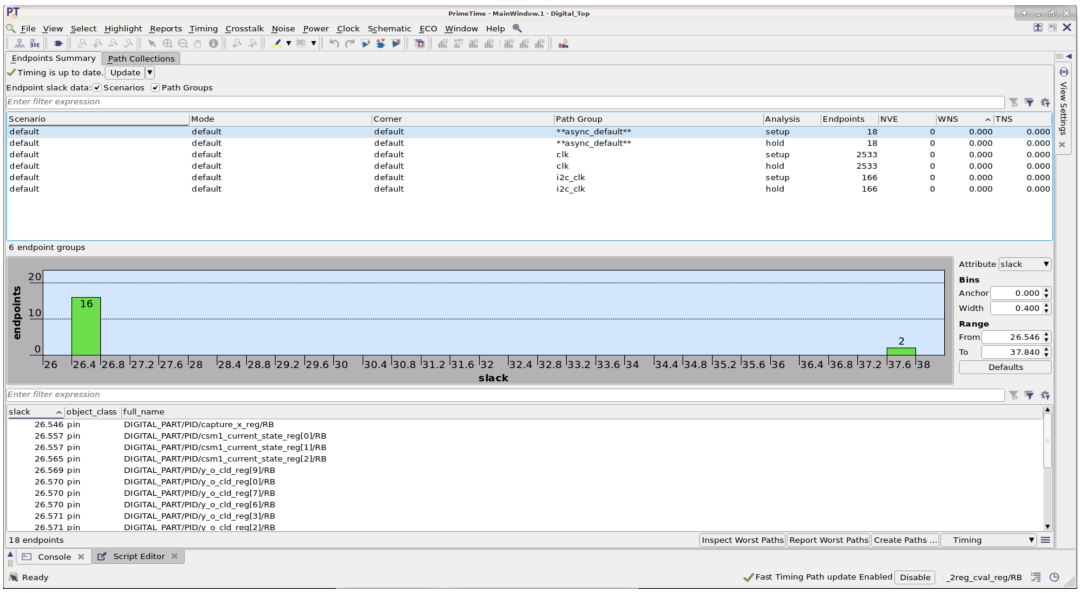

FIGURE 4.17: Timing Analysis of the Min Corner After ECO

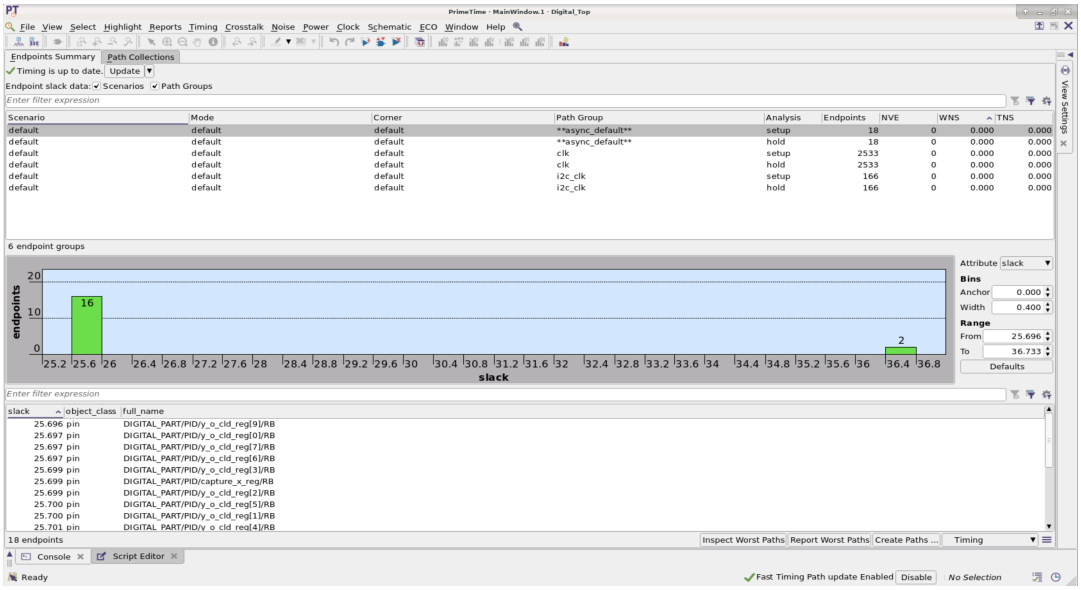

FIGURE 4.18: Timing Analysis of the Max Corner After ECO

```
****************************************
Report : timing
-path_type full
-delay_type min
-max_paths 1
-sort_by slack
Design : Digital_Top
Version: Q-2019.12-SP1
Date   : Sun Aug 27 18:45:19 2023
```

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


  Startpoint: i2c_cntrl/U_6/Protocolunit/shift_reg_out/shift_reg_reg[3]
              (rising edge-triggered flip-flop clocked by i2c_clk')
  Endpoint: i2c_cntrl/U_6/Protocolunit/shift_reg_out/shift_reg_reg[4]
              (rising edge-triggered flip-flop clocked by i2c_clk')
  Last common pin: INV4_G1B6I3_1/O
  Path Group: i2c_clk
  Path Type: min

  Point                                             Incr       Path
  ------------------------------------------------------------------------
  clock i2c_clk' (rise edge)                     500.00     500.00
  clock network delay (propagated)                 0.48     500.48
  i2c_cntrl/U_6/Protocolunit/
      shift_reg_out/shift_reg_reg[3]/C (EDFMHRLQ)
                                                   0.00     500.48 r
  i2c_cntrl/U_6/Protocolunit/
      shift_reg_out/shift_reg_reg[3]/Q (EDFMHRLQ)
                                                   0.55 &   501.04 f
  U6007/O (OR2N2AXL)                               0.38 &   501.42 f
  i2c_cntrl/U_6/Protocolunit/
      shift_reg_out/shift_reg_reg[4]/D0 (EDFMHRLQ)
                                                   0.00 &   501.42 f
  data arrival time                                         501.42

  clock i2c_clk' (rise edge)                     500.00     500.00
  clock network delay (propagated)                 1.62     501.62
  clock reconvergence pessimism                   -0.12     501.49
  i2c_cntrl/U_6/Protocolunit/
      shift_reg_out/shift_reg_reg[4]/C (EDFMHRLQ)
                                                 501.49 r
  library hold time                               -0.15     501.34
  data required time                                        501.34
  ------------------------------------------------------------------------
  data required time                                        501.34
  data arrival time                                        -501.42
  ------------------------------------------------------------------------
  slack (MET)                                                 0.07

****************************************
Report : timing
-path_type full
-delay_type max
-max_paths 1
-sort_by slack
Design : Digital_Top
Version: Q-2019.12-SP1
Date   : Sun Aug 27 18:41:14 2023
****************************************
```

```
Startpoint: DIGITAL_PART/FSM/current_state_reg[0]
             (rising edge-triggered flip-flop clocked by clk)
Endpoint: DIGITAL_PART/reg_tdc/tdc_reg_o_reg[889]
             (rising edge-triggered flip-flop clocked by clk)
Last common pin: INV5_G1B10I2/O
Path Group: clk
Path Type: max

Point                                          Incr       Path
-----------------------------------------------------------------------
clock clk (rise edge)                          0.00       0.00
clock network delay (propagated)               2.75       2.75
DIGITAL_PART/FSM/current_state_reg[0]/
                            C (DFSRLQ)         0.00       2.75 r
DIGITAL_PART/FSM/current_state_reg[0]/
                            Q (DFSRLQ)         1.91 &     4.66 f
U_PTECO_HOLD_BUF437/O (BF)                     2.26 &     6.92 f
U1482/O (INV)                                  2.32 &     9.24 r
U1707/O (NAND2XL)                              2.51 &    11.76 f
U1711/O (INV)                                  1.47 &    13.23 r
U1709/O (OR2N2AXL)                             1.03 &    14.26 r
U1737/O (BF3)                                  2.34 &    16.60 r
U1758/O (BF2)                                  2.50 &    19.10 r
U1759/O (INV)                                  0.71 &    19.81 f
U3724/O (NAND2)                                0.50 &    20.31 r
U1788/O (BF3)                                  2.05 &    22.35 r
U1793/O (BF2)                                  3.23 &    25.58 r
U3754/O (OR2N2AXL)                             2.71 &    28.30 f
U_PTECO_HOLD_BUF144/O (BF8)                    1.02 &    29.32 f
U_PTECO_HOLD_BUF659/O (DEL10)                 15.87 &    45.19 f
DIGITAL_PART/reg_tdc/tdc_reg_o_reg[889]/
                           D0 (EDFMHRLQ)
                                               0.00 &    45.19 f
data arrival time                                        45.19

clock clk (rise edge)                         50.00      50.00
clock network delay (propagated)               0.93      50.93
clock reconvergence pessimism                  0.11      51.04
DIGITAL_PART/reg_tdc/tdc_reg_o_reg[889]/
                            C (EDFMHRLQ)      51.04 r
library setup time                            -1.40      49.64
data required time                                       49.64
-----------------------------------------------------------------------
data required time                                       49.64
data arrival time                                       -45.19
-----------------------------------------------------------------------
slack (MET)                                              4.45
```

Chapter 5

Conclusion and Future Work

The goal of this thesis was to introduce the ToF camera technology and the principle that lies underneath the concept and development of the delay asymmetry compensation logic of Panoptes Testchip. Furthermore, it describes the ideas and principles behind the addition of extra hardware logic for the debugging features that facilitate an interactive environment for the user to identify any abnormal operation, in case there occurs any. Also, the concepts and principles elaborated in this thesis give an idea regarding the measurement of distance in centimeter range. All these concepts and principles cater to the need of "intelligent mobility" for autonomous driving functionalities.

There are certain areas in the existing design where further improvements can be possible. Since outputs of TDC are very sensitive to the arrival time of startstop pulse which in turn affects the measurement results, the TDC can be redesigned for an optimal solution. The digital part can be further optimized to reduce gate counts which in turn reduces area and cost. The PID controller designed for the system is complex as well as extremely powerful which serves more than what the system demands. So, a fairly simpler PID module could be designed. The **AllowedAbsErr** parameter of the stability analysis module greatly influences the time of assertion of the stability flag which has an impact on the overall system performance. Therefore, the effects of varying this parameter with respect to the parameters of the TDC model as well as the delays of TIA, comparator, predelay element, shutter and laser drivers can be studied further and more simulations can be carried out to further improve the whole system. The implementation scripts used for the backend digial design could be further improved. Interrupt signals can be generated in case of any erroneous operations. Also, some additional test logic can be introduced into the digital part. To make the design verification more robust and efficient, a SystemVerilog testbench environment can be created. By using the components like driver, monitor, scoreboard etc., this SystemVerilog 'layered testbench' approach can facilitate the generation of random stimulus to catch hidden bugs in the design. Also, assertion based verification can be introduced to increase the functional coverage. Since functional safety is of utmost concern in case of automotive applications, some additional design concepts could be introduced for the same.

Achieving high precision in the centimeter range for distance measurement is still a challenge. This research project incorporates innovative research concepts and methodologies to address that issue and results in an improvement in the accuracy of ToF distance measurement. Moreover, this thesis along with the previous works[3][16][6] can provide with a great learning opportunity for someone who is new to the field of digital design.

Bibliography

[1] Prof. Dr.-Ing. Michael Karagounis et al. *Development of Time-of-flight 3D and polarization camera for automotive applications*.

[2] Stephan Henzler. *Time-to-Digital Converters. Springer Series in Advanced Microelectronics*. Springer, 2010.

[3] Felix Schneider. *Delay asymmetry Compensation Logic*.

[4] John K. Ousterhout. *Tcl and the Tk Toolkit*.

[5] "IEEE Standard for Floating-Point Arithmetic". In: IEEE Std 754-2019 (Revision of IEEE 754-2008), pp. 1–84. DOI: 10.1109/IEEESTD.2019.8766229. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8766229&isnumber=8766228.

[6] Andreas Pille. *Optimierung eines Local Passive Interpolation Time-to-Digital Converters mit Sub-Gate Delay für eine Time-of-Flight Anwendung*. 2021.

[7] Alexander Horstkötter. *MatlabModelExplanation*. Microsoft Powerpoint Presentation.

[8] Jonathan Valdez and Jared Becker. *Understanding the I2C Bus*. Tech. rep. Texas Instruments, 2015. URL: www.ti.com/lit/an/slva704/slva704.pdf.

[9] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Signals and Systems*.

[10] Erwin Kreyszig. *Advanced Engineering Mathematics*.

[11] Katsuhiko Ogata. *Discrete-Time Control Systems*.

[12] "IEEE Standard for Verilog Hardware Description Language". In: IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001), pp. 1–590. DOI: 10.1109/IEEESTD.2006.99495. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1620780&isnumber=33945.

[13] "IEEE Standard for Verilog Register Transfer Level Synthesis". In: IEEE Std 1364.1-2002, pp. 1–108. DOI: 10.1109/IEEESTD.2002.94220. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1146718&isnumber=25839.

[14] "IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language". In: IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012), pp. 1–1315. DOI: 10.1109/IEEESTD.2018.8299595. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8299595&isnumber=8299594.

[15] Stormy Attaway. *MATLAB: A Practical Introduction to Programming and Problem Solving*.

[16] Jitikantha Sarangi. *Synopsys Digital Design Flow and Use of Synopsys Toolchain in Digital Design*. 2021.

[17] Synopsys Support Community. URL: https://solvnetplus.synopsys.com/.

[18] K. Åström and T. Hägglund. *PID Controllers: Theory, Design, and Tuning*.

Appendix A

MATLAB Code for Plotting TDC Outputs with respect to Predelay

```
x = linspace(0, 5000, 51);
time = (0:100*10^-12:5*10^-9)';
startstop_edge1_edge3 = importdata('tdc_count_edge1_edge3.
    txt');
subplot(2,2,1)
plot(x,startstop_edge1_edge3)
ylim([435 465])
title('Startstop pulse- edge\_1 & edge\_3')
xlabel('Pre delay time in ps')
ylabel('TDC\_reg count')

startstop_edge3_edge5 = importdata('tdc_count_edge3_edge5.
    txt');
subplot(2,2,3)
plot(x,startstop_edge3_edge5)
ylim([435 465])
title('Startstop pulse- edge\_3 & edge\_5')
xlabel('Pre delay time in ps')
ylabel('TDC\_reg count')

startstop_edge2_edge4 = importdata('tdc_count_edge2_edge4.
    txt');
subplot(2,2,2)
plot(x,startstop_edge2_edge4)
ylim([435 465])
title('Startstop pulse- edge\_2 & edge\_4')
xlabel('Pre delay time in ps')
ylabel('TDC\_reg count')

startstop_edge4_edge6 = importdata('tdc_count_edge4_edge6.
    txt');
subplot(2,2,4)
plot(x,startstop_edge4_edge6)
ylim([435 465])
title('Startstop pulse- edge\_4 & edge\_6')
xlabel('Pre delay time in ps')
ylabel('TDC\_reg count')
```