Universidade do Minho
Escola de Engenharia

João Afonso Magalhães de Azevedo

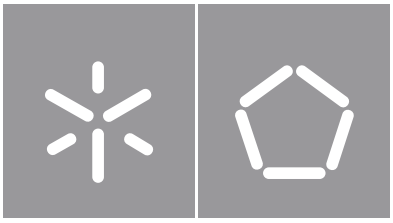
LiDAR: Reconfigurable Hardware Based Data Acquisition

October 2019

LiDAR: Reconfigurable Hardware Based Data Acquisitiontítulo

João Azevedo

UMinho | 2019



Universidade do Minho
Escola de Engenharia

João Afonso Magalhães de Azevedo

LiDAR: Reconfigurable Hardware Based Data Acquisition

Dissertação de Mestrado
Engenharia Eletrónica Industrial e Computadores
Sistemas Embebidos e Computadores

Trabalho efetuado sob a orientação do(a)
Professor Doutor Jorge Cabral

março de 2019

Acknowledgements

I would like to thank Professor Jorge Cabral for the opportunity of developing this project, and for all the guidance given in the development of this work. I would especially like to thank Rui Machado for all the guidance and support provided throughout the project. Furthermore, for the patience and willingness to help me whenever problems arose.

A huge thanks to all my friends, for every good moment and for every advice and incentives given. To Filipa, who was always there to help, offer advice, and help me gain a new breath whenever I felt less motivated, and also for the fantastic thesis template. To Sérgio and Miguel for being my partners in this last year and for all the hilarious chats during lunch. To Sara and Daniel for the friendship and companionship throughout the course. To Álvaro for being my short friend. And to Daniel, Luis, and Joana for being there every step of the way since always.

To my family for everything they have done for me.

This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 037902; Funding Reference: POCI-01-0247-FEDER-037902] .

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Resumo

É expectável que nas próximas décadas exista um aumento na procura das ADAS, potenciado pelos interesses dos reguladores e dos consumidores em aplicações que protejam o condutor e reduzam o número de acidentes. Tanto os OEMs, como os seus fornecedores aperceberam-se que, apesar das ADAS ainda estarem numa fase inicial, podem-se tornar uma característica diferenciadora entre as diversas marcas de automóveis, e por isso, uma das suas principais fontes de rendimento. Além disso, as tecnologias usadas nas ADAS poderão vir a ser utilizadas para criar veículos autónomos, os quais se estão a revelar como um dos principais focos da pesquisa e desenvolvimento.

Existem três principais soluções de sensores usadas nas ADAS. Primeiro, existem as soluções baseadas em sensores óticos, que são as soluções mais versáteis e económicas. No entanto, este tipo de soluções é facilmente afetado pelo mau tempo e outros fatores ambientais. Para além do facto de necessitarem o uso de algoritmos complexos para reconhecerem objectos. A segunda solução incorpora o uso de RADARs de longo e curto alcance, com o objetivo de determinar a distância, velocidade e direção dos objetos. Estes sensores são pouco afetados por condições meteorológicas adversas. Porém, existe um compromisso entre o alcance e o ângulo de medição do sensor.

A última solução envolve o uso de sistemas de LiDAR. Estes sistemas usam pulsos de laser para examinar meio-envolvente, de modo a gerar uma imagem tridimensional completa do mesmo. O LiDAR é menos sensível à luz e às condições meteorológicas e consegue fornecer diretamente a localização dos objetos à sua volta. Devido à crescente utilização das ADAS, existe a necessidade de desenvolver sensores LiDAR mais avançados. Para suprir essa necessidade e para ultrapassar algumas das limitações dos sensores atuais, a divisão Chassis Systems Control, do grupo Bosch, está atualmente a desenvolver uma solução de um sensor LiDAR para a indústria automóvel, projeto onde se insere esta dissertação.

Nesta dissertação foi desenvolvido um Sistema de Aquisição para o sensor LiDAR. Este sistema mede o TOF dos pulsos de laser usado pelo LiDAR. Para isso, vários periféricos de TDC foram desenvolvidos numa FPGA. A precisão de medição do sistema varia entre os 232.17 ps e os 188.66 ps, com um valor médio de 207.47 ps.

Palavras Chave: *ADAS*, *Chassis Systems Control*, *FPGA*, *LiDAR*, *TDC*

Abstract

There is an expected increase in the demand for Advanced Driver-Assistance Systems (ADAS) over the next decade, incited by regulatory and consumer interest in safety applications that protect drivers and reduce accidents [1]. Even though ADAS applications are still beginning, both the OEMs and their suppliers are realizing that they could become one of the essential characteristics differentiating the various automotive brands, consequently, one of their most important revenue sources. Furthermore, the technologies used in ADAS could be used in the future to create fully autonomous vehicles, which are now becoming a major focus of research and development.

There are three main sensor solutions used in ADAS. Firstly, there are optical sensors and camera based-solutions. These are the most versatile and cost-efficient solutions. However, they are easily affected by poor weather and other environmental hazards. Furthermore, they require complex software algorithms to recognize objects [1]. The second solution incorporates short and long range Radars for determining the distance, speed, and direction of objects. These sensors work better than the others in adverse weather conditions. Nonetheless there is typically a compromise between the measurement range and angle [1].

The last type of solution involves using LiDAR systems, which use laser pulses to scan the surroundings and generate a complete and precise three-dimensional image of the environment. The LiDAR is less sensitive to light and weather conditions than optical systems and provides the location of the surrounding objects directly. Due to the ever-growing use of ADAS, there is a need to develop a more advanced LiDAR sensor. To answer that need and to overcome some of the limitations of the current LiDAR sensors, the Chassis Systems Control of the Bosch Group is developing an automotive LiDAR, and the current *Master's thesis* is integrated in the project.

In this *Master's thesis*, an Acquisition System for Bosch's LiDAR sensor was developed. For measuring the Time-of-Flight of the laser pulses of the LiDAR, to do so multiple TDC Peripherals were developed in an FPGA platform. The measurement precision of the developed Acquisition System varies between 232.17 ps and 188.66 ps, with an average precision of 207.47 ps.

Keywords: *ADAS*, *Chassis Systems Control*, *FPGA*, *LiDAR*, *TDC*

Table of Contents

List of Figures

List of Tables

List of Listings

List of equations

Acronyms List

ABS Anti-lock Braking System.

ACC Adaptive Cruise Control.

ADAS Advanced Driver-Assistance Systems.

ADC Analog-to-Digital Converter.

AMBA Arm Advanced Microcontroller Bus Architecture.

ASIC Application-Specific Integrated Circuit.

AV Autonomous Vehicles.

AXI Advanced eXtensible Interface.

BSD Blind Spot Detection.

Carry Fast Carry Logic.

CIN Carry In.

CLB Configurable Logic Block.

CO Carry Out.

DNL Differential nonlinearity.

DSP Digital Signal Processor.

ECU Electronic control units.

ESC Electronic Stability Control.

FCW Forward Collision Warning.

FF Flip-Flop.

FIFO First In First Out.

FPGA Field-Programmable Gate Array.

GPS Global Positioning System.

HA Highway Assist.

INL Integral nonlinearity.

LDW Lane Departure Warning.

LiDAR Light Imaging Detecton and Ranging.

LSB Least Significant Bit.

LUT Look-up-Table.

MCU Microcontroller Unit.

OEM Original Equipment Manufacturer.

PCB Printed Circuit Board.

PS Processing System.

PVT Process, Voltage, Temperature.

Radar Radio Detection And Ranging.

RAM Random Access Memory.

RMS Root-Mean Square.

RTOS Real Time Operating System.

TCS Traction Control System.

TDC Time-to-Digital Converter.

TDL Tapped-Delay Line.

TOF Time-of-Flight.

TSR Traffic Sign Recognition.

Chapter 1

Introduction

The uses and functionalities of the Advanced Driver-Assistance Systems (ADAS) had a substantial evolution since the first ADAS was introduced in the market in 1978 by Bosch, with the mass-production of the Anti-lock Braking System (ABS) [2]. From that moment on, a vast array of different ADAS was developed and commercialized by various companies, their use has become increasingly widespread firstly in luxury cars, and more recently even in entry-level cars. Some of them, due to the associated safety enhancements are so widespread that they have become mandatory in every new car, an example of this is the use of the Electronic Stability Control (ESC), that became a legal requirement in the European Union since 2014 [2].

Although ADAS are on the brink of becoming ubiquitous in every new car [1], with the both the European Union and the United States mandating that every vehicle has to be equipped with autonomous emergency-braking systems and forward-collision warning systems by 2020 [1], they still have an enormous innovation potential, with the development of new automotive sensors like the LiDAR. Moreover, with the improvement of the existing sensors, a new range of ADAS is on the verge of becoming feasible. Thse new ADAS will benefit from the higher quality data provided by the augmented sensor, and new sensors like the LiDAR, and the ever-increasing available computational power in low-power devices, that will allow the creation of multi-sensor systems that use data fusion techniques in order to acquire a richer and complex dataset that provides a more comprehensive description of the surroundings of the vehicle [3].

The data obtained using data fusion in the automotive context has two major benefits [4]: enhanced coverage, where data from sensors with separate fields-of-view are fused to achieve a broader overall coverage; and increased confidence, in which data from sensors with shared fields-of-view are used to validate object detection.

Hence, the availability of more complete and reliable data will enable the possibility of developing ADAS with the objective of implementing level 4 and level 5 autonomous vehicles, a level of automation where the driver input is reduced to a minimum [5]. This higher level Autonomous Vehicles (AV) is now

under-development and is expected to become available in new cars by the beginning of the next decade [6].

1.1 Contextualization

Mobility is fundamental nowadays, and it is directly correlated with the social well-being of the population and the amelioration of their quality of life. It is the base of all commercial trading and, therefore, is essential for economic success [2].

Nevertheless, the advantages of the rise of mobility come with a price they implicate a grave problem for society and the populations. Beyond the obvious cost related to vehicles, fuel, the construction and maintenance of the road and automobile infrastructure, mass mobilization carries various economic, societal and environmental problems: consumption of fossil fuels and other resources, noise and exhaust pollution, the loss of productivity due to traffic congestion and the often fatal, traffic incidents [2]

There are several measures and technological solutions put in place in order to tackle the problems of mobility. However, there is a new technological solution that is presently becoming available that could change mobility and the problems connected to it forever, Autonomous Vehicles (AV). Although automobiles currently possess some type of automation, usually level 1 or level 2 automation in newer cars, with level 3 automation available in luxury cars, these levels of automation still requires the driver to perform the vast majority of the driving tasks, as it can be seen in Figure 1.1, thus it does not substantially reduce the most significant factor of accidents, the human error [2]. In order to diminish this factor, a higher level of automation needs to be achieved, as only in level 4 and level 5 automation does the driver input become minimal [7].

When level 5 automation is achieved, profound changes will affect the way mobility occurs. Currently, there are two main scenarios of how this change can happen. In the first scenario, autonomous cars are owned by ride-sharing services that dispatch AV to pick up several passengers traveling along similar routes, where AV are used to connect passengers between their home and public transport which would be used in more distant commutes. Many traditional car owners could decide to no longer own a personal vehicle as shared-mobility AV would fulfill their needs. Hence, road congestion would drop because there would be fewer vehicles [8].

In the other scenario, the number of cars purchased increases dramatically with the availability of AV, since everybody could now use a car even without a drivers license. The increase in the number of cars

would lead to a shortage of parking spaces that would force AV to circle the roads in search of available spots which would make the roads even more congested. This would increase the number of miles traveled by year and the pollution associated with it [8].

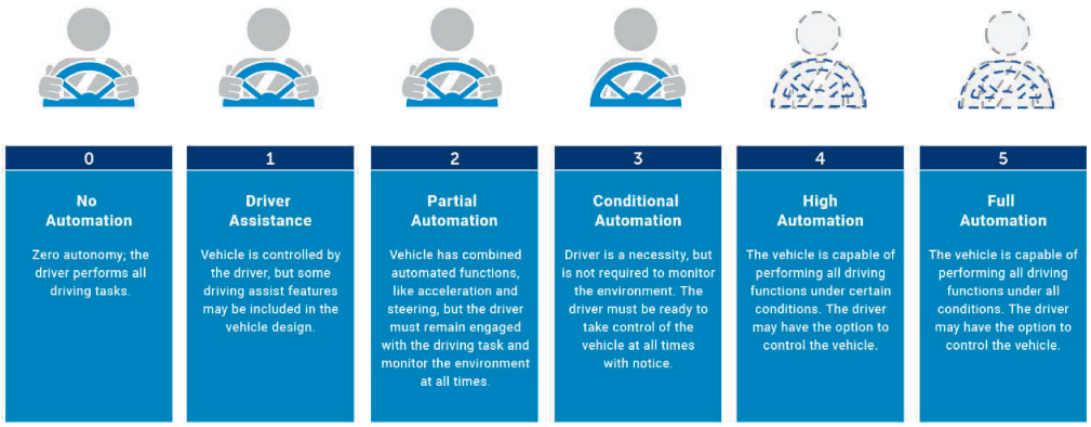


Figure 1.1: Automated Driving Levels has defined by the Society of Automotive Engineers. Extracted from [7]

As high-level automation in vehicles becomes imminent, automotive companies publish road-maps of how they will implement it in their own products. Most companies plan on releasing automobiles with highway self-driving capabilities by 2020 or 2021. While some have more ambitious plans like Renault-Nissan that plans on releasing AV by 2020 and truly drive-less cars by 2025, while Ford plans on having a level 4 vehicle in 2021 with no gas pedal, no steering wheel and where the passenger will not have to assume control in a predefined area [6]. Despite of this first wave of AV, most companies believe that there is still a long time before true level 5 autonomy is achieved [6].

As it can be seen above, although self-driving cars are imminent, there is still a long way to go until true automation exists. Mainly due to the necessity of improvements mainly on four different technologies: processors, sensors, software algorithms, and mapping [1]. Both the Electronic control units (ECU) and the Microcontroller Unit (MCU) is essential for most ADAS applications. Therefore for ADAS to advance processors need better performance, but still maintaining the low-power consumption requirements.

The sensors gather data of their immediate environment; they have a limitation in their measurement range, are sensible to the weather conditions and have difficulty in tracking moving objects. So, many industry players are trying to improve individuals sensors, additionally, there have been attempts to optimize the systems using sensor fusion [1]. The limited functionality and high cost of current sensors may be the most significant constraint to ADAS uptake.

The software algorithms use the data collected by the sensors to synthesize the environment surrounding a vehicle in real-time, providing an output to the driver or specify how a system should actively intervene in vehicle control [1]. Which requires the implementation of very robust and complex software to be deployed in the car since every decision of this software is critical to ensuring the safety of the passenger. These algorithms should also accurately predict the possible human behavior in various situations, to correctly act accordingly with it. The mapping of the road and the accurate detection of the car's position in it can aid the software algorithms as some constraints and conditions of the road could be already known to let the software to take measures to enhance the security of the passengers preemptively. The detection of the position of the car is based on the use of GPS. However, its coverage sometimes fails; when that happens, the car should have accurate and detailed mapping system to help prevent accidents [1].

1.2 Motivation

Creating and enhancing the ADAS sensors is a necessary step to achieve a level 4 or level 5 automation. Currently, vehicles depend on Radar and cameras to acquire information about the surroundings, however, the data collected by them is not complete enough to be used in higher-level automation, so to complete this data, a new sensor is being used, the Light Imaging Detecton and Ranging (LiDAR).

The LiDAR uses a laser, with an invisible wavelength, to emit a pulse to a target object and measure the Time-of-Flight (TOF) of the pulse from emission to reception. As light moves at a constant speed, the distance between the object and the sensor can be determined with high precision [9]. The LiDAR provides a higher range and accuracy compared to camera vision systems, and it offers a sturdy operation in all lighting conditions, while cameras have problems with luminosity changes, can be blinded by direct light and have limited use in poorly lit conditions. The Radar is a good sensor for basic ADAS functions, as radio signals are less affected by environmental factors than light, yet it is not as reliable for detecting pedestrians, static objects or objects that move laterally, as LiDAR. It is also sensible to interference with other Radars or refraction of its own signal [9]. Therefore the use of a LiDAR can complement the data acquired by the Radar and the camera systems, and the use of sensor-fusion algorithms to understand the environment at a more granular level [10], making it a fundamental sensor in the future of ADAS systems.

Hence huge investments are being made by OEM and suppliers in the development of new LiDAR technologies [11]. One of the fundamental components of the LiDAR is the Time-to-Digital Converter

(TDC), which in combination with other electronic circuits is used to measure the TOF of a laser pulse, has it can be seen in Figure 1.2.

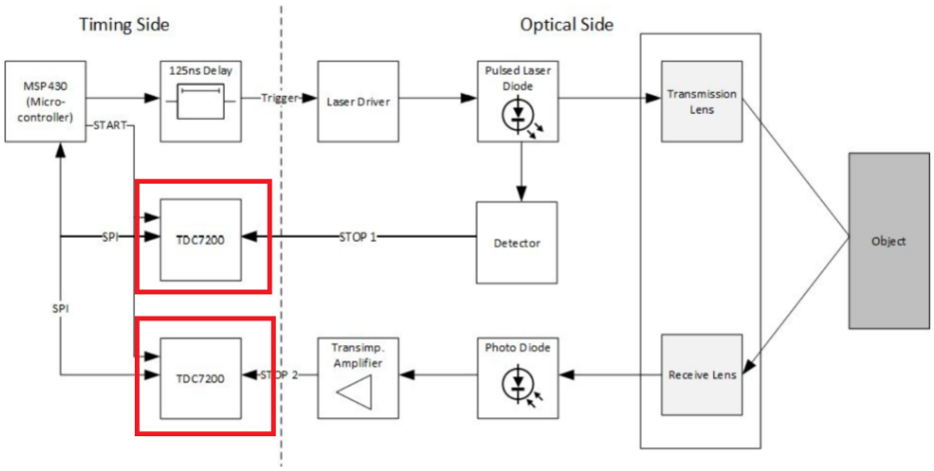


Figure 1.2: Block Diagram of a Texas Instruments LiDAR. Extracted from [12]

This dissertation is part of a project that is being developed by the Chassis Systems Control division, of the Bosch group, which aims to develop a new LiDAR solution for the automotive industry. The sensor will contain various improvements compared to the solutions that are available in the market, for example, it should allow greater precision in the measurement of distances, as well as to characterize the velocity of the target in a single measurement. In the scope of this dissertation, a new TDC will be developed in FPGA. The TDC should have a high resolution, and additionally, it should have an auto-calibration method in order to compensate for the effect of fluctuations in the temperature and supply voltage of the sensor. Finally, it should have multi-channel capabilities due to LiDAR using more than one TDC channel.

1.3 Objectives

After taking in account the motivation, it is the goal of this dissertation to develop a Acquisition System based in TDC for FPGA for a LiDAR Sensor. As so, the main goals of this dissertation are as follows:

- Study the various sensors used in ADAS;
- Study the different TDC architectures;

- Development a TDC Peripheral in a FPGA;
- Development of an auto-calibration system for the TDC Peripheral;
- Implementation of an Acquisition System with sixteen TDC Peripherals, implementing the necessary additional logic;

1.4 Dissertation Structure

The document is divided into six chapters, with a structure that follows the same logical order as the development process of this *Master's Thesis*.

The first chapter introduces the current technological concepts, referring the context and the motivation for the development of the current project, and also its objectives.

The second chapter analyzes the core concepts of this thesis, giving a more in-depth overview of the ADAS applications and the sensors used in them. It has emphasis in the LiDAR sensor, as it is the goal the project is to develop a acquisition System for it. Finally, there is a thorough examination of the TDCs, where the terminologies used are explained, and the different architectures are exposed and compared.

The third chapter gives a overview of how the system is designed, and then explains in detail the design of each module of TDC. Additionally it also explains how the test of the system work and how to interpret its results.

The fourth chapter focuses on explaining how each module of TDC was developed and implemented on FPGA.

The fifth chapter presents the results of the tests of Acquisition System, drawing some conclusions about them.

Chapter six presents the main conclusions of the project, and proposes solutions for the problems identified in the current Acquisition System.

Chapter 2

State of the Art

In order to develop a data acquisition solution for automotive LiDAR, it is first fundamental to understand what are the various existing ADAS applications and what is the objective of their use. It is equally important to have the context of where the LiDAR is used in ADAS. As well it is crucial to study the technologies associated with the LiDAR and the platform where it will be developed. Furthermore, it is likewise necessary to know which other types of sensors are used in ADAS, to identify which are the advantages and disadvantages of the LiDAR compared to the other sensors.

Additionally, it is also necessary to research the different Re-configurable Hardware platforms available to perceive which platform is more suited for being used in the TDC and how each platform might affect the TDC implementations. Finally, it is required to study the various existing TDC architectures, taking into account the advantages and disadvantages of each of them.

Hence, this chapter provides a technical and technological overview of all the topics mentioned before, providing an in-depth analysis of them.

2.1 Advanced Driver-Assistance Systems (ADAS)

The ADAS is a set of technologies and systems with the goal of helping the driver with the driving process.They are designed to enhance the driving experience, making it safer and more comfortable, and diminishing the risks associated with it. They provide information to the driver and automate difficult and repetitive tasks. Although they cannot entirely prevent accidents, the ADAS can better protect the driver from some human errors, which are the cause of most traffic accidents [2]. The first ADAS introduced in the market was the Anti-lock Braking System (ABS) in 1978. Since then, there are many new ADAS, bringing a higher level of automation to the driving task.

There are several levels of automation in ADAS, from level 0 automation in which the ADAS only provide information to the driver, to level 4 and level 5 automation where the vehicle is fully autonomous needing little intervention from the driver. Hence it is important to study the various levels of ADAS automation to understand the various ADAS applications fully.

2.1.1 Level 0 Systems

Level 0 ADAS do not perform any vehicle control task; they provide information to the driver helping him monitor the surrounding environment or his own status [5]. Some examples of this type of systems are:

Parking Systems

Parking systems were introduced on cars in the 1980s, they provide an acoustic, and in more modern systems, a visual warning, depending on the distance between the vehicle and surrounding objects while parking the car, helping the driver to avoid damaging the car [5].

Traffic Sign Recognition (TSR)

The TSR shows the driver information such as current speed limits and other road rules and warnings. These functionalities use a forward-looking camera to detect and read the road signs in combination with other information sources like navigation maps [5].

Blind Spot Detection (BSD)

The system warns the driver if there is an obstacle in the blind spot of the rear-facing mirrors, which the driver may not have noticed, using visual and acoustic warnings if the drivers show an intention to change the trajectory that may lead to a collision with the obstacle [5].

2.1.2 Level 1 Systems

This type of systems takes care of a single functionality in specific cases. They still leave the authority to the driver, but control the vehicle using actuators.

Anti-lock Braking System (ABS)

The Anti-lock Braking System supports the driver while braking, avoiding wheel blockage even when the full brake force is required, avoiding tire saturation, which allows the reduction of braking distance and keeping the vehicle controllable [5].

Traction Control System (TCS)

This system helps the driver when he is accelerating, preventing the wheels from spinning by cutting the engine torque, improving the acceleration performance, and keeping the vehicle stable [5].

Adaptive Cruise Control (ACC)

The ACC maintains the vehicle at the desired speed, reducing the speed if there is a vehicle traveling ahead in order to maintain a safe distance between them.

2.1.2.1 Level 2 Systems

Level 2 systems have the same level of authority as the driver as Level 1. However, they are capable of performing much more capable maneuvers, leading the car in the desired trajectory at the desired speed.

Highway Assist (HA)

This ADAS combines ACC, lane centering system, and BSD to control both the speed and trajectory of the vehicle. The system can also perform an overtake maneuver after the driver turns the indicator on.

Autonomous Obstacle Avoidance

This system, like the HA, controls both the speed and trajectory of the vehicle. In this case, to avoid the collision with an obstacle.

Autonomous Parking

It assists the driver in finding a suitable parking spot and then performing the parking maneuver without any other driver input besides choosing the parking spot.

2.1.2.2 Level 3 Systems

Even if the functions of level 3 systems are similar to level 2 systems, there is a meaningful difference between them. Level 3 system assumes the authority of maneuvers in specific scenarios, and only when they are not capable of handling the situation or when they detect a self fault, do they warn the driver and ask him to take back the control of the vehicle. This means that the driver always has to be ready to take back control of the car. These systems have to provide redundancy of sensors and in the decision-making ECU so they can safely operate in every scenario.

While in level 1-2 systems each sensor host the logic to acquire data, process it, make decisions based on it and command actuators, which means that each function is based on the perception of one or two sensors, in level 3 systems, the sensors mainly have a perception task, and then send the data to the main ECU, which use the data of every sensor available to create a richer and more thorough reconstruction of the environment, and takes decisions based on it and controls the actuators [5].

2.1.2.3 Level 4 Systems

These systems can make decisions and manage an extended number of scenarios when compared with level 3 systems. In those specific scenarios, they perform all of the necessary tasks, so that driver intervention is not required most of the time. This requires a centralized intelligence with all-around sensing. Additionally, these systems should be able to continue working without driver intervention in case of failures [5].

2.1.2.4 Level 5 Systems

This is the final level of automation, where the vehicle is fully autonomous [5], it does not require any intervention from the driver and has such a redundancy, sensing coverage, and decision capabilities that it does not require a steering wheel or pedals. The driver becomes a passenger, where the only user interface needed is the setting of the destination.

2.2 Sensors

As it was seen in section 2.1 to enable the implementation of ADAS with higher-level automation, it is indispensable to have sensors with the capability of providing a large amount of data, with quality suitable

for the target applications.

None of the current sensors has the capability to supply all the information needed by ADAS, as it can be seen in Figure 2.1. Due to this limitation, there is a growing tendency of fusing the data of different sensors [13], which creates a more reliable, robust and complete dataset, where the shortcoming of each sensor's date is compensated by the data of another sensor. Using various sensors ensures that if the data from one of them becomes unreliable or the sensor stops working, there is redundancy of the data being acquired so that the ADAS can still continue functioning with the maximum security possible. The main automotive sensors are explained below.

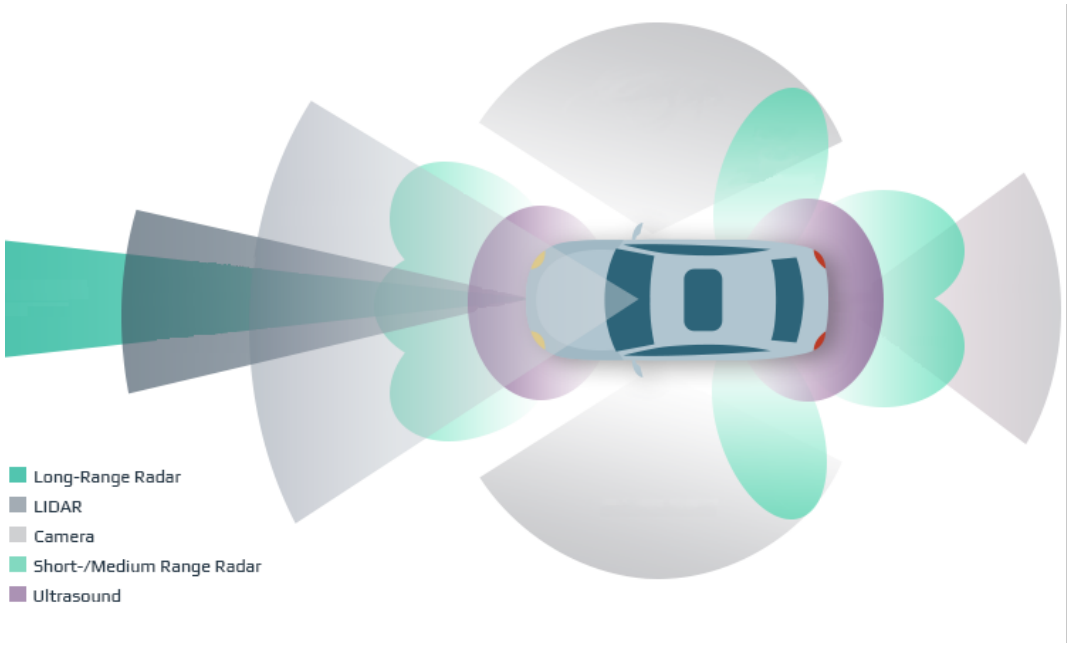


Figure 2.1: Comparison between the action range of the different sensor used in ADAS. Adapted from [14]

2.2.1 Ultrasound sensor

The ultrasound sensor uses the TOF of ultrasonic waves, with the purpose of measuring the distance between an object and the vehicle. This feature is widely used in parking assist and Blind Spot Detection systems. As a result of its intrinsic functional characteristics, this type of sensors is mainly used in a short-range and low-speed application [15].

2.2.2 Radio Detection And Ranging (Radar)

The Radar is based on the transmission of electromagnetic waves, through a transmitter, that reflect off objects and later return to a receiver, giving information about the object's location and speed [16]. It consists of a system divided into two parts. On one side there is the transmitter, which produces electromagnetic waves, in the radio or microwave domain, those waves are then propagated through a transmitting antenna, that is often used as a receiver for later receiving the reflected waves. On the other side, there is a receiving antenna and a receiver connected to a microprocessor, that determines the characteristics of the object [17].

This type of sensor has a long operating range, and since it uses electromagnetic waves, its operation capacity is not affected by the different environmental and weather conditions. Furthermore, it has an excellent reaction time, which is very important when the vehicle is at high speed. On the other hand, it has less precision compared with other sensors. In certain situations, due to reflection or other disturbances, it can detect an object with a wrong size, which can lead to the identification of small objects as very large objects [17].

2.2.3 Video Sensors

The video sensors, are the only sensors that can collect and interpret information about the color of the objects, due to the way they capture data. Therefore, they are considered the best sensors for object classification and texture interpretation [18]. They are the most widespread sensor in the market, which makes them an extremely cheap sensor with multiple options available.

However, the massive amounts of data produced by these types of sensors make the use of complex algorithms, that demands a significant computing power, necessary. This necessity is a hindrance to the utilization of these sensors in ADAS application because the computational power of the systems installed in the automobiles are not powerful enough to ensure that the data is processed fast enough to still be valid in some situations [19].

2.3 Light Imaging Detecton and Ranging (LiDAR)

There is one more sensor used in ADAS, the LiDAR, although, it is still not so widespread as the sensors presented in section 2.2. It is a surveying technology, that uses near infra-red light, to send laser pulses

that reflect in the object and return to the sensor, where they are collected and registered by the receptor. Based on the TOF of each laser pulse, it is possible to obtain the distance of the object where each pulse was reflected. By joining the distance information of all the points, it is possible to create a 3D point cloud of the scenery, as it can be seen in Figure 2.2. Through the use of the point cloud, it is possible to detect the position and the direction of the movement of all the objects that are in the sensors range [20].

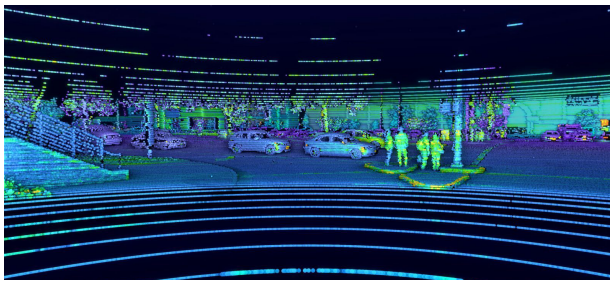


Figure 2.2: Example of a Point Cloud of a scenery. Adapted from [21]

The main drawbacks of the LiDAR sensors are the high production and maintenance cost. Nonetheless, those costs have been diminishing in recent years, mostly because of their ever-growing use and development by the automotive industry. These are high precision sensors with the ability of even detecting small objects, thanks to the wavelength used. Additionally, due to the nature of their functioning, they automatically obtain the distance of the object in each time they read the data. In situations where there is bad weather, the sensing ability of the LiDAR diminishes greatly, becoming an unreliable sensor [22].

2.3.1 Types of LiDAR sensors

There are two different categories of automotive LiDAR sensors, these categories are based on the type of technology in use.

2.3.1.1 Electro-Mechanical LiDAR

These are the first generation of LiDAR sensors for automotive applications. These LiDAR are mechanical spinning systems assembled from various moving parts, which are arranged to produce and emit an array of laser beams towards the targeted area. They are bulky, expensive, and prone to wear and tear in tough terrain. The usually are installed on the top of the vehicle and continuously rotate to scan the surroundings of the vehicle covering a long-range [23].

2.3.1.2 Solid State LiDAR

Solid State LiDARs are regarded as the future of LiDAR in ADAS systems, as they are highly durable, reliable, affordable, and commercially viable. Contrarily to electro-mechanical LiDAR, these are built on a single chip, which integrates all the components of the sensor like the emitter, receiver, and processors. As they are implemented in a single microchip, solid state LiDARs are compact in size, with a low-weight and cost efficient [23].

2.4 Time-to-Digital Converter (TDC)

The operation of a LiDAR sensor is based on its ability to measure the TOF of a laser pulse accurately. Therefore, a circuit capable of performing such measurements is required. The circuit used is usually based on a TDC in combination with other electronic circuits, as it can be seen in Figure 1.2. In this case, two TDCs are combined with an optical circuit to measure the TOF of a laser pulse. Unlike the counters based on the system clock, which have a low maximum resolution, due to limitations connected with power consumption and clock jitter associated with the use of high-frequency clocks [24], the TDC can achieve much higher resolutions using a lower frequency clock.

There are two main platforms used in the implementation of TDC, the Field-Programmable Gate Array (FPGA) which is a reconfigurable system, and the Application-Specific Integrated Circuit (ASIC). Reconfigurable systems refer to the capability of a system to incorporate some extent of customization into the hardware used. It offers a good balance between an efficient implementation and a flexible one. There are several types of reconfigurable hardware besides the ones already referenced above as it can be seen in Figure 2.3, each one has its advantages and disadvantages.

2.4.1 The TDC in ASIC

The ASICs are custom-made integrated circuits, designed for specific applications. Their functionality is determined by a custom mask, which represents the main cost of ASIC development since the production of each individual unit has significantly smaller cost. That makes the use of ASIC, instead of FPGA, profitable, and viable for applications that are already fully developed and tested, and that are produced at a great volume. The crossover point where the ASIC becomes cost-effective is exemplified in Figure 2.4.

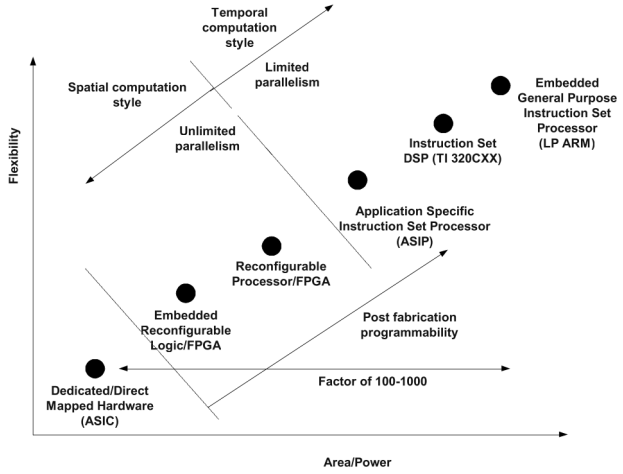


Figure 2.3: Various types of reconfigurable Hardware. Extracted from [25].

The high cost of prototyping and producing a single device, associated with the high production time and low flexibility make ASIC a sub-optimal platform for research and development [26].

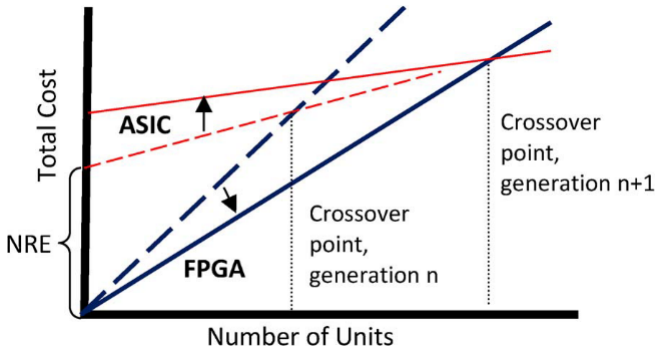


Figure 2.4: FPGA versus ASIC Crossover Point. Extracted from [27]

The ASIC TDCs are implemented with less focus on where will they be applied on and are more focused on which type of TDC architecture should be used to obtain the best TDC possible for specific metrics. A TDC developed in this platform usually has higher linearity, and the same linearity of one developed in FPGA, however, it as a high development cost a long time-to-market.

2.4.2 The TDC in FPGA

The FPGA is based on the use of reconfigurable logic blocks, which are named Configurable Logic Block (CLB) in Xilinx boards, that are constituted by, a Look-up-Table (LUT), an arithmetic operation block called Carry, and optionally by type D Flip-Flop, has it can be seen in Figure 2.5. The LUT offers generic logic blocks, where it is possible to implement whichever type of logic functions needed . The Flip-Flops have

several functions, from being used in state-machines to being used as registers, or even used in pipe-lining operations, or any other application where the system has to work based on the clock [28]. Additionally, FPGA have other elements besides the CLB, some have the purpose of supplying extra memory like the block RAM, available in Xilinx boards, others implement specific advanced arithmetic operations like the DSP, and many other elements that exist in order to connect correctly the signals that propagate within the board.

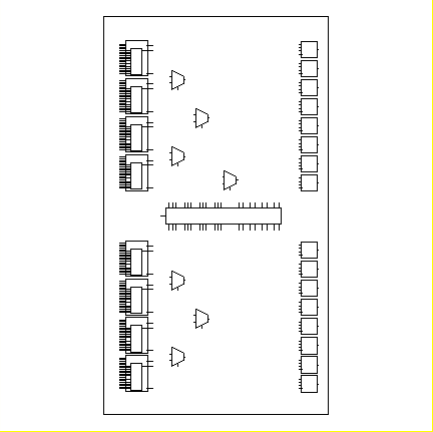


Figure 2.5: Example of a CLB in a Xilinx Board

The improvements that happen in recent years in FPGA have closed the performance level gap between the FPGA and ASIC circuits. The TDC research done in FPGA takes advantage of these performance improvements to enhance the performance of the newly developed TDC. The performance of FPGA based TDC is device dependent, mainly due to restrictions of what type of cells are available, of the technology used in the fabrication process and the available routing resources of each board, which that a high performing TDC architecture in one device could be unusable in another device. This problem arises from non-linearities in the available cells and non-homogeneities in the propagation of the signals due to the routing resources, which are very different from board to board, even in the same model of FPGA. Thus FPGA based TDCs are very dependent on the application where they will be used, meaning that many applications would not be enhanced with their use.

2.4.3 TDC Terminology

In order to tackle the thematic TDC's it is necessary to several fundamental concepts.

The Hit signal: it is the input signal which the time is going to be measured, an example of a Hit is seen in Figure 2.6. In the case of the LiDAR is usually connected to a photo-detector that sets the signal high when the laser pulse is emitted and sets th signal low when the reflected laser pulse is received.

The Start of the signal happens is the rise edge of the Hit signal, the Start event is exemplified in Figure 2.6, the measurement of the Hit begin at the moment the Start occurs. The Stop signal is the fall edge of the signal (see Figure 2.6), and marks the term of the measurement.

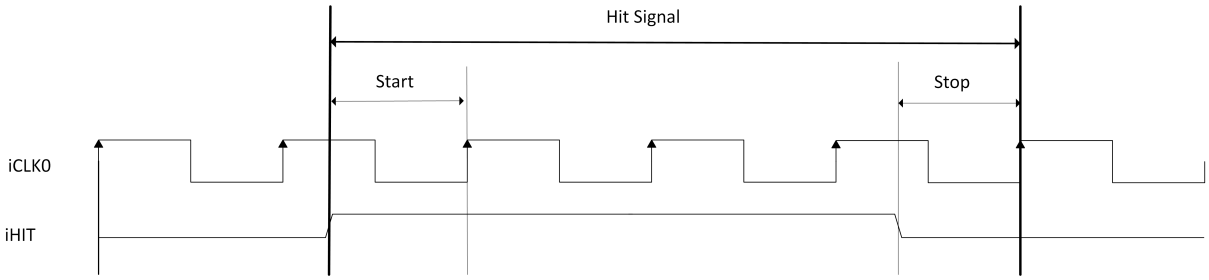


Figure 2.6: Example of Hit Signal, with Start and Stop specified.

The Bins or Taps of a TDC are the points, which usually are Flip-Flop in digital TDC, where the status of the signal is sampled in order to measure the delay line. The Timestamp or Thermometer Code is the code obtained by the gathering of all the of the status of the Taps.

2.4.4 TDC architectures

There are several TDC architectures available, each one with advantages and disadvantages, so it is mandatory to proceed with a careful study of some of this architectures in order to proceed with a well-thought-out choice of which architecture should be used.

2.4.4.1 Analog Time-to-Digital Converter

The most conventional approach to time-to-digital conversion is to first to convert the time interval into a voltage, and later digitize the voltage using an ADC [29]. The start and stop events are used to control the pulse with the width related to the time interval measured. An analog integrator transforms this pulse into a voltage, this works by charging a capacitor with a fix current source. So the quantity of charge stored by the capacitor is proportional to the charge time of the capacitor. That amount of charge is later read by an ADC, as it can be seen in the block diagram in Figure 2.7. There is a trade-off between a long measurement interval and a high-resolution value, as the maximum resolution of the ADC is limited by analog constraints a low resolution and vice versa. A high resolution measurement of long time intervals would require a two

stage approach a coarse quantization of the long time and a fine one for the remainder of the time [29]. Although it is possible to implement TDC with resolution under 50ps, the electronic components vastly increase the area and power consumption of the architecture [30].

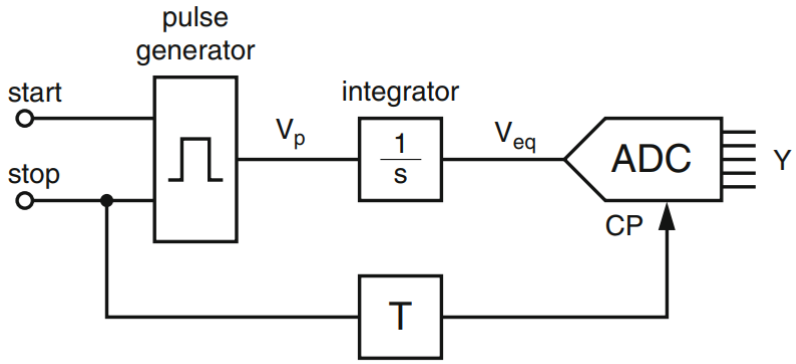


Figure 2.7: Block diagram of a basic analog time-to-digital converter. Adapted of [29].

2.4.4.2 Coarse Counter TDC

The Coarse Counter TDC is the simplest digital TDC where the value of a counter is incremented at every clock cycle. There are two main approaches to this architecture; in one the counter value starts at 0 and only increments when the input signal is high. In the other, the counter is always incrementing, and every time there is a start or a stop event the timestamp is captured, and the time of the hit is measured by the difference between the start and stop timestamp.

2.4.4.3 Pulse Shrinking Delay Line

The idea of a pulse shrinking delay line is that the hit signal starts a ring oscillator that increments a counter every cycle. Therefore the counter value is equivalent to the input pulse-width. The pulse is propagated along a delay-line that contains a deliberate non-homogeneity that causes a reduction of the pulse width at every cycle. Hence as the pulse propagates through the delay-line, it becomes increasingly smaller until it is to narrow to be detected at a certain point, even though it is not zero [29]. This exiting pulse width is recognized as an offset problem, which occurs because of the driving capability limitation of logic gates [31].

The delay-line is then read by flip-flops, that were reset before the measurement to have a logic value of one, after the measurement the one-zero point of the line indicates where the pulse has vanished, using

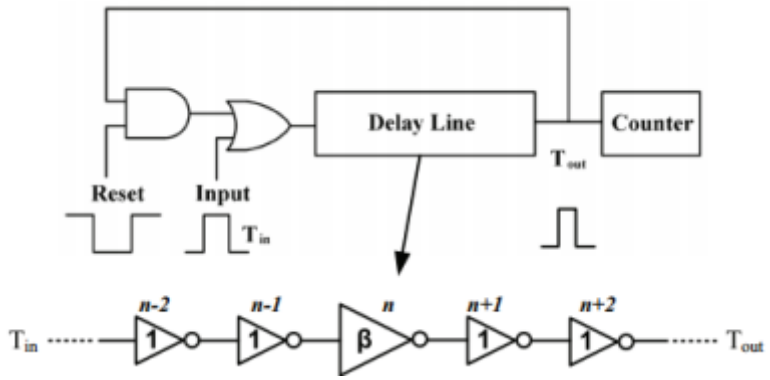


Figure 2.8: Example of the pulse shrinking delay line. Adapted of [32].

this information in combination with the value of the counter it is possible to calculate the time interval of the pulse. In order to reduce the logic consumption of this solution this type of TDC can be implemented in a loop-structure, exemplified in Figure 2.8, where the pulse is propagated in a loop of pulse-shrinking elements, the output of the delay-line is feedback to the input and to AND gate for circular operation. A high-resolution counter is used to count the number of times the input signal is feedback until it vanishes [32].

2.4.4.4 Tapped-Delay Line (TDL)

A Tapped-Delay Line TDC functions by propagating the hit signal through a delay line, which is sampled at every clock cycle. The block diagram of the TDL TDC can be seen in Figure 2.9, in which the hit signal is propagated first by an input stage that detects the changes in the hit signal value. Although it is not mandatory, it simplifies the Time-to-Digital Converter (TDC) system. Afterward, the signal is propagated through the delay line, which is the core of the TDL and sets the resolution and linearity of the system. The delay line then is sampled by the sample line, which stores the delay value in a thermometer code. The code is then decoded to a binary value. Finally, there might be a need for implementing a calibration stage to reduce the non-linearities of the system.

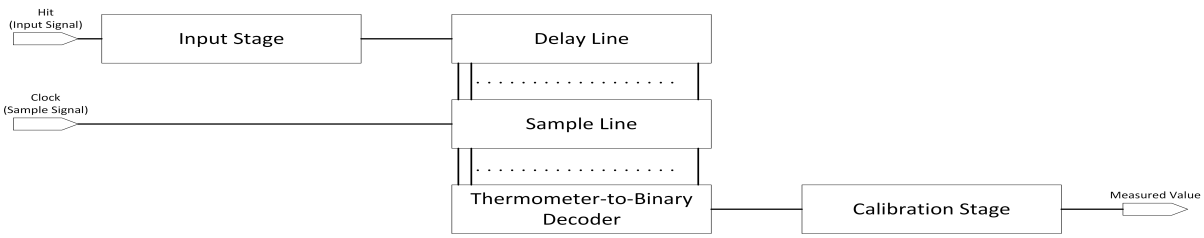


Figure 2.9: Example of the TDL TDC architecture Block Diagram.

The delay line consists of a line of cascading multiple delay elements, which have similar delay time, connected together. The value of the hit can be measured by multiplying the delay of each element by the number of elements that the signal crossed. The delay time of the elements is the maximum resolution possible of the TDC [33].

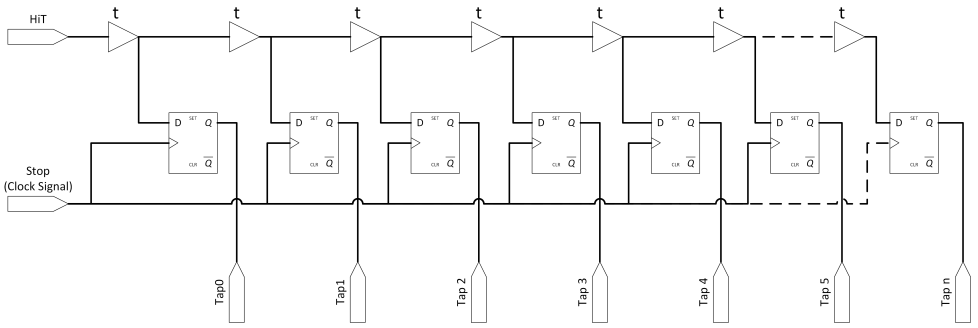


Figure 2.10: Example of the TDL architecture.

2.4.4.5 Ring Oscillator TDC

The basic Ring Oscillator TDC uses a delay line configured into a ring format, where the pulses are propagated iteratively until the conversion is completed [30]. The TDC consists of an odd number of inverters, which are used to generate multiple phases at a high frequency. The phase of the last inverter is into a loop counter which is enabled at the beginning of the Hit and disabled at the stop signal. The loop counter records the total iteration number in the Ring Oscillator during the conversion The sampling logic is triggered by both the start and stop signal, records the status of each delay stage.

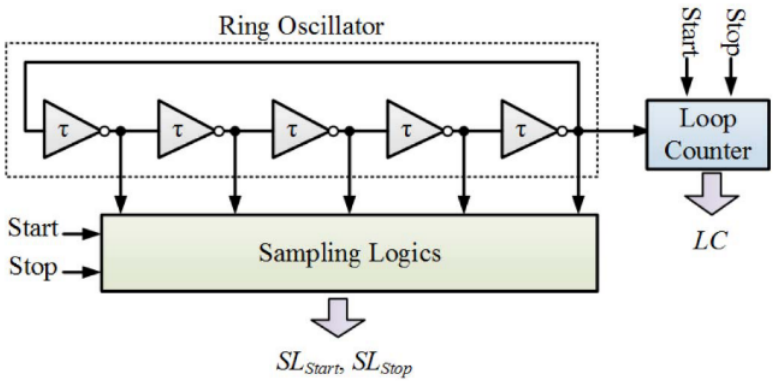


Figure 2.11: Example of the Ring Oscillator architecture. Extracted from [30]

2.4.5 TDC Performance Comparison

To better understand the current scenario of FPGA-based TDC, it is essential to analyze the results from recent TDC implementations. This will help with the identification of drawbacks of the TDC architectures, which could become problematic if not tackled. To do so, a summary of the performance of various TDCs are presented in Table 2.1.

Ref-Year	Device/ Technology	LSB	Precision	DNL	INL	DR	Dead Time	Power	Resource Utilization
Unit		ps	ps	LSB	LSB	ns	ns	mW	logic units
Analog TDC									
[34]-14	CMOS-350	57	30	0.53	0.7	75	-	7.9	0.01mm2
[35]-00	BiCMOS-800	32	-	-	0.16	2500	-	350	11.9mm2
[36]-06	CMOS-350	50	0.19	-	+/-1.1	250	-	0.75	0.23mm2
ASIC TDCs									
[37]-14	350nm	40	2.2	-	+/-0-6	22	1e08	0.0016	0.025mm2
[38]-15	130nm	77	-	-	-	>19.6	330	0.004	-
[39]-14	130nm	5	2.5	[-0.8:0.6]	[-1.2:0.4]	-	-	43	-
[40]-16	180nm	15	20	+/-0.31	+/-0.67	1.28e03	<20	45	0.195mm2
FPGA-based TDCs									
Pulse Shrinking TDC									
[31]-16	Spartan-3AN	-	115	-	+/-1	15	-	-	-
TDL TDCs									
[41]-15	Virtex-6	3	6.5	[-1:3.5]	[-10:6]	-	-	-	-
[33]-16	Kintex-7 Xilinx UltraScale	2.3	3.9	-	-	440	4	-	-
[42]-17	Virtex-5	60	-	[-0.66:0.65]	[-0.54:0.24]	-	-	-	-
[43]-17	Zynq-7000	5	5.8	[0:3.4]	-	50	2.63	-	8832
[44]-15	Virtex-6	23.9	24	[-1:3]	+/-3	>1e09	30	-	-
[45]-18	Virtex-7 Xilinx UltraScale	10.5 5	14.59 7.8	[-0.05:0.08] [-0.12:0.11]	[-0.05:0.08] [-0.15:0.48]	-	-	-	377 19794
81-18	Spartan-6	25.6	37	[-0.9:1.23]	[-043:2.96]	-	8.69	131	415 slices

Table 2.1: TDC Performance Comparison

2.5 Conclusion

This chapter gave an overview of all of the technologies associated with the LiDAR sensor and the other sensors used in ADAS.

It also showed how the various re-configurable Hardware platforms available affect the implementation of the TDC.

Furthermore it gave a summary of different TDC architectures, providing advantages and disadvantages of each one.

So after studying the various TDC and the platforms in which they can be development the choice of developing a TDL based TDC in FPGA was done. The use of a FPGA as the development platform was made due to the flexibility, low-cost and low development time associated with this platform, which lets the focus of this dissertation be the improvement of the TDC architecture and not only its deployment, in opposition to the ASIC platform where the deployment of the TDC would be the primary focus. The choice of deploying a TDC based on a Tapped-Delay Line architecture was made because of its simplicity, as well due to its high resolution and high linearity.

Chapter 3

System Design

After the study of the various technologies used in automotive sensors and a global overview of the ADAS that take advantage of them, in combination with the theoretical insight of the TDC architectures taking into account the different trade-offs of each of them, it is now possible to define how the system will be developed.

Firstly it is essential to specify what are the fundamental blocks that constitute the Acquisition System. After this step, it is necessary to specify how each block interacts with one another. The next step is to determine how the input signal is going to be detected and later define which TDC architecture is going to be implemented. A calibration module was designed to guarantee that the TDC in development can achieve higher resolutions with good linearity. Afterward, the Coarse Counter Block of the system is defined. Then the data storage and data communication systems are specified and designed. Finally, the tests to the TDC system are defined, and the theoretical concepts behind the tests are explained to understand better how the test correctly evaluate the performance of the system.

3.1 Acquisition System Architecture

The goal of this dissertation is to develop an acquisition system to capture the TOF of laser pulses from a LiDAR. Firstly the system must measure the width of the input signal, which is directly correlated with the TOF of the laser pulse, with the highest resolution possible. Next, the measured value has to be calibrated to reduce the measurement errors introduced by the non-linearities of the system. Finally, it has to store the value of the measurements so the PS can later read it via the AXI bus.

The system is divided into two main modules: the TDC_peripheral, which is responsible for measuring, calibrating, and storing the measured value of the input signal. It is constituted by the Input Stage; the TDC and the FIFO; and the AXI_peripheral which is comprised by AXI bus. Various TDC_peripherals can be

connected to the same AXI_peripheral. The TDC_peripheral is designed in a way so it can be connected directly to a processor without the need to use an AXI_peripheral.

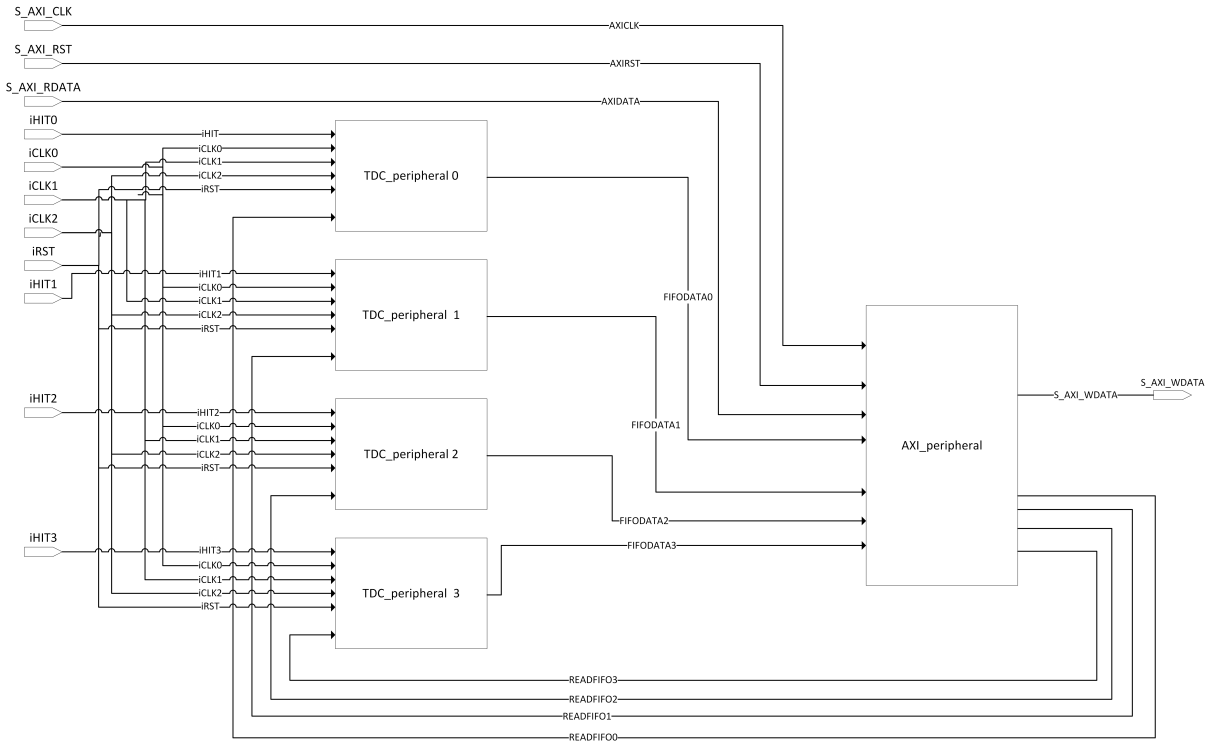


Figure 3.1: Diagram of the Main Block of the System

The Hit signal is first propagated through the Input Stage, as seen in Figure 3.2, it monitors the hit signal and detects the beginning and end of the signal, which is between the signal rise and fall edge. When the signal starts, the Input Stage sends a STORESTART signal to the TDC block, when the signal stops, it sends a STORESTOP signal. The Input Stage has a veto circuit with the purpose of letting only one Hit signal propagate through the TDC block at a time, to prevent the propagation of multiple signals which would make the TDC block unusable, the veto signal does not let a new Hit signal to be propagated until the VALUEREADYOUT is set.

The TDC Block is responsible for measuring the width of the Hit signal. The Hit signal is captured two times, once when the STORESTART signal is received and another when the STORESTOP signal is received. After the signal is stored, the measured value is converted into a binary number, which is then calibrated to mitigate the influence of the non-linearities in the acquisition process. After the calibration is done, the value of the signal is ready to be written into the FIFO Block. This value is available in the NEWDATA bus. When this happens, the NEWVALUE signal is set indicating that a new value is ready, and the VALUEREADYOUT is also set to disable the veto circuit of the Input Stage.

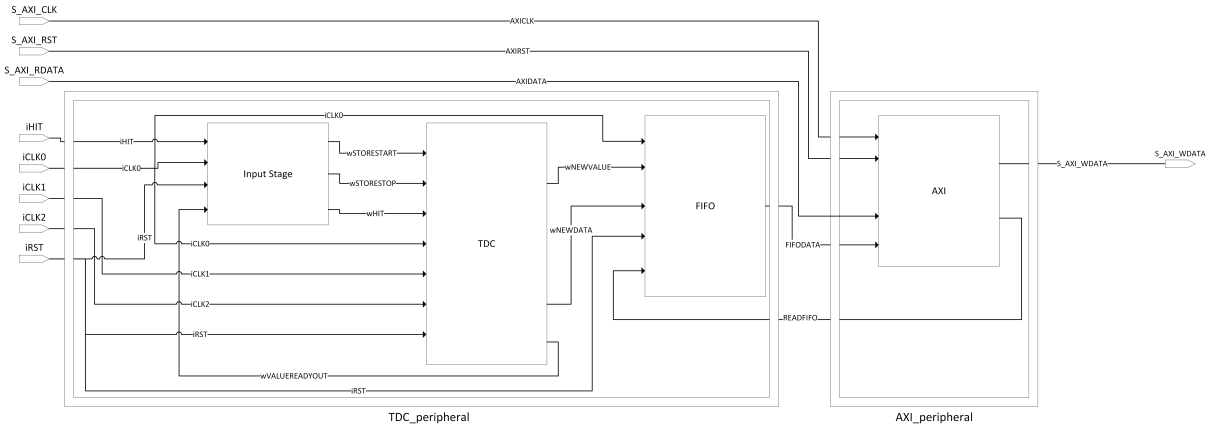


Figure 3.2: Block Diagram of the System

The FIFO block has the role of storing the values acquired by the TDC block. Whenever the NEWVALUE signal is set high, the FIFO stores the value available at the NEWDATA bus. If the FIFO is full, the NEWDATA value is not stored. When the READFIFO signal is set, the oldest value of the FIFO becomes available on the FIFODATA bus, and the value is deleted from the FIFO. Although the FIFO is connected to the AXI bus, that is not necessary as the FIFO is designed in a way that it can be easily connected to any circuit. It can also be directly integrated into a soft-core processor.

As seen above, the AXI bus is not mandatory, yet it was implemented to enable the communication between the acquisition system and the PS of the FPGA. The AXI block receives various commands from the PS and executes actions according to them. Whenever it receives a command to read a value from the FIFO, it enables the READFIFO signal to flag the FIFO block that it wants to read a new value, after reading it, the value is available to be read from the PS.

3.2 Input Stage

The Input Stage is used for two different purposes, monitoring the rise and fall edges of the Hit signal and generating the appropriate control signals to respond to these changes. Furthermore, it also has the purpose of blocking the propagation of multiple Hit signals through the TDC while the value of the current measurement is not yet written into the FIFO.

The edge_detector_ffd0 stores the current value of the Hit signal, and the edge_detector_ffd1 stores the last value of the signal, as seen in Figure 3.3. They are updated at every clock rise edge. The values stored in both Flip-Flops are compared to detect variations of the Hit signal. If the value stored in the first FF is high and the value stored in the second one is low, then a rise edge of the Hit signal has occurred,

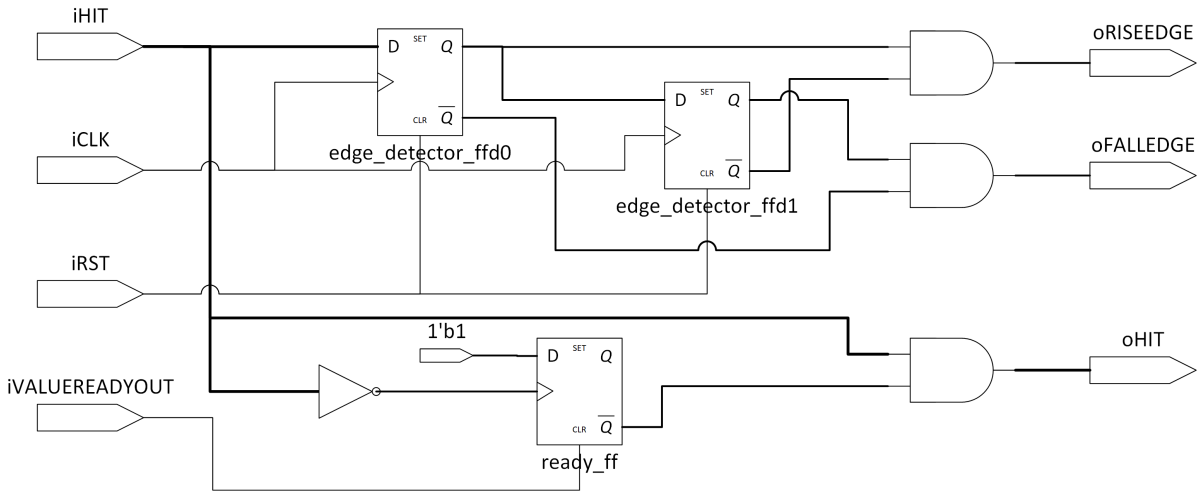


Figure 3.3: TDC Block Diagram

and the oRISEEDGE signal is set at high. Contrarily the detection of a fall edge happens when the value stored in the first FF is low, and the Hit value in the last clock was high. In this case, the oFALLEDGE signal is set to high.

After the occurrence of the fall edge of the Hit signal, the veto circuit becomes active, preventing the propagation of any new hits through the TDC until the iVALUEREADYOUT signal is received, indicating that the processing of the last Hit was complete and that the system is now ready to acquire a new one.

3.3 TDC Design

The TDC is the core block of the Acquisition System; it has the function of measuring the width of the Hit signal. The basic idea behind the hit measurement is to have a coarse counter that runs at system clock rate and a TDL which measures the Hit at a sub-clock period resolution. The measurement of the Hit signal is done by combining these two different measurements, the Fine and Coarse Time. The division between Fine and Coarse Time is explained in Figure 3.4 it can be seen that there are two Fine Time measurements in one Hit signal, the first is the $FineTime_{Start}$, which measures the time between the beginning of the Hit signal and the next rise edge of the system clock, the second one is the $FineTime_{Stop}$, which measures the time between the end of the Hit signal and the rise edge of the next system clock. The resolution of the Fine Time measurements defines the resolution of the Acquisition System. Whilst the Coarse Time measures the number of clock cycles that the Hit signal lasts. The $TotalTime$ is calculated using equation 3.1.

$$TotalTime = CoarseTime + (FineTime_{Start} - FineTime_{Stop}) \tag{3.1}$$

Equation 3.1: $TotalTime$ Calculation.

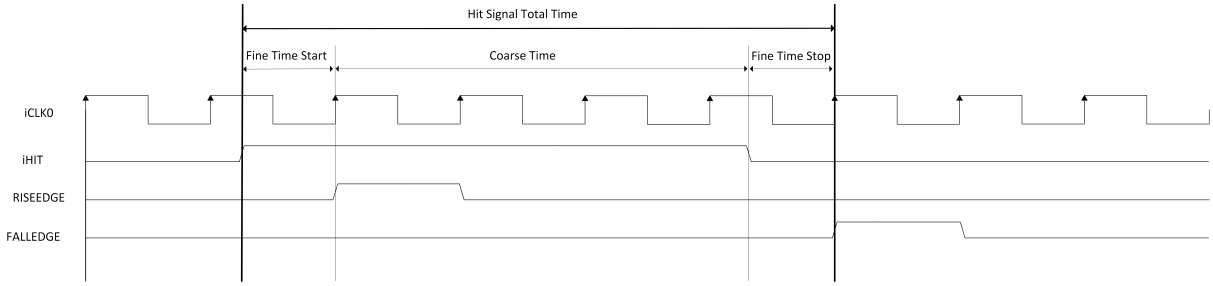


Figure 3.4: Example of the Hit Signal

The TDC has to be able to capture both the Fine and Coarse Time to provide the complete measurement of the Hit. Thus, there is a need for developing two different measurement methods to obtain the $TotalTime$ measurement. In order to measure the Coarse Time, a Coarse Counter connected to the Synchronizer was developed, as seen in Figure 3.5. The Synchronizer is necessary to synchronize the Fine and Coarse measurements. The Coarse Counter starts counting when the Hit signal is set and increments the counted value at every rise edge of the clock. When the signal is reset, the Counter stops incrementing, and when the wSTORESTOP signal is set, the value of the Counter is stored. The Synchronizer guarantees the correctness of the Counter value, reducing the errors connected with problems between the beginning and end of the Hit signal and the rise edge of the clock. These problems and the need for a Synchronizer will be explained in greater detail in Subsection 3.3.4.

In order to implement an Acquisition System that correctly measures the Fine Time, first a TDC architecture must be chosen. In this case, a Tapped-Delay Line architecture was selected. The TDL is the most straightforward TDC architecture to implement; it is implemented by cascading various Carry-logic elements to create a delay line, which is used to interpolate the Fine Time. The Hit signal is propagated through the delay line of the TDL, which is sampled whenever the wSTORESTART and wSTORESTOP signals are set. The sampled states of the delay line are then transformed into a Thermometer code, reflecting the number of elements of the delay line through which the Hit signal propagated before being sampled. This generates two Thermometer codes one for the wSTORESTART signal that corresponds to the $FineTime_{Start}$ and another one for the wSTORESTOP correlated with the $FineTime_{Stop}$. These Thermometer codes are then decoded and transformed into the corresponding binary code by the correspondent Decoder Module, the wBINSTART, and the wBINSTOP codes, as seen in Figure 3.5. Although the Start and the Stop Decoder are analogous, there are differences in the Start and Stop Thermometer

code that create the need for designing two different modules.

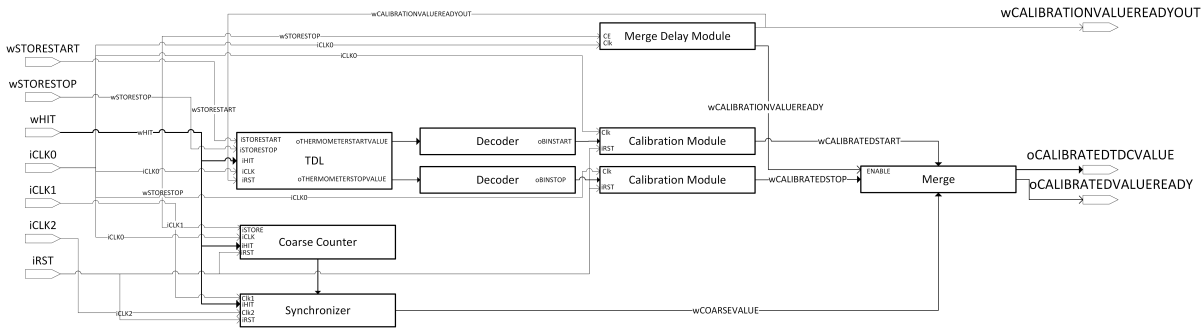


Figure 3.5: TDC Block Diagram

After both binary codes are generated, the values are calibrated by the correspondent Calibration Module. These modules take into account the non-linearities of the measurements done by the TDL and output a calibrated value; with higher linearity than the uncalibrated value however, it has a lower resolution. The calibration process increases the Dead Time of the TDC; this happens due to the time the module takes to output the calibrated value. The TDC is only ready to make a new measurement after the current calibration value is written.The Merge Delay Module delays the merge of the Coarse and Fine Time measurements until the calibrated values are stable and ready to be outputted. When the wCALIBRATIONVALUEREADY is set the Merge Module, joins the wCALIBRATEDSTART, the wCALIBRATEDSTOP, and the wCOARSEVALUE into the oCALIBRATEDTDCVALUE bus and sets the oCALIBRATEDVALUEREADY signal to indicate to the FIFO that there is a new value to be stored. It also sets the VALUEREADYOUT signal so the Input Stage veto circuit is disabled and the TDC can acquire a new Hit signal.

3.3.1 Tapped-Delay Line (TDL)

The basic principle of the TDL is to measure the Fine Time of the Hit signal by propagating it through a delay line so the system can sample it. The delay line is composed of multiple cascading Carry-logic elements that theoretically have the same delay time between them. A bank of type D Flip-Flop registers the status of the line. The delay line and the FFs form the TDL (see Figure 3.7). Specifically, in this case, the element used is the Carry8 element; the Hit signal is propagated through the element as seen in Figure 3.6. The propagation of the Hit signal through the element can be sampled in the CO pins of the Carry8. Multiple Carrys are connected with each other to create the delay line, and this is done by connecting the last CO pin of the current Carry to the CIN of the next element. The sum of the delays of all elements of the line has to be larger than a complete clock cycle The TDL architecture was chosen due to its high time

resolution and a small measurement dead time compared with other architectures [46], as seen in Table 2.1.

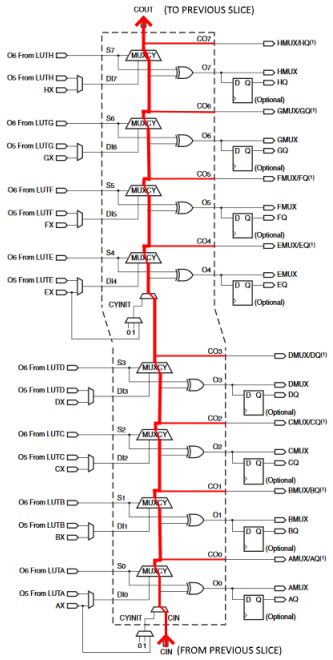


Figure 3.6: Carry8 Cell Signal Propagation

Whenever a new Hit signal arrives, the signal propagates along the TDL and the FFs sample the status of the cells at the system clock rate (an example of this can be seen in Figure 3.7). In the TDL-TDC, the equivalent cell delay and their uniformity determine the achievable resolution. There are two main contributions to the equivalent cell delay, the physical delay of the Carry-logic, and the time skew of the system clock signal routing to each FF. As the FPGA has the logic resources divided into predetermined blocks, such as the slices in Xilinx FPGAs, and a large number of slices constitutes the TDL, the physical delay of the cells may differ between the same slice or between several slices of the Carry-logic. The clock signal that is connected to the FFs that sample the delay line come from a global buffer and a clock distribution network, this makes the time skew of the clock signal at the Flip-Flop something that cannot be neglected and is another factor that creates non-linearities in the measurement done using the delay line.

One method to reduce the non-linearities of the delay line is to connect additional elements in parallel with the outputs of the Carry-logic elements to increase the time-constant of the signal, which increases the delay value of the signal, making it possible to detect elements with a minimal delay delay, that previously were undetected. This technique was first proposed in [47]. In this dissertation, the elements used to

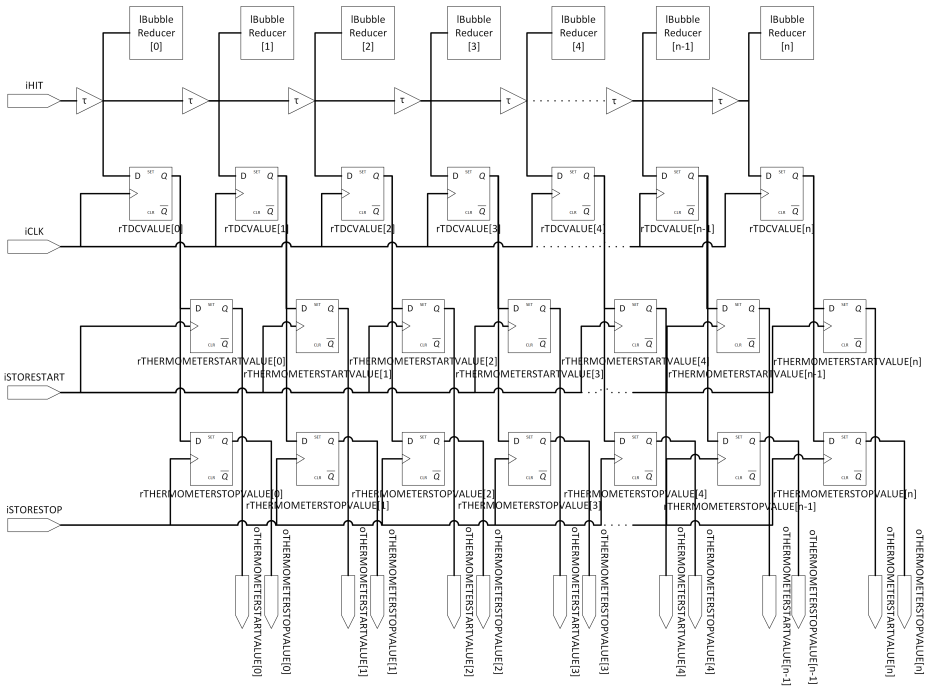
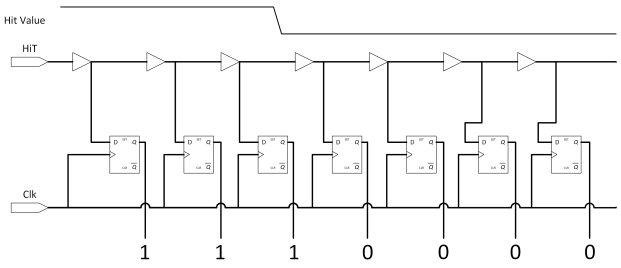
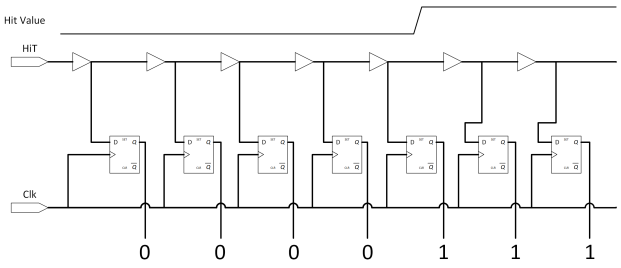


Figure 3.7: Tapped-Delay Line Architecture Example

increase the time-constant is the LUT. It is connected in parallel to the FF in the CO pin of the Carry-logic elements.



(a) Value at the Rise edge of the Hit signal



(b) Value at the Fall edge of the Hit signal

Figure 3.8: Example of the Thermometer Code exemplifying a Rise and a Fall edge of the Hit signal

The status of the delay line captured at the Start and Stop of the Hit is called the Thermometer Code.

It is stored on the rTHERMOMETERSTARTVALUE FF for the rising edge, and on the rTHERMOMETERSTOP-VALUE FF for the falling edge. The output of this code and its relation to the delay line is presented in Figure 3.8.

3.3.2 Decoder Module

The Decoder is responsible for reading the Thermometer Code of the TDL and convert it into a binary value corresponding to the last position crossed by the rise or fall edge of the Hit signal. The TDC uses two different designs of the same Decoder, one for the $FineTime_{Start}$, where the Decoder looks for the last propagated high value, and one for the $FineTime_{Stop}$, where the Decoder looks for the last propagated low value.

The Decoder is based on the priority encoder circuit (see Figure 3.9), which is used to compress multiple binary inputs into a smaller number of outputs.

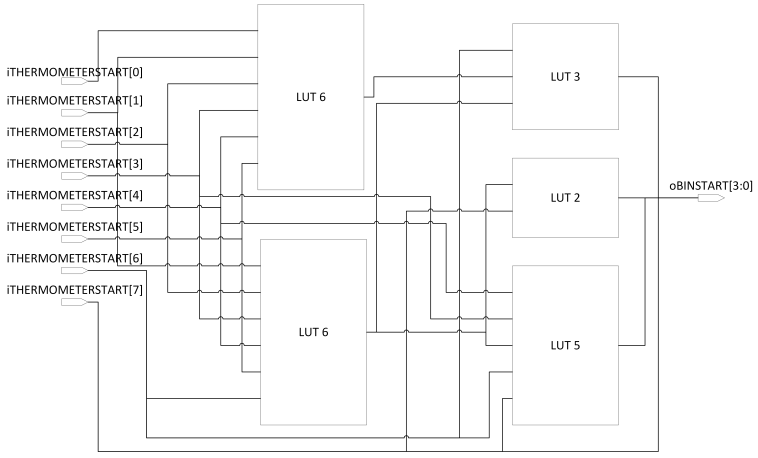


Figure 3.9: Priority Encoder Example for 8 inputs

3.3.3 Calibration Module

In TDL based TDCs implemented on FPGA, and more prominently in the ones implemented on the latest technology FPGAs, there is a substantial nonlinearity problem in the delay time of each cell, as seen in Figure 3.10. This problem has several causes from the effect of the PVT specific to each cell of the delay line, to the time skew associated with the routing of the system clocks connected to each Flip-Flop. It considerably degrades the performance of a TDC, as each cell contributes with a different width for the total delay of the delay line, which in turn makes the results measured unreliable.

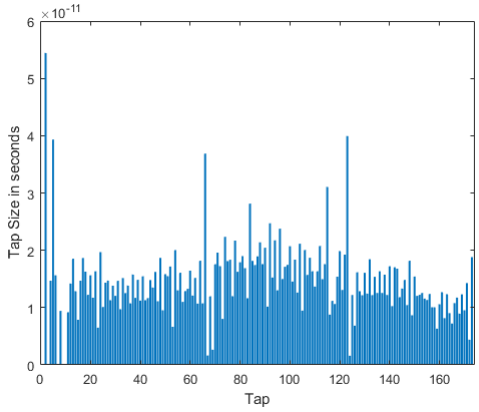


Figure 3.10: Size differences between various taps of the same delay line.

If the nonlinearities of the delay line become too prominent, a new problem emerges the zero-width bins [46]. As a result of different clock skew of the Flip-Flops that sample the delay line, some taps might be registered ahead of time, which will reduce the width of the previous tap, an example of this phenomenon is given in Figure 3.11. The previous tap can have a zero-width delay if the amount of time the tap is registered ahead is equal to the physical delay of the current delay element. If the amount of time ahead is greater than the physical delay of the element, the tap always sees the Hit signal before the previous tap, creating a negative tap [46]. Thus it is essential to develop and implement some type of calibration that can enhance the linearity of the delay cell.

3.3.3.1 Calibration Strategies

There are two main calibration strategies that can be implemented. One is the online calibration, which is when the delay lines are analyzed in real-time, in order to characterize the nonlinearity, by the system where the TDC is implemented. This can be done either by software, where the software application interprets the results and returns values that take in account the nonlinearity, or directly in hardware where the results of the TDC are stored and later interpreted to calibrate the results. The second is called offline calibration, where a high number of measurements is stored in a computer interpreted using a specially developed software that creates a nonlinearity table, that is later embedded in the TDC system to calibrate the results.

This second strategy has a significant disadvantage; it obligates that for each TDC channel, there has to be an offline analysis and then the construction of a calibration table, which increments the necessary development time of the TDC. Additionally, if there is a change in the nonlinearity of the delay line during the

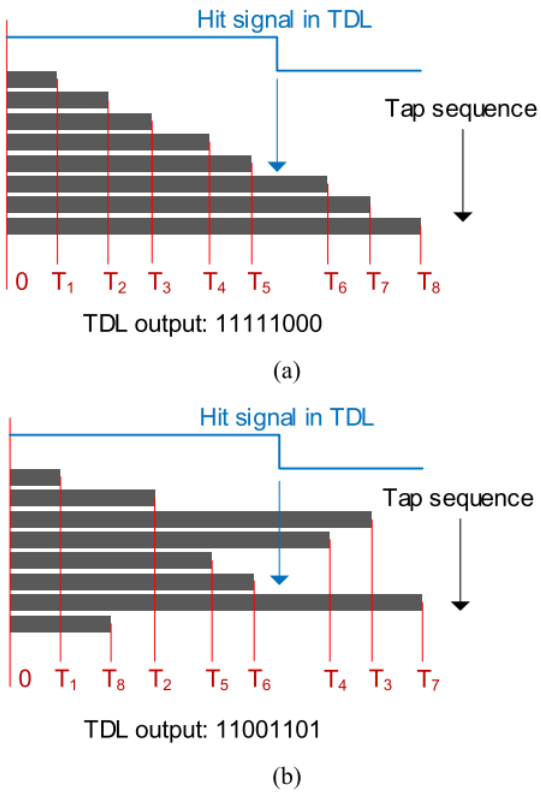


Figure 3.11: Zero-width Tap Capture. For a Hit signal, the outputs of the TDL where (a) the sequence of tap is consistent with the real order of the delay and (b) tap sequence is not consistent creating a zero-width tap on the Thermometer code. Adapted of [48].

functioning or over time, the calibration becomes unreliable contrarily the online calibration is automatic, not requiring any user input, keeping the TDC calibrated over time.

Furthermore, there are two different approaches of how to apply calibration to a TDC channel. First, there is the bin-by-bin calibration, which can be done both online and offline it is based on the use of the Code Density test, which is used to calculate the nonlinearity of the delay line. After the characterization of the delay line, it is possible to calibrate the values measured by the TDC. This measurement already weights in the different widths of each delay cell.

Therefore a Code Density test is performed in the delay line, with the purpose of calculating the real delay width of each cell after the end of the test, the real width of each cell is integrated in order to create a new delay line where the different sizes of the delay cells is reflected in the measurement, has it can be seen in Figure 3.12. The main shortcoming of this approach arises if a significant number of zero-width taps exist in the delay line; this damages the performance of the bin-by-bin calibration. There are some techniques to mitigate the zero-width bins in the delay line, like the use of bin realignment, which involves analyzing the delay line and swapping the order of the taps, so the new order of the taps coincides with the

original order in which the taps are captured in the delay line. The use of this technique involves an offline analysis of the delay line, which has the same disadvantages of the use of offline calibration strategies [48].

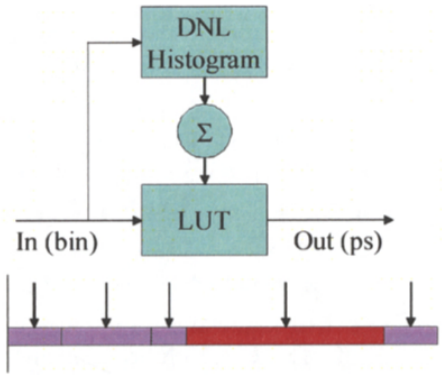


Figure 3.12: Functional block of an online Bin-By-Bin calibration. Adapted of [49].

The second approach is the use of bin decimation calibration. It is an effective solution where there is a trade-off between greater linearity of the delay line and a small loss in measurement resolution. The idea of bin decimation is re-organizing the current taps into several groups and create a new delay line constituted by this new tap groups which are sampled instead of the original delay line. The goal of this technique is to create a new delay line with a more evenly distributed cell width, substituting the use of bin-by-bin calibration [46].

To implement this calibration, the number of ideal bins that will be used (N) to interpolate the system clock (T_{clock}) is defined. Hence the ideal width can be calculated as T_{clock}/N. The position of each ideal bin is depicted by iB_k, in Figure 3.13.

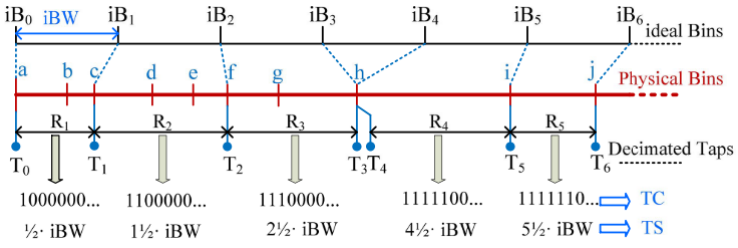


Figure 3.13: Explanation of the Bin Decimation Calibration. Extraction of [50].

3.3.3.2 Calibration Module Design

So to correctly calibrate both the $FineTime_{Start}$ and the $FineTime_{Stop}$ measurements, a Decimation Calibration based Module was designed. The Calibration Module is divided into three blocks, as seen in

Figure 3.14. First, there is the Dual-Port RAM, which is divided into two equal sections. One section, the Acquisition Section, stores the values of the Hits so it can later be used in the generation of the Calibration Table. The other section, the Consultation Section, stores the Calibration Table after it was generated, so that it can be consulted, returning the calibrated tap values. The Calibration Table stores the equivalent tap value of the new Decimated delay line, for each tap of the original delay line. The two sections of the RAM are interchangeable hence the section that was used for the Acquisition Section afterward will be used for the Calibration Table, this switch is done by changing the most significant bit of the two registers that point to the first position of each section. The rRAMposition point to the Acquisition Section and the rCONSULTATIONposition points to the Consultation Section.

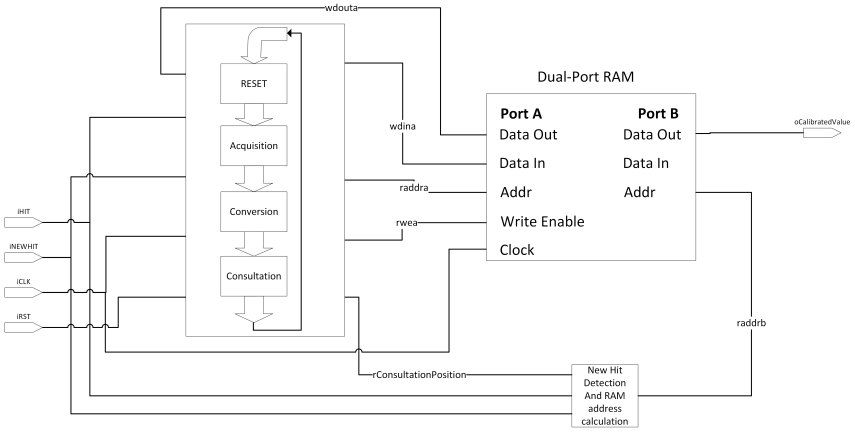


Figure 3.14: Calibration Module Diagram.

The second block, the Calibration Generator, is responsible for the acquisition and calculation tasks it is connected to the Dual-Port RAM. The behavior of this block is explained in the State Machine presented in Figure 3.15. The block starts at the RST state where it first cleans all the values of the RAM that is not being used for consultation. This is necessary, so there are no errors in the storing of the new Hits.

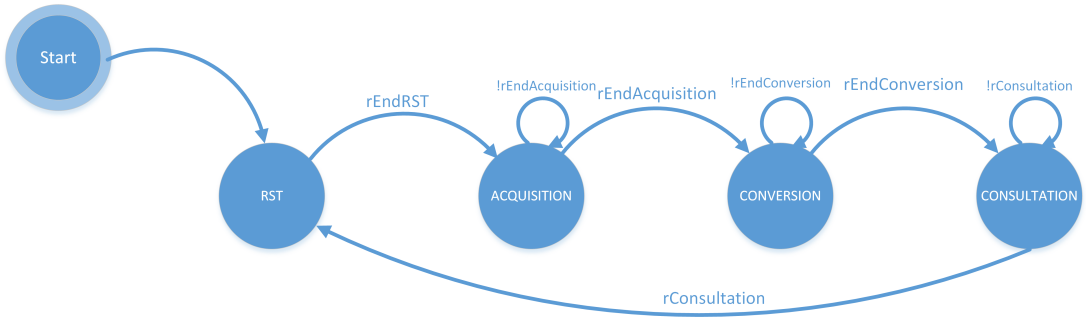


Figure 3.15: Calibration Module State Machine.

After all the values of the RAM are cleaned, the rEndRST flag is set, and the ACQUISITION state begins. In this state, the module acquires and stores the binary value outputted by the decoder module. Whenever the iNEWHIT signal is set high, it indicates that a new Hit value was received. When this happens, the value stored in the Acquisition Section of the RAM that corresponds to the position pointed by the binary value of the Hit is incremented by one indicating that another Hit with that corresponding value was received. This state continues until the necessary number of Hits is acquired and stored in the RAM the number of hits stored has to be high enough to correctly reflect the real width of each cell in the delay line, using the same principle than the Code Density test, which is explained later in section 3.6. Once the necessary number of Hits was acquired, the rEndAcquisition flag is set, and the CONVERSION state starts.

In this state, the Calibration Table is generated the generation process uses the real width of the cells of the delay line to create a new Decimated delay line in which its elements all have the same ideal size. The size of the new elements has to be previously defined. The number of taps (N) used to interpolate the system clock (T_{clock}) give the value of the ideal size of the new taps, which is calculated as T_{clock}/N. To create the Decimated line, the algorithm, showed in Algorithm 1, was designed. This algorithm starts at the beginning of the delay line, and it sums the real width of each cell into a sum variable. Every time a new width value is read and summed the variable is tested whether its value exceeds the size of the ideal tap if it exceeds, then the $currentTap$ value is incremented by one, and the value of the sum is decremented by the size of an ideal tap. Afterward, the value in the RAM position that stored the real width of the current cell is substituted by the value present in the $currentTap$. This variable contains the value of the ideal tap of the Decimated line that the current tap belongs too. This process is repeated from the first to the last cell of the original delay line, creating the new Calibration Table. Finally, after the end of this process, the rEndConversion flag is set, and the CONSULTATION state starts.

Algorithm 1 Calibration Table Generation Algorithm

```
1:  sum ← 0
2:  currentTap ← 0
3:  for i ← 0 to NTaps do
4:      sum ← sum + dataA[i + rRAMPOSITION]
5:      if sum > DECIMATED_HITS then
6:          currentTap ← currentTap + 1
7:          sum ← sum − DECIMATED_HITS
8:      end if
9:      dataA[i + rRAMPOSITION] ← currentTap
10: end for
```

In the CONSULTATION state the two sections of the RAM, the Acquisition Section, that now holds the

newly constructed Calibration Table, and Consultation Section, that holds the old Calibration Table switch with each other. This process is done in one clock cycle, guaranteeing that the Calibration Module is always functional and does not have any dead time due to section switching step of the calibration. When this happens, the rConsultation flag is set, and the state machine goes back to the RST state and repeats the process explained.

The third block is the Consultation Manager. After the first Calibration Table is generated, at every new Hit the value of raddrb is set to the value of the Hit so that the calibrated value of the Hit is outputted in oCalibrationValue.

3.3.4 Coarse Counter

The Coarse Counter measures the Coarse Time of the Hit. The value of the counter is incremented at every clock cycle through the duration of the Hit. Its value is stored when the iSTORE signal becomes high. The diagram of the Counter can be seen in Figure 2.1. In this dissertation, the Coarse Counter and Synchronizer used are based on the ones proposed in [51].

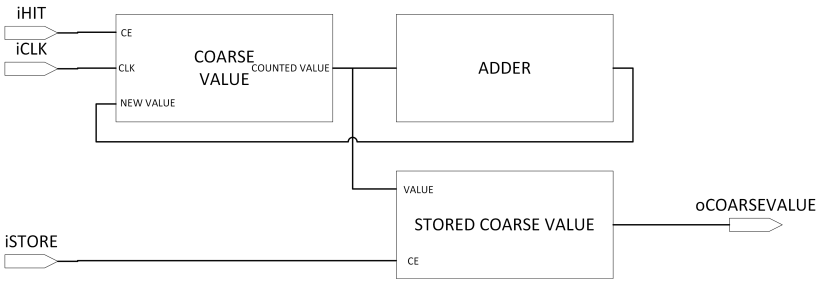


Figure 3.16: Coarse Counter Diagram.

The TDC is prone to have a synchronization error between the Coarse and Fine measurements. These errors affect the measured values by one clock cycle. These types of errors occur whenever the rise or fall edge of the Hit signal arise near the rising edge of the clock. Thus the necessity of implementing a Synchronizer to minimize the occurrence of these errors arises, (see in Figure 3.17) the Synchronizer is based on the use of two Coarse Counter connected to two different phased clocks as arbitrators. In the case of the input signal arriving at the same time as the main Coarse Counter clock, the main Counter may or may not count the clock, and the value sampled at the TDL will be zero. In this case, the first phased Counter is used to obtain the correct counting value. This scenario is exemplified in Figure 3.18 in the Hit1 case. In another situation, the rising edge of the Hit may arrive at the same position as in the previous case, but additionally, the falling edge arrives at a point between the rise edge of the main

clock and the fall edge of the second phased clock. This may cause a counting error in the first phased Counter, and therefore a second phased Counter is needed. With these three Coarse Counter, the main one plus the two phased ones, every possible counting error conditions can be identified. In Figure 3.18, every error condition is illustrated.

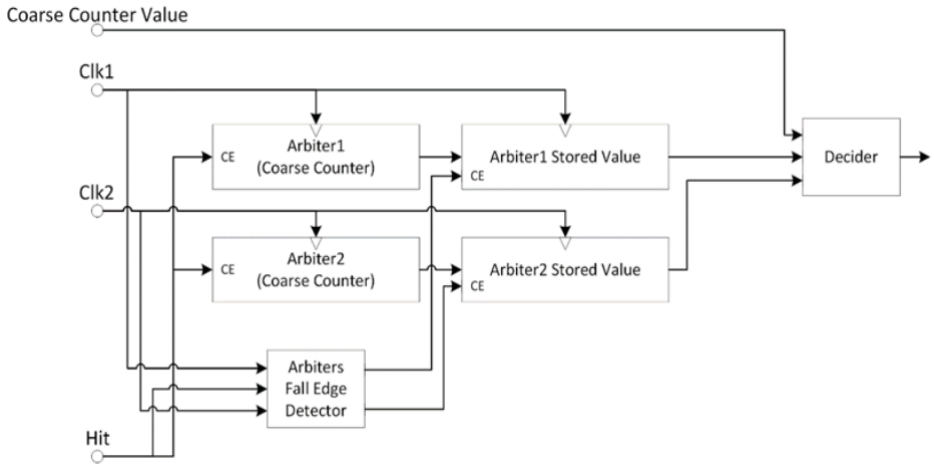


Figure 3.17: Synchronizer Block Diagram. Adapted from [51]

The Clk0 in Figure 3.18 is the system clock and is responsible for sampling the delay line and for incrementing the main Counter. In a scenario where the Hit signal does not occur near the Clk0 rise edge, the Hit2 scenario of Figure 3.18, the value of the main Coarse Counter suffers no metastability and the value captured by it can be directly used. In the other scenarios, metastability might happen, making the value of the main Coarse Counter ambiguous. When this happens the first phased Counter is used, although in some conditions this Counter might also have inaccurate values, due to the other edge of the input signal occurring between the rise edges of the phased clocks, so the second phased Counter is used to determine the actual value of the first phased clock. In the case of the Hit3, both phased clocks have the same value, but the value sampled by the delay line will be lower than the phase difference between clocks, meaning that the stop of the Hit signal occurs after the rise edge of Clk0 but before the rise edge of Clk1. When this happens, the value of the first phased Counter needs to be incremented by one before comparing with the main coarse Counter. Lastly, if the value of the two phased Counters is different, like in the Hit4 scenario, where the stop signal arrives between the rise edge of Clk1 and Clk2, the second Counter has the correct value and can directly be compared to the main Counter.

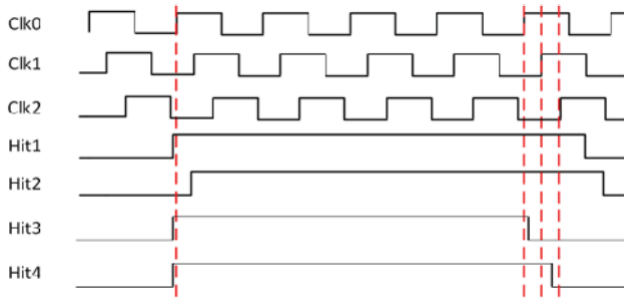


Figure 3.18: Metastability measurement scenarios. Adapted from [51]

3.4 FIFO Module

The FIFO is used to store the measurements done by the TDC. It was implemented in a way that both the write and read parts are completely independent of each other and can function at different clock frequencies.

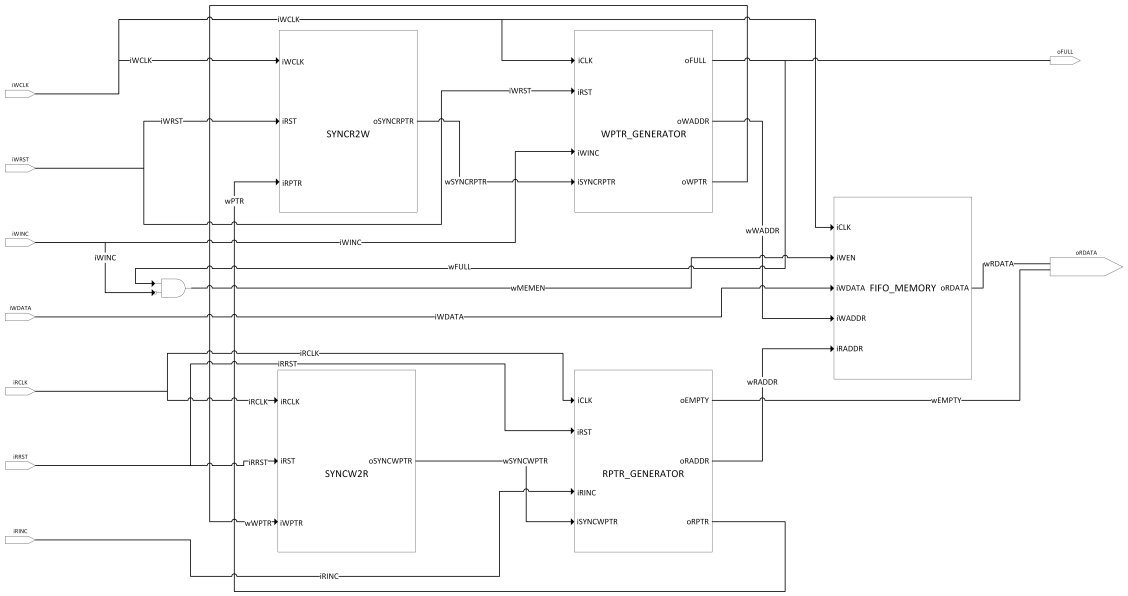


Figure 3.19: FIFO Module Diagram.

The FIFO functions by using two different pointers, one for pointing to the reading position and another pointing to the write position. The addresses of these two pointers are generated by the WRPTR_GENERATOR and the RPTR_GENERATOR, respectively (Figure 3.19). Since these two pointer generators work at different clock rates, some kind of synchronization between them is needed. For that purpose, there are two synchronization blocks which ensure that both pointers work correctly with one another.

Whenever the iWINC signal is set, if the wFULL signal is low, the value present in iWDATA is stored in the FIFO. The WRPTR_GENERATOR generates the wFULL signal, and it is set whenever the FIFO is full.

Whereas when the iRINC is set, if the wEMPTY signal is high the read pointer is not incremented, and the previous value merged with the wEMPTY is outputted in the oRDATA. This merge is done to indicate to the software reading the value that no new value was available and that the read value is invalid and should be discarded. Contrarily if the wEMPTY value is low, the read pointer is incremented, and a new value is available at oRDATA. The FIFO_MEMORY outputs the value of the position pointed by the read pointer, and it only stores the value available in the iWDATA at the address pointed by the write pointer when the iWEN signal is set.

3.5 AXI Module

The Advanced eXtensible Interface (AXI) is a protocol part of the Arm Advanced Microcontroller Bus Architecture (AMBA) specification. It is a point to point interconnect designed to have high performance and high speed for microcontroller systems. The protocol is designed to avoid bus sharing, which allows a higher bandwidth and lower latency. The protocol provides a set of rules for how the different modules on a chip communicate with one another. It requires a handshake-like procedure before all transmissions that allow different components to talk to each other without interfering with one another. The protocol allows burst type communication for continuous transfer of data. The AXI protocol was chosen due to its flexibility.

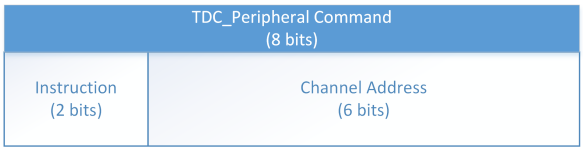


Figure 3.20: TDC_Peripheral Command format.

The AXI Module was developed with the intent of allowing multiple TDC_Peripherals connected to it. In order to enable multiple TDC_Peripherals, a set of commands passed via the AXI bus were created and implemented. Whenever the AXI Module receives a new command, the type of command is assessed, and the action that corresponds to is executed. The commands have a size of 8 bits, where the 2 two most significant bits are used to indicate the command, and the six least significant bits are used for passing the address of the TDC_Peripherals, where the command will be executed. The format of the commands is shown in Figure 3.20. There are four different commands, the READ_CHANNEL, the READ_ALL, the RST, and the NOP command. The READ_CHANNEL is used for reading a new value of the FIFO of the specified channel in the address part of the command. The READ_ALL command reads a new value

from the FIFO of every existing channel. The RST command is used to reset all the channels. When this happens, all the values stored in their FIFO are cleared. This command should always be executed before the first use of the TDC_Peripherals. The NOP has the objective of clearing the registers used to control the TDC_Peripherals.

The channel which is going to be read is indicated by the axi_araddr register which is then read address register of the AXI protocol. The reading operation does not update the values of the FIFOs , meaning that the READ_CHANNEL or the READ_ALL commands are always needed. The AXI Module supports burst transfers, which means that the values of all the TDC_Peripherals can be read in a single AXI transfer.

3.6 System Test

To correctly access the performance of the designed TDC, several metrics and tests have to be defined. In this section, those tests and metrics are explained and defined.

3.6.1 Code Density Test

The Code Density test is used for measuring the real width of the taps of the delay line. This is done by connecting the input of the TDC to an oscillator or wave generator these instruments are configured to output a square wave that will generate the Hit signals for the TDC. The frequency of this wave must be non-correlated with the TDC system clock. Accordingly, there is an equal probability of the Hit signal being in any position of the clock period, meaning that the number of times the Hit signal is detected in each tap of the TDC reflects the real tap width. This also allows for detecting zero-width taps [33]. A vast number of Hits has to be measured and recorded to reflect the real width of the delay line correctly. After all the Hits were recorded the equation 3.2 is used to calculate the real width of the taps.

$$Wi = Ni * \frac{T}{N} \tag{3.2}$$

Equation 3.2: Code Density Test Calculation.

The Wi is the ith tap width, Ni is the number of times that the tap was recorded, N is the total number of measurements done and T is the system clock period [51].

3.6.2 Differential nonlinearity (DNL) and Integral nonlinearity (INL)

Both the DNL and the INL use the results of the Code Density Test to measure the non-linearities of the delay line. The DNL is defined as the deviation of a single tap width form its theoretical value [46], while the INL represents how large can the measurement error be in a single measurement [52]. The DNL is calculated using equation 3.3, where Wi is the calculated width of the ith tap, and the Wid is the theoretical width of the tap [53].

$$DNL = \frac{Wi - Wid}{Wid} \tag{3.3}$$

Equation 3.3: DNL Calculation.

The INL is calculated by the equation 3.4.

$$INL = \sum_{i=0}^{N-1}(Wi - Wid) \tag{3.4}$$

Equation 3.4: INL Calculation.

The value of the DNL and INL are normalized to one Least Significant Bit (LSB), which is the shortest time that the TDC can measure. In the case of the TDL-TDC, this is the ideal width of a tap. The normalized results are obtained by dividing the results from equation 3.3 and 3.4 by the Wid.

3.6.3 Precision and Resolution Test

The Resolution in a TDC also referred to as the LSB, is the minimum time step that can be distinguished. The TDC precision, also called standard deviation, is described as how far from the expected value can the measurement be. It can be calculated by equation 3.5, where σ_q is the quantization error, obtained by $LSB/\sqrt{6}$, σ_{DNL}, and σ_{INL} are the standard deviation of the TDC DNL and INL respectively. σ_{CLK} is the jitter of the system clock, and σ_{extra} are external sources of jitter.

$$\sigma_{TDCrms} = \sqrt{\sigma_q^2 + \sigma_{INL}^2 + \sigma_{DNL}^2 + \sigma_{CLK}^2 + \sigma_{extra}^2} \tag{3.5}$$

Equation 3.5: Precision Calculation.

3.7 Conclusion

In this chapter, the fundamental blocks of the Acquisition System were defined, and their interactions determined. Afterward, the TDC block was planned and each of its parts designed, with particular attention to the design of the TDL and the Calibration Module, as they are the modules that define the resolution and linearity of the system. Then the data communication were designed in a way that the Acquisition System can easily be connected to another AXI peripheral or embedded directly in a processor.

Chapter 4

Implementation

After defining the fundamental blocks of the Acquisition System and designing the different blocks of the system, it was possible to proceed with its implementation. The objective is to implement a sixteen channel Acquisition System.

Thus, this chapter provides an overview of the implementation of the system. For each of the modules defined in the previous chapter, there is a section where it its explained how they were implemented and what challenges arisen in the implementation process.

Furthermore, in order to ensure an optimal distribution of the sixteen channels of the TDC acquisition system, it was necessary to place some of the system's modules manually. Therefore there is a section about the floorplanning of some of the modules of the Acquisition System.

4.1 Input Stage Implementation

The implementation of the Input Stage is divided into two sections that reflect the functionalities of the block, the detection of the Hit rise and fall, and the Hit veto circuit.

As seen in section 3.2, the rise and fall edge detection uses two Flip-Flops to sample the status of the Hit signal. The input of the edge_detector_ffd0 is connected to the Hit signal (Line 6 of Listing 4.1), and its output is connected to the input of the edge_detector_ffd1 (Line 2 and 9 of Listing 4.1). The outputs of both Flip-Flops are connected to the wEDGE array, to detect the edges of the signal. Both positions of the wEDGE are assessed, as seen in Line 15 for rise edge and Line 16 for the fall edge.

In order to guarantee that the implementation of the Input Stage respects the circuit of the design and utilizes the minimum hardware possible, several specific steps were taken. First, both edge_detector Flip-Flops were explicitly declared, to guarantee that the FDCE component was used, this specific Flip-Flop has an asynchronous clear which is essential for the operation of the circuit. Additionally, the synthesis

attribute dont_touch was used (Line 1 and 8 of Listing 4.1) to prevent the optimization or absorption into other logic blocks of a specific part of the design, it also prevents optimizations along the hierarchy boundaries, and it also affects the place and route to prevent logic optimization. This attribute is essential as it is the only way to guarantee that the design tools do not change any of the necessary logic while trying to optimize the circuit, which could render the circuit useless.

```
1  (* dont_touch = "TRUE" *) FDCE #(.INIT(1'b0)) edge_detector_ffd0(
2     .Q(wEDGE[0]),
3     .C(iCLK0),
4     .CE(1'b1),
5     .CLR(iRST),
6     .D(wHIT)
7  );
8  (* dont_touch = "TRUE" *) FDCE #(.INIT(1'b0)) edge_detector_ffd1(
9     .Q(wEDGE[1]),
10    .C(iCLK0),
11    .CE(1'b1),
12    .CLR(iRST),
13    .D(wEDGE[0])
14 );
15 (* dont_touch = "TRUE" *)assign wRISEEDGE = wEDGE[0] & ~wEDGE[1];
16 (* dont_touch = "TRUE" *)assign wFALLEDGE = wEDGE[1] & ~wEDGE[0];
```

Listing 4.1: Input Stage Hit Detection

The veto circuit has the function of ensuring that only one Hit signal is propagated through the TDC circuit. The veto circuit uses a Flip-Flop, the ready_ff to generate the wREADY signal, the wREADY is negated before being used in the veto process (Line 9 of Listing 4.2). The wNEGHIT is the negated value of the iHIT signal, and it is connected to the clock pin of the ready_ff. So whenever iHIT ends, a rise edge of the wNEGHIT occurs, triggering the FF and setting the value of the wREADY to one. While the wREADY is high, the veto circuit does not allow any other Hit signal to be propagated (see Line 8 of Listing 4.2). When the wREADYRESET becomes high, the value of the ready_ff is cleared, making the wREADY value low, which lets the veto circuit propagate a new Hit value. The synthesis attribute dont_touch was also used in this case.

```
1  (* dont_touch = "TRUE" *) assign wNEGHIT=!iHIT;
2  (* dont_touch = "TRUE" *) FDCE #(.INIT(1'b0)) ready_ff(
3     .Q(wREADY),
4     .C(wNEGHIT),
```

```
5     .CE(1'b1),
6     .CLR(wREADYRESET),
7     .D(1'b1)
8  );
9  (* dont_touch = "TRUE" *)assign wAUXHIT = iHIT & !wREADY;
10 (* dont_touch = "TRUE" *)assign oHIT=wAUXHIT ;
```

Listing 4.2: Input Stage Veto Circuit

4.1.1 TDL

The TDL is the core component of the TDC, and its implementation can be divided into four sections: the delay line, the sample stage, the store stage, and bubble reducer. The implementation of each section will be explained.

First, the implementation of the delay line of the TDL is examined. It was implemented, as shown in section 3.3.1. It is composed of several connected Carry elements that form a Carry chain. To correctly and efficiently connect the Carrys together, a generate construct was used (Line 1 of Listing 4.3). The purpose of the generate construct is to create multiple instantiations of modules and code, or to instantiate blocks of code conditionally. There are two type of generate constructs which can be used separately or together: the generate loop construct that allows multiple instantiations of a block of code, controlled by a variable index; and the conditional generate constructs which select one code of block between multiple blocks, this includes if-generate and case-generate forms [54].

As each Carry has eight CO, therefore the number of Carry elements generated is one-eighth of the number of taps of the delay line (see Line 2 of Listing 4.3). In Line 4 of Listing 4.3, the conditional if-generate is used, because of the singular way that the inputs of the first Carry of the delay line are connected. In this particular case, the CIN is the input of the delay line. Therefore the iHIT signal must be connected to it (Line 11 of Listing 4.3). Whereas in all the other cases, the connection between the Carry elements is done by joining the most significant CO pin of the previous Carry element to the CIN pin of the current element (Line 23 of Listing 4.3).

The number of taps of the delay line needed to interpolate an entire system clock period, which is 2.5 ns, is of 692 taps. As the width of a single taps is 3.8 ps, however as seen in [51], the number of taps of the TDL should be bigger than the clock period, so the 692 taps where chosen, and the total width of the delay line is $692 * 3.8e-12 = 2.63e-9s$.

The Carry element used to create the TDL is specific to the Xilinx FPGA family used. This element has two different configurations; in one, it is a standard 8-bit Carry; in the other, the Carry is divided into two independent 4-bit Carrys. In order to facilitate the creation of the delay line, the 8-bit Carry configuration was used as it can be seen in both Line 7 and 19 of Listing 4.3. Furthermore, due to the specificities of creating a delay line based on a Carry chain the dont_touch synthesis attribute must be used to prevent optimizations, in this case, the attribute is used in the first Carry element, seen in Line 6 of the same Listing, to prevent the optimization of the connection between the Hit signal and the Carry which could render the delay line unusable. Finally, all the DI pins of the Carrys must be zero (Line 13 and 25 of Listing 4.3), guaranteeing that the Hit is signal is not altered while it is being propagated through the line, all of the S pins must be one (Line 14 and 26 of Listing 4.3), so the Hit can correctly propagate through delay line.

```
1  generate
2     for(i=0; i <= `NUM_STAGES/`CARRY_NUMBER-1; i=i+1)
3     begin : generate_block
4        if(i==0)
5        begin
6           (* dont_touch = "TRUE" *)
7           CARRY8 #(.CARRY_TYPE("SINGLE_CY8") )
8           CARRY8_inst (
9              .CO(wFINEVALUE[`CARRY_NUMBER-1:0]),
10             .O(),
11             .CI(iHIT),
12             .CI_TOP(1'b0),
13             .DI(`CARRY_NUMBER'b00000000),
14             .S(`CARRY_NUMBER'b11111111)
15          );
16       end
17       else
18       begin
19          CARRY8 #(.CARRY_TYPE("SINGLE_CY8") )
20          CARRY8_inst (
21             .CO(wFINEVALUE[`CARRY_NUMBER*(i+1)-1:`CARRY_NUMBER*i]),
22             .O(),
23             .CI(wFINEVALUE[`CARRY_NUMBER*i-1]),
24             .CI_TOP(1'b0),
25             .DI(`CARRY_NUMBER'b00000000),
26             .S(`CARRY_NUMBER'b11111111)
27          );
```

```
28        end
29     end
30  endgenerate
```

Listing 4.3: Tapped-Delay Line delay line implementation

The next step is to implement the Sample Stage of Flip-Flop that is responsible for sampling the status of the delay line every clock cycle. To spawn all the FFs used in the first sampling stage the generate loop construct is used (line 1 of Listing 4.4). To do so the D port of the FF is connected to the CO pins of the delay line. This connection is made by the wFINEVALUE wire array. The connection of the wire array to the FF is seen in Line 9 of Listing 4.4 and the connection of the wire array to the Carry can be seen in Line 9 and 21 of Listing 4.3. As seen before, the dont_touch synthesis attribute is also used in this situation to prevent optimizations that could harm the performance and functioning of the system.

```
1  generate
2     for(j=0;j<=`NUM_STAGES-8;j=j+1)
3     begin
4        (* dont_touch = "TRUE" *) FDCE #(.INIT(1'b0)) rTDCVALUE(
5           .Q(wTDCVALUE[j]),
6           .C(iCLK),
7           .CE(1'b1),
8           .CLR(1'b0),
9           .D(wFINEVALUE[j])
10       );
11    end
12 endgenerate
```

Listing 4.4: Sample Stage of Flip-Flop

Afterward, the Store Stage is implemented to store the value of the Sample Stage. The value of the Sample Stage is only stored when the Start or Stop of the Hit Signal occurs. If a rise edge occurred, the rTHERMOMETERSTARTVALUE Flip-Flops stores the status of the FFs, otherwise if it was a fall edge, the status of the Sample Stage is stored by the rTHERMOMETERSTOPVALUE. Both the Start and Stop store Flip-Flops are created with the use of the generate loop construct (Line 1 and 13 of Listing 4.5) and use the dont_touch synthesis attribute (Line 4 and 6 of Listing 4.5), to prevent undesired optimizations. The wTDCVALUE wire array makes the connection between both stages, the connection with the Sample Stage is seen in Line 5 of Listing 4.4, and with the Store Stage in Line 9 and 21 of Listing 4.5 in which it is connected to the D pin of the Flip-Flop.

The iSTORESTART and iSTORESTOP signals are connected to the CE pin of the Start and Stop Store FF respectively, (Line 7 and 19 of Listing 4.5). This pin is the clock enable, and only when it is enabled is the value in the D pin stored in the FF at the rise edge of the clock. The output of these Store Stage FFs are connected to the wTHERMOMETERSTARTVALUE wire array for the Start Store and are connected to the wTHERMOMETERSTOPVALUE wire array for the Stop Store, Line 8 and 17 of Listing 4.5.

```
1  generate
2     for(k=0;k<=`NUM_STAGES-8;k=k+1)
3     begin
4        (* dont_touch = "TRUE" *) FDCE #(.INIT(1'b0)) rTHERMOMETERSTARTVALUE(
5        .Q(wTHERMOMETERSTARTVALUE[k]),
6        .C(iCLK),
7        .CE(iSTORESTART),
8        .CLR(iRST),
9        .D(wTDCVALUE[k])
10       );
11    end
12 endgenerate
13 generate
14    for(l=0;l<=`NUM_STAGES-8;l=l+1)
15    begin
16       (* dont_touch = "TRUE" *) FDCE #(.INIT(1'b0)) rTHERMOMETERSTOPVALUE(
17       .Q(wTHERMOMETERSTOPVALUE[l]),
18       .C(iCLK),
19       .CE(iSTORESTOP),
20       .CLR(iRST),
21       .D(wTDCVALUE[l])
22       );
23    end
24 endgenerate
```

Listing 4.5: Store Stage of Flip-Flop

Finally, the Bubble Reducer is implemented; it is based on a method proposed in [47] to reduce the non-linearities of the delay line. The generate loop construct is used to create all the LUT needed (Line 2 of Listing 4.6). Additionally, the dont_touch synthesis attribute is used to prevent optimizations (Line 1 and Line 5 of Listing 4.6). The input of the LUTs is connected to the wFINEVALUE wire array (Line 7 of the Listing 4.6), to increment the time-constant of the signal. The wire array connected to the output of the LUTs is left unconnected as it does not have any usefulness, but it has to exist for the LUTs to be created.

```
(* dont_touch = "TRUE" *) wire [`NUM_STAGES-1:0] wBubbleReducer;
generate
   for(p=0; p <= `NUM_STAGES-1; p=p+1)
   begin
      (* dont_touch = "TRUE" *) LUT1 #(.INIT(2'h0)) lBubbleReducer (
      .O(wBubbleReducer[p]),
      .I0(wFINEVALUE[p])
      );
   end
endgenerate
```

Listing 4.6: Bubble Reducer

4.1.2 Decoder Module

The Decoder Module has the purpose of decoding the Start and Stop Thermometer code in the least time possible. As seen in Section 3.3.2, the decoder is based on the priority encoder circuit, which outputs the position of the most significant bit of the input, at logic level one in the Start and at logic level zero in the Stop. The decoder circuit is a purely combinational circuit meaning that every time the value of (in the case of the Start Decoder) wTHERMOMETERSTART changes, the value in the rBINSTART also changes. This helps to speed up the decoding process. In order to implement this type of combinational circuit, the decoding process was written inside an always block (Line 1 of Listing 4.7). The decoder reads through the Thermometer code (Line 4 of the Listing 4.7) and detects the last position through which the signal propagated. The detection of the last propagated position is done in line 6 of Listing 4.7, where for each position of the Thermometer code it tests if the current position has the value of one and the next four values of the code are zero, if they are, the decoded value is updated and the analysis of the Thermometer code continues.

```
always @(wTHERMOMETERSTART)
begin
   rBINSTART = 0;
   for(i=0;i<`NUM_STAGES -20; i = i + 1'b1)
   begin
      if(wTHERMOMETERSTART[i] & ~wTHERMOMETERSTART[i+1] &
          ~wTHERMOMETERSTART[i+2] & ~wTHERMOMETERSTART[i+3] &
          ~wTHERMOMETERSTART[i+4])
      begin
         rBINSTART = i + 1;
```

```
9       end
10    end
11 end
```

Listing 4.7: Decoder for the start Thermometer Code

In the case of the Stop Decoder, the implementation is similar to the Start Decoder, except for of the mechanism that detects the last position, which in the case of the Stop Decoder, it searches for the last zero propagated (Line 6 of Listing 4.8). For each position of the Stop Thermometer code it tests if the current position of the code has the value of zero and the next four position the value of one, if this is true than the rBINSTOP value that holds the decoded value is updated with the value of the current position and the analysis of the Thermometer code continues. Similarly to the Start Decoder, the analysis of the code only ends when all the code was analyzed.

```
1
2  always @(wTHERMOMETERSTOP)
3  begin
4     rBINSTOP = 0;
5     for(i=0;i<`NUM_STAGES -20; i = i + 1'b1)
6     begin
7        if(~wTHERMOMETERSTOP[i] & wTHERMOMETERSTOP[i+1] & wTHERMOMETERSTOP[i+2] &
             wTHERMOMETERSTOP[i+3] & wTHERMOMETERSTOP[i+4])
8        begin
9           rBINSTOP = i + 1;
10       end
11    end
12 end
```

Listing 4.8: Decoder for the stop Thermometer Code

4.1.3 Calibration Module

The Calibration Module is responsible for re-adjusting the results of the TDC, to reduce the impact of the non-linearities of the delay line. The implementation of the Calibration Module centers on the use of a Dual-Port RAM. Furthermore, a State Machine is used. It is divided into two parts one for controlling the state of the machine and another to implement the behavior of each state. Finally, a Consultation Manager is implemented to expedite the calibrated values.

As the Calibration Module is based on bin decimation calibration, it must re-organize the taps of the TDL, depending on their real width, in order to create a new delay line where the width of each bin is as close as possible to an ideal bin, creating a more evenly distributed cell size. In order to create that calibrated delay line, an ideal bin with 15.2 ps of width is set. This ideal width is equivalent to the delay of four uncalibrated taps. Therefore the number of taps in the new delay line is 173, which is one-fourth of the number of bins of the uncalibrated TDL.

Firstly the central component of the Calibration module is declared, the RAM memory block, seen in Listing 4.9. The dont_touch synthesis attribute is used to prevent any type of optimizations.

```
(* dont_touch = "TRUE" *) blk_mem_gen_0 calibrationMemory(
   .clka(iCLK),
   .addra(raddra),
   .dina(wdina),
   .douta(wdouta),
   .ena(rena),
   .wea(rwea),
   .clkb(iCLK),
   .addrb(raddrb),
   .dinb(wdinb),
   .doutb(wdoutb),
   .enb(renb),
   .web(rweb)
);
```

Listing 4.9: Calibration Table Memory

Afterward, the section state machine that controls the behavior of the calibration module is declared. As seen in Figure 3.15 in Section 3.3.3, the starting state of the state machine is the RST state, which is written into the rState variable. Whenever a reset happens, the value of rState becomes the value of RST, and the operation of the State Machine is reset (Line 1 and 3 of Listing 4.10). The case statement (Line 6 of Listing 4.10) is used to execute the code corresponding to the current state. The code executed at each state is very similar, the flag that signals the end of the current state is assessed, and if the variable is set, the state changes to the next state. Otherwise the state continues the same for another clock cycle. If for any reason, the value of the rState variable becomes unreliable or erroneous, the rSTATE is reset to the RST state, which restarts module and the calibration process from the beginning.

```
if(iRST)
begin
```

```
3     rState<=RST;
4     end
5  else
6  case(rState)
7     RST: if(rEndRST==1)
8        begin
9           rState<=ACQUISITION;
10       end
11    ACQUISITION: if(rEndAcquisition==1)
12       begin
13          rState<=CONVERSION;
14       end
15    CONVERSION: if(rEndConversion==1)
16       begin
17          rState<=CONSULTATION;
18       end
19    CONSULTATION: if(rConsultation==1)
20       begin
21          rState<=RST;
22       end
23    default: rState<=RST;
24 endcase
```

Listing 4.10: Calibration State Machine

The behavior of the Calibration module at each state is specified in the second section of the state machine, although its current state is controlled by the state machine in Listing 4.10.

In the RST, in the first iteration, every control variable starts with the default value of zero (Line 5 to Line 13 of the Listing 4.16). The Port A of the RAM is enabled to start the process of cleaning the values stored. Furthermore, the address value of Port A is the rRAMposition pointer, which points to the Acquisition Section of the RAM. The Port A write enable is set to high, and the input value of Port A, which is zero in this case as it will be seen later in this section (Listing 4.16), is written into the RAM.

The value of one is given to the rRST register to signal that the first cycle of the RST already ran. After the first iteration, the process of cleaning the values of the RAM continues, by incrementing the rCount variable (Line 32 of Listing 4.11), until all the values are cleaned. When the last position is cleaned, the rEndRST variable, which signals the end of the current state, is enabled, the Port A is disabled, and the rCount variable is cleared (Line 25 to Line 29 of Listing 4.11), and the state changes in the next clock cycle.

```
RST:
begin
   if(rRST==0)
   begin
      rEndRST<=0;
      rEndAcquisition<=0;
      rEndConversion<=0;
      rCalibrationCount<=0;
      rOverFlow<=0;
      rSum<=0;
      rTap<=0;
      rConsultation<=0;
      rWrite<=0;
      raddra<=0+rRAMposition;
      rena<=1;
      rwea<=1;
      rRST<=1;
      rCount<=1'b1;
   end
   else
   begin
      raddra<=rCount+rRAMposition;
      if(rCount==`NUM_STAGES)
      begin
         rEndRST<=1;
         rena<=0;
         rwea<=0;
         rweb<=0;
         rCount<=0;
      end
      else
         rCount<=rCount+1'b1;
   end
end
```

Listing 4.11: Calibration Reset State

In the Acquisition State, firstly, the auxiliary variables used in the last state are cleared. Whenever the iNEWHIT signal is set, it indicates that a new Hit value was received, as it was explained in Section 3.3.3. When this happens the wRISEDGE signal is set for one clock cycle. Meanwhile, a new acquisition process

begins; it has several steps.

In the first step (Line 7 through Line 13 of Listing 4.12), the value of the address port of Port A is changed to the value of the Hit value plus the value of the rRAMposition. Port A is enabled, and the value of the rWrite variable is set to one.

In the next step (Line 15 to 19 of Listing 4.12), the value of the rCount is incremented, this register stores the total number of Hit values already captured by the Calibration Module. This step is also important to let the output value of the Port A of the RAM stabilize so it can later be used.

In the third step (Line 20 to 24 of Listing 4.12), the write enable is set to one and the value at the data input port, which in this state is the value of the output of Port A incremented by one (Listing 4.16), is written into the RAM. By incrementing the value in the current position of the memory it indicates that one more Hit was captured with the value corresponding to the current position of the RAM.

In the last step (Line 25 to Line 33 of Listing 4.12), the Port A write enable is set to zero, and the value of rWrite is reset to the default value. Additionally, the value of rCount is assessed, and if it corresponds to the number of hits expected, the rEndAcquisition is set, and the Acquisition state ends.

```
1  ACQUISITION:
2  begin
3     rRST<=0;
4     rEndRST<=0;
5     if(wRISEEDGE)
6     begin
7        if(rWrite==0)
8        begin
9           raddra<=iHIT+rRAMposition;
10          rWrite<=1;
11          rena<=1;
12          rwea<=0;
13       end
14    end
15    if(rWrite==1)
16    begin
17       rCount<=rCount+1'b1;
18       rWrite<=2;
19    end
20    if(rWrite==2)
21    begin
22       rwea<=1;
23       rWrite<=3;
```

```
24      end
25      if(rWrite==3)
26      begin
27          rwea<=0;
28          rWrite<=0;
29          if(rCount==`CALIBRATION_HITS)
30          begin
31              rEndAcquisition<=1;
32          end
33      end
34  end
```

Listing 4.12: Calibration Acquisition State

In the Conversion state, the values of the Hits stored in the RAM are used to generate the Calibration Table. The memory positions used to store the number of Hits during the Acquisition State will be substituted by the calculated Calibration value. Similar to the Acquisition state, the construction of the Calibration Table is divided into multiple steps, which implement Algorithm 1 presented in Section 3.3.3. Before the start of this construction, the auxiliary variables used in the last state are cleared.

In the first step (Line 4 to Line 10 Listing 4.13), the value of the address port of Port A is set to the beginning of the Acquisition Section. Port A is then enabled (Line 8 of Listing 4.13). The next step (Line 11 to 14 of Listing 4.13), is used so the value outputted by the RAM stabilizes so it can be used for the calculations of the Calibration Table.

In the third step of the CONVERSION state (Line 15 to Line 33 of Listing 4.13), upfront the rSum variable, which stores the current summation value, is added with the value of the current position outputted by the Port A of the RAM (Line 17 of Listing 4.13). This value corresponds with the number of Hits captured at the current position. If the summation of these two values is higher than the value corresponding to the ideal bin size value, the rTap value is incremented by one, and the rOverFlow flag is set (Line 19 and 20 of Listing 4.13). Then the value of the current position is summed to the rSum variable. The auxiliary counter is incremented by one indicating that another value of the Calibration was calculated. If the value rCalibrationCount is equal to the number of taps of the delay line plus one, to guarantee that the last position of the delay line is calibrated, then the rEndConversion variable is set to one indicating that the Calibration Table was completely generated. Otherwise, the rWrite value is set to three in order to execute the last step of the Conversion.

In this last step (Line 34 to Line 43 of Listing 4.13), the write enable is set to one so the value at the input

of Port A, which in this state is the rTap value as seen in Listing 4.16, is written into the current memory position, substituting the number of Hits that arrived at the current position. If an overflow occurred and the rOverFlow flag was set, then the value corresponding to an ideal bin is subtracted to the rSum.

```
CONVERSION:
begin
   rEndAcquisition<=0;
   if(rWrite==0)
   begin
      raddra<=rCalibrationCount+rRAMposition;
      rWrite<=1;
      rena<=1;
      rwea<=0;
   end
   if(rWrite==1)
   begin
      rWrite<=2;
   end
   if(rWrite==2)
   begin
      if(rSum+wdouta[18:0]>`DECIMATED_HITS)
      begin
         rTap<=rTap+1;
         rOverFlow<=1;
      end
      rSum<=rSum+wdouta[`RAM_WIDTH-1:0];
      rCalibrationCount<=rCalibrationCount+1;
      if(rCalibrationCount==`NUM_STAGES+1)
      begin
         rEndConversion<=1;
         rwea<=0;
      end
      else
      begin
         rWrite<=3;
      end
   end
   if(rWrite==3)
   begin
      rwea=1;
      rWrite<=0;
```

```
38        if(rOverFlow)
39        begin
40           rSum<=rSum-`DECIMATED_HITS;
41        end
42        rOverFlow<=0;
43     end
44  end
```

Listing 4.13: Calibration Conversion State

The CONSULTATION state has the purpose of swapping between the Acquisition and Consultation Section in which the Calibration RAM is divided into. As the newer Calibration Table is now on the Acquisition Section, and the Consultation Section holds an outdated Table. These roles of the Section are given by the value of the most significant bit of both the rRAMposition and rCONSULTATIONposition pointers, so to change them only the value of the most significant bit of each pointer must change with each other. If the rBlockSelection selection flag is zero, which is the case in the first time the Calibration Table is generated, the most significant bit of rRAMposition becomes one, and the most significant bit of rCONSULTATIONposition becomes zero, and the value of the selection flag is set to one (Line 8 through 10 of Listing 4.14). If the selection flag is one, firstly, the flag is set to zero, and then the value of most significant bits of both pointers are changed (Line 14 to Line 16 of Listing 4.14).

If it is the first time that the Calibration Table was generated, meaning that before there were no Calibration values available and the Calibration was disabled, the rFirst flag is set to one (Line 19 through Line 22), which enables the functioning of the Consultation circuit. Finally, the rEndConsultation flag is set, the state machine is reset, and the whole process starts again.

```
1  CONSULTATION:
2  begin
3     rEndConversion<=0;
4     if(rEndConversion==1)
5     begin
6        if(rBlockSelection==0)
7        begin
8           rRAMposition<=11'h400;
9           rCONSULTATIONposition<=11'h0;
10          rBlockSelection<=1;
11       end
12       else
13       begin
```

```
14          rBlockSelection<=0;
15          rRAMposition<=11'h0;
16          rCONSULTATIONposition<=11'h400;
17        end
18    end
19    if(rFirst==0)
20    begin
21        rFirst<=1;
22    end
23    rEndConsultation<=1;
24 end
```

Listing 4.14: Calibartion Consultation State

The Calibration Module circuit is only enabled after the generation of the first Calibration Table. Only then is the Consultation Manager enabled and the module starts providing calibration values for the received Hit (Line 1 of Listing 4.15). The Port B of the RAM is enabled only after the rFirst flag is set (Line 3 of Listing 4.15). The Consultation Module provides the calibrated value for the current Hit (Line 6 of Listing 4.15).

```
1 if(rFirst==1)
2 begin
3    renb<=1;
4    if(wRISEEDGE)
5    begin
6        raddrb<=iHIT+rCONSULTATIONposition;
7    end
8 end
```

Listing 4.15: Calibrated Value Consultation

The value available at the data input port of Port A of the RAM changes depending on the current state of the state machine as it was referred before. Therefore, to calculate the values for the various states, a combinational circuit was used to reduce the time needed to provide the result.

In the ACQUISITION state, the value available at the input is the value of the output port of the RAM incremented by one, which is used to increment the number of times a Hit arrived with the value corresponding to the current position.

In the CONVERSION state. the value in the input port is the rTap. This value is the calibrated ideal bin calculated at the current state, which is correlated to the current position of the RAM.

In the RST state, the value in the input port is zero, since it is used to clean the values that were previously present at the Acquisition Section of the RAM. In the other states, the port is not used, so the value is zero.

```
assign wdina=(rState==ACQUISITION)? wdouta+1'b1:(rState==CONVERSION)?
    rTap:(rState==RST)?`RAM_WIDTH'h0:`RAM_WIDTH'h0;
```

Listing 4.16: Port A Input

4.1.4 Coarse Counter

The Coarse Counter is responsible for measuring the Coarse Time of the Hit signal. As a result of the synchronization problem explained in Section 3.3.4, a Synchronizer must be implemented. To implement it two extra Coarse Counters were used.

The Coarse Counter implemented measures the duration of the Hit signal in the reference clock cycles through the value stored in rCOARSEVALUE, which is incremented at each clock cycle if the Hit signal is enabled (the Hit signal is used as the counter enable signal) (see Line 8 and 10 of Listing 4.17) When the iSTORE signal is set, the rCOARSEVALUE value is stored so it can later be used (see Line 17 of Listing 4.17), the iSTORE signal is only set after the end of the Hit signal.

```
1  always @(posedge iCLK)
2  begin
3     if(iRST)
4     begin
5        rCOARSEVALUE <= 0;
6     end
7     else
8     if(iHIT)
9     begin
10       rCOARSEVALUE <= rCOARSEVALUE + 1'b1;
11    end
12 end
13 always @(posedge iCLK)
14 begin
15    if(iSTORE)
16    begin
17       rCOARSEVALUESTORED <= rCOARSEVALUE;
18    end
19 end
```

Listing 4.17: Coarse Counter Implementation

As seen in Section 3.3.4 to implement the Coarse Counter and the Synchronizer, three identical Counters are used. The Coarse Counter is connected to the reference clock of the system (see Line 2 of Listing 4.18), while the Synchronizer's Counters are connected to two different clock sources (see Line 9 and 16 of Listing 4.18), the iCLK1 is connected to a clock with the same frequency that the reference clock but with a phase of 72 degrees while the iCLK2 is configured the same way with the difference of having a phase of 144 degrees.

```
1  COARSECOUNTER coarse_cnt_inst(
2     .iCLK(iCLK0),
3     .iRST(wENDOFCONVERTION),
4     .iHIT(wHIT),
5     .iSTORE(wFALLEDGE),
6     .oCOARSEVALUE(wCOARSEVALUE)
7  );
8  COARSECOUNTER coarse_cnt_arbiter_inst(
9     .iCLK(iCLK1),
10    .iRST(wENDOFCONVERTION),
11    .iHIT(wHIT),
12    .iSTORE(wAFALLEDGE),
13    .oCOARSEVALUE(wCOARSEARBITERVALUE)
14 );
15 COARSECOUNTER coarse_cnt_arbiter2_inst(
16    .iCLK(iCLK2),
17    .iRST(wENDOFCONVERTION),
18    .iHIT(wHIT),
19    .iSTORE(wA2FALLEDGE),
20    .oCOARSEVALUE(wCOARSEARBITER2VALUE)
21 );
```

Listing 4.18: Coarse Counter and Synchronizer Counter

The Synchronizer Decider uses the value of the two Synchronizer Counters and the values outputted by the Decoder Module to analyze if a metastability error occurred. If it has, the Decider changes the value of the Coarse Counter accordingly to correct it (see Line 4 to Line 15 of Listing 4.19). In order to reduce resource utilization, the implementation of the Synchronizer was combined with the Merge Module, so the counter value is only corrected at the moment of merging the Coarse and Fine measurements.

```
1  assign wSTARTPHASEDECIDER = (wBINSTART > 210) ? 1 : 0;
2  assign wSTOPPHASEDECIDER = (wBINSTOP > 210) ? 1 : 0;
3  assign wCALIBRATIONTDCVALUE =
4  (rBINSTART==0 & wCOARSEVALUE<= wCOARSEARBITERVALUE & wCOARSEARBITERVALUE >
       wCOARSEARBITER2VALUE)?
5  {wCOARSEARBITERVALUE, wCalibratedStart, wCalibratedStop}:
6  (rBINSTART == 0 & wCOARSEVALUE <= wCOARSEARBITERVALUE & wCOARSEARBITERVALUE ==
       wCOARSEARBITER2VALUE & ~wSTOPPHASEDECIDER)?
7  {wCOARSEARBITERVALUE, wCalibratedStart, wCalibratedStop}:
8  (rBINSTART == 0 & wCOARSEVALUE <= wCOARSEARBITERVALUE & wCOARSEARBITERVALUE ==
       wCOARSEARBITER2VALUE & wSTOPPHASEDECIDER)?
9  {wCOARSEARBITERVALUE + 1'b1, wCalibratedStart, wCalibratedStop}:
10 (rBINSTOP == 0 & wCOARSEVALUE >= wCOARSEARBITERVALUE & wCOARSEARBITERVALUE
       <wCOARSEARBITER2VALUE)?
11 {wCOARSEARBITERVALUE, wCalibratedStart, wCalibratedStop}:
12 (rBINSTOP == 0 & wCOARSEVALUE >= wCOARSEARBITERVALUE & wCOARSEARBITERVALUE ==
       wCOARSEARBITER2VALUE & ~wSTARTPHASEDECIDER)?
13 {wCOARSEARBITERVALUE,wCalibratedStart,wCalibratedStop}:
14 (rBINSTOP == 0 & wCOARSEVALUE >= wCOARSEARBITERVALUE & wCOARSEARBITERVALUE ==
       wCOARSEARBITER2VALUE & wSTARTPHASEDECIDER)?
15 {wCOARSEARBITERVALUE - 1'b1, wCalibratedStart, wCalibratedStop}:{wCOARSEVALUE,
       wCalibratedStart, wCalibratedStop};
```

Listing 4.19: Synchronyzer Decider

4.2 FIFO Module Implementation

The FIFO stores the values of the measurements done by the TDC, allowing the PS to read the values at a lower rate than the Hit sampling rate.The FIFO is constituted by various interconnected blocks (see Figure 3.19). Firstly, the implementation of the connections between the various block, and how they connect to the outputs has to be explained. Afterward, each module is analyzed and explained.

The FIFO memory (see Line 2 through 9 of Listing 4.20) connections are divided into the read and write sections, in the write section the data input is connected directly to the inputs of the FIFO, the write pointer is connected to the Write Pointer Generator. Finally, the write memory is connected to the wMEMEN signal. Whenever this signal is set, a new value is written in the FIFO, and if the FIFO is full, then no value can be written. This is guaranteed in Line 1 of Listing 4.20, where only when the wFULL is not set can a new value be written. The wFULL signal is set to one when the FIFO is full.

The Read-to-Write Synchronization Module (see Line 10 to 15 of Listing 4.20) receives the value of the read pointer from the Read Pointer Generator and outputs its value, synchronized with the write clock, so that it can be used in the Write Pointer Generator. While the Write-to-Read Synchronization Module (see Line 16 to 21 of Listing 4.20) receives the value of the write pointer from the Write Pointer Generator and outputs it synchronized with the read clock, so that it can be used Read Pointer Generator.

Whenever the Write Pointer Generator Module (see Line 22 to 30 of Listing 4.20) receives the iWINC signal, it increments the value of the write pointer. There are two ports that output the write pointer. One is connected to the FIFO memory, and the other, which is encoded in gray code, is connected to the Write-to-Read Synchronization Module. Finally, the module also outputs the wFULL wire, which signals if the FIFO is full or not.

The Read Pointer Generator (see Line 31 to 39 of Listing 4.20) is connected iRINC signal when it is received, the value of the read pointer is incremented. The read pointer is outputted into two different modules, the FIFO memory, and the Read-to-Write Synchronization Module, which receives the value of the pointer encoded in gray code. Lastly, it also generates the wEMPTY flag, which indicates if there are values to be read in the FIFO. The wEMPTY flag is merged into the value read from the FIFO with the intention of indicating if the value read from the FIFO was valid or if it was an invalid because it was read before.

```
1  assign wMEMEN = (wWINC & ~wFULL);
2  FIFO_MEM fifo_mem_inst(
3     .iCLK(iWCLK),
4     .iWEN(wMEMEN),
5     .iWDATA(iWDATA),
6     .iWADDR(wWADDR),
7     .iRADDR(wRADDR),
8     .oRDATA(wRDATA)
9  );
10 SYNCR2W r2w_sync_inst(
11    .iWCLK(iWCLK),
12    .iRST(iWRST),
13    .iRPTR(wRPTR),
14    .oSYNCRPTR(wSYNCRPTR)
15 );
16 SYNCW2R w2r_sync_inst(
17    .iRCLK(iRCLK),
18    .iRST(iRRST),
19    .iWPTR(wWPTR),
```

```
20     .oSYNCWPTR(wSYNCWPTR)
21 );
22 FIFO_WPTR_GEN wptr_gen_inst(
23     .iCLK(iWCLK),
24     .iRST(iWRST),
25     .iWINC(wWINC),
26     .iSYNCRPTR(wSYNCRPTR),
27     .oFULL(wFULL),
28     .oWPTR(wWPTR),
29     .oWADDR(wWADDR)
30 );
31 FIFO_RPTR_GEN rptr_gen_inst(
32     .iCLK(iRCLK),
33     .iRST(iRRST),
34     .iRINC(iRINC),
35     .iSYNCWPTR(wSYNCWPTR),
36     .oEMPTY(wEMPTY),
37     .oRPTR(wRPTR),
38     .oRADDR(wRADDR)
39 );
40 assign oRDATA={wEMPTY,wRDATA};
41 assign oFULL = wFULL;
42 assign oEMPTY = wEMPTY;
```

Listing 4.20: FIFO Module Block Declaration

The FIFO_MEMORY is implemented using an array of registers (see Line 1 of Listing 4.21), which is inferred into a BRAM during the synthesis process explained in [55]. The iWEN is used to control the memory write mechanism, only when the iWEN signal is set, is the value present in iWDATA written into the memory (see Line 5 and Line 7 of Listing 4.21), the position in which it is written is given by the iWADDR pointer. The position of the value which is being read is given by the iRADDR (see Line 3 of Listing 4.21).

```
1 reg [`CALIBRATION_MEMLENGTH-1:0] rFIFOMEM [0:`CALIBRATION_MEMDEPTHPOS-1];
2 assign oRDATA = rFIFOMEM[iRADDR];
3 always @(posedge iCLK)
4 begin
5     if(iWEN)
6     begin
7         rFIFOMEM[iWADDR] <= iWDATA;
8     end
9 end
```

Listing 4.21: FIFO Memory

The Read-to-Write Synchronization is used to synchronize the value of the read pointer with the write clock so that it can be used in the WPTR_GENERATOR module. The synchronization is done in Line 10 and 11 of Listing 4.22. The value of the of iRPTR is first written into rSyncDFF1, and in the next clock cycle it is being outputted by the module (see Line 14 Listing 4.22), this clock delay guarantees that the read pointer in spite of being generated at a different clock rate is stable and ready to be used.

The implementation of the Write-to-Read Synchronization is equal to the Read-to-Write one. The main changes are the name of the registers used and what is connected to the inputs and outputs.

```
1  always @(posedge iWCLK or negedge iRST)
2  begin
3     if(!iRST)
4     begin
5        rSyncDFF1 <= 0;
6        rSyncDFF2 <= 0;
7     end
8     else
9     begin
10       rSyncDFF1 <= iRPTR;
11       rSyncDFF2 <= rSyncDFF1;
12    end
13 end
14 assign oSYNCRPTR = rSyncDFF2;
```

Listing 4.22: FIFO Read-to-Write Synchronization Module

The Write Pointer Generator generates the position in which the next value of the FIFO will be written. Additionally, it is also used in the generation of the FIFO full flag. The value of the pointer is set to zero whenever the system is reset (see Line 3 to Line 6 of Listing 4.23). The value of the pointer is incremented every time the iWINC wire is set, and the FIFO is not full (see Line 16 of Listing 4.23). The value is then converted to Gray Code to be later used in the generation of the full flag and passed into the Read Pointer Generation to be used in the generation of the empty flag (see Line 11, Line 14 and Line 17 of Listing 4.23). The conversion of the value of the pointer from binary to Gray Code is necessary when the pointer must be used in two asynchronous clock domains. The Gray code ensures that only a single bit changes as the pointer value is incremented, so if one clock happens to catch a transition in the other clock domain, at most, one bit can be metastable, reducing the ambiguity between two adjacent states of the FIFO. The

generation of the wFULL flag is done by comparing the current position of the write pointer encoded in Gray Code with the position of the read pointer, if the value of both is equal then the wFULL flag is set to one otherwise it is reset to zero. The value of the two most significant bits of the synchronization pointer must be negated to be used in the comparison. The value of the flag is then written into a register and outputted (see Line 27 of Listing 4.23).

```
1  always @(posedge iCLK or negedge iRST)
2  begin
3     if(!iRST)
4     begin
5        rWBIN_ADDR <= 0;
6        rWPTR <= 0;
7     end
8     else
9     begin
10       rWBIN_ADDR <= wWNEXTBIN_ADDR;
11       rWPTR <= wWNEXTGRAY_ADDR;
12    end
13 end
14 assign oWPTR = rWPTR;
15 assign oWADDR = rWBIN_ADDR[`CALIBRATION_MEMDEPTH-1:0];
16 (*dont_touch = "TRUE"*)assign wWNEXTBIN_ADDR = rWBIN_ADDR + (iWINC & ~rFULL);
17 (*dont_touch = "TRUE"*)assign wWNEXTGRAY_ADDR = (wWNEXTBIN_ADDR >> 1) ^
       wWNEXTBIN_ADDR;
18 assign wFULL = (wWNEXTGRAY_ADDR ==
       {~iSYNCRPTR[`CALIBRATION_PTRLENGTH-1:`CALIBRATION_PTRLENGTH-2],
       iSYNCRPTR[`CALIBRATION_PTRLENGTH-3:0]});
19 always @(posedge iCLK or negedge iRST)
20 begin
21    if(!iRST)
22    begin
23       rFULL <= 0;
24    end
25    else
26    begin
27       rFULL <= wFULL;
28    end
29 end
30 assign oFULL = rFULL;
```

Listing 4.23: FIFO Write Pointer Generator

The Read Pointer Generator generates the current position of the FIFO that is being read.It is also used in the generation of the FIFO empty flag The value of the pointer is set to zero every time the reset signal is set (see Line 3 to Line 6 of Listing 4.24). The value of the pointer is incremented whenever the iRINC is set, and a value is available to be read in the FIFO (see Line 16 of Listing 4.24). The value of the pointer is encoded into Gray Code to be used in the comparison that generates the empty flag, and to be passed to the Write Pointer Generator to be used for the full flag comparison (see Line 11 and Line 14 of Listing 4.24). The wEMPTY flag is the result of the comparison between wRNEXTGRAY_ADDR and the synchronization pointer from the Write Pointer, if the value of both is equal, then the FIFO is empty, and the flag is set (Line 18 of Listing 4.24). The value of the flag is then written into a register and outputted (see Line 27 of Listing 4.24).

```
1  always @(posedge iCLK or negedge iRST)
2  begin
3     if(!iRST)
4     begin
5        rRBIN_ADDR <= 0;
6        rRPTR <= 0;
7     end
8     else
9     begin
10       rRBIN_ADDR <= wRNEXTBIN_ADDR;
11       rRPTR <= wRNEXTGRAY_ADDR;
12    end
13 end
14 assign oRPTR = rRPTR;
15 assign oRADDR = rRBIN_ADDR[`CALIBRATION_MEMDEPTH-1:0];
16 assign wRNEXTBIN_ADDR = rRBIN_ADDR + (~rEMPTY & iRINC);
17 assign wRNEXTGRAY_ADDR = (wRNEXTBIN_ADDR >> 1) ^ wRNEXTBIN_ADDR;
18 assign wEMPTY = (wRNEXTGRAY_ADDR == iSYNCWPTR);
19 always @(posedge iCLK or negedge iRST)
20 begin
21    if(!iRST)
22    begin
23       rEMPTY <= 1;
24    end
25    else
26    begin
27       rEMPTY <= wEMPTY;
28    end
```

```
29 end
30 assign oEMPTY = rEMPTY;
```

Listing 4.24: FIFO Read Pointer Generator

4.3 AXI Module Implementation

The AXI Module enables the communication of multiple TDC_Peripherals with the PS. As seen in Section 3.5, a set of commands was developed to allow the connection of several TDC_Peripherals. Furthermore, the read section of the AXI Module was redesigned for the same purpose.

The AXI Module receives various commands from Processing System; whenever a new command is written into the AXI, it must be read and identified. The commands received from the PS have the format depicted in Figure 3.20. The arrival of a new command is identified when the mem_wren is set to one (see Line 3 of Listing 4.25). The recognition of the type of command is done with the use of a switch (see Line 3 to Line 24 of Listing 4.25). The instruction part of the command is evaluated to identify which command was received (see Line 5 of Listing 4.25). The rReadFIFO register is connected to the READFIFO input of every TDC_Peripheral and is used to request a new value from the FIFO of the peripheral, while the rRST register has the purpose of resetting all the TDC_Peripherals to their initial states clearing the calibration acquired so far.

The NOP command clears the values of both the rReadFIFO and the rRST register. The READ_CHANNEL command is used to request a new value from a specific TDC_Peripheral. The channel is specified by the channel address part of the command, and this is done by setting to one the bit of the rReadFIFO connected to the channel (see Line 13 of Listing 4.25). The READ_ALL command requests a new value from all peripherals, thus every bit of rReadFIFO is set to one. The RST command is used to reset all of the TDC_Peripherals, which is done by enabling the rRST register (see Line 22 of Listing 4.25). When the mem_wren is zero, which means that there is no command to be read, the value of the rReadFIFO and rRST are cleared (see Line 28 and 29 of Listing 4.25).

```
1 always @(posedge S_AXI_ACLK)
2 begin
3   if(mem_wren)
4   begin
5   case(data_in[`INSTR_SIZE-1+`ADDR_SIZE:`ADDR_SIZE])
6     `NOP:
```

```
7        begin
8           rReadFIFO<=0;
9           rRST<=0;
10       end
11       `READ_CHANNEL:
12       begin
13          rReadFIFO[data_in[`ADDR_SIZE-1:0]]<= 1'b1;
14       end
15       `READ_ALL:
16       begin
17          rReadFIFO<=32'hffff_ffff;
18       end
19       `RST:
20       begin
21          rReadFIFO<=0;
22          rRST<=1;
23       end
24    endcase
25    end
26    else
27    begin
28       rReadFIFO<=0;
29       rRST<=0;
30    end
31 end
```

Listing 4.25: AXI Full Write

The read section of the AXI Module reads values from the various TDC_Peripherals and makes it available in the PS. When the PS requests to read a memory position from the AXI Module, the axi_rvalid is set to one (see Line 1 and Line 3 of Listing 4.26). Subsequently, the value of the FIFODATA of the TDC_Peripheral in the position given by axi_araddr is written into the output register axi_rdata and later passed into the Processing System (see Line 5 of Listing 4.26).

```
1  always @(axi_rvalid)
2  begin
3     if (axi_rvalid)
4     begin
5        axi_rdata <=wOutputChannels[axi_araddr/4];
6     end
7     else
```

```
8     begin
9        axi_rdata <= 32'h00000000;
10    end
11 end
```

Listing 4.26: AXI Full Read

The implementation of multiple TDC_Peripherals uses the generate loop construct; the number of channels implemented is given by the NUMBER_CHANNELS macro. The iREADFIFO of every channel is connected to the rReadFIFO (see Line 11 of Listing 4.27), the reset of the channels is connected to the rRST (see Line 10 of Listing 4.27), and the output of the channels is connected wOutputChannels (see Line 12 of Listing 4.27). The Hit signal of each channel is connected to the wire array that contains all of the Hit signals that are measured by the Acquisition System (see Line 9 of Listing 4.27).

```
1  generate
2     for(k=0; k <= `NUMBER_CHANNELS-1; k=k+1)
3     begin : generate_block
4     TDC_peripheral channel
5     (
6        .iCLK(iCLK),
7        .iCLK1(iCLK1),
8        .iCLK2(iCLK2),
9        .iHIT(iHIT[k]),
10       .iRST(rRST),
11       .iREADFIFO(rReadFIFO[k]),
12       .oCALIBRATEDDATA(wOutputChannels[k]),
13       .oFULL(wFULL[k])
14    );
15    end
16 endgenerate
```

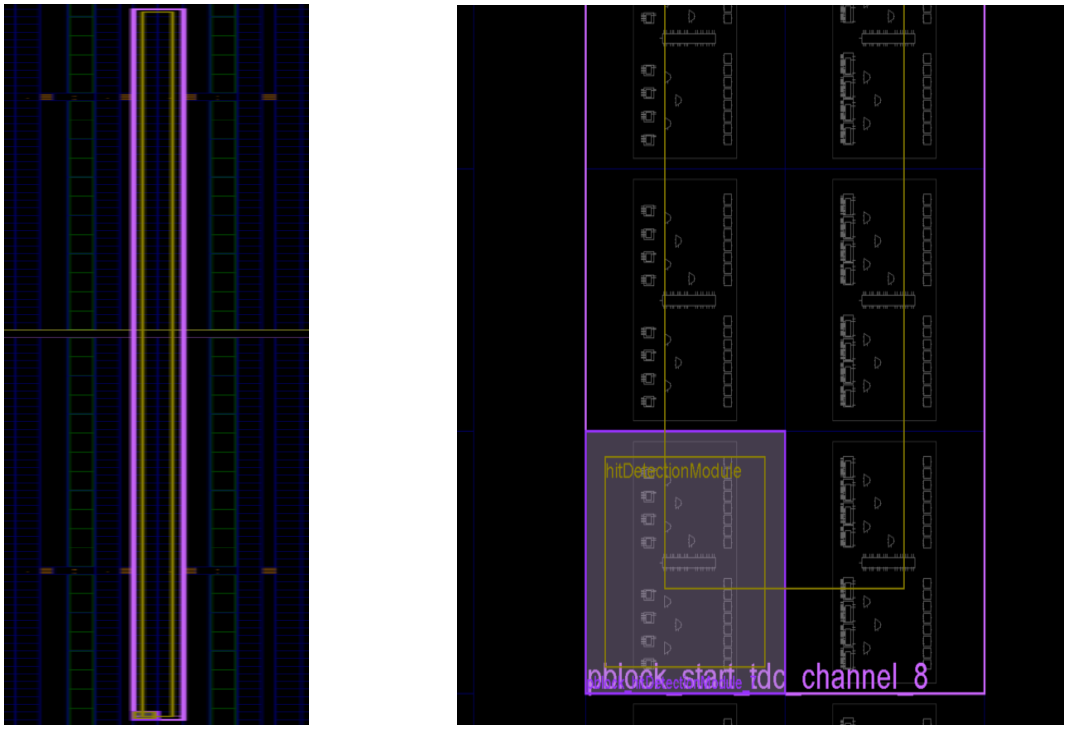
Listing 4.27: Implementation of multiple TDC_Peripherals with the AXI Peripheral

4.4 Floorplanning and Constraints

In order to implement a Acquisition System with sixteen TDC channels, there was the need to floorplan the placement of some of the modules. The floorplan has two main goals: to ensure that the various TDC channels are evenly distributed in terms of resource utilization, and to ensure that they are placed in way that ensures the best performance possible to all channels.

After analyzing the design of the TDC it was concluded that the modules that have more influence in the performance of the of a single channel are the TDL Module and the Input Stage Module, making it necessary to plan how and where these two modules are placed. Both Modules must be instantiated together as the Hit signal is outputted from the Input Stage into the TDL.

There are several techniques possible to constraint the placement of Modules in a FPGA, in the current case the Pblock was used. A Pblock is a collection of cells and one or more rectangular areas that specify the device resources contained by the Pblock [56]. One or more modules can be assigned into a single Pblock and a Pblock can be declared inside another Pblock.



(a) **(b)**

Figure 4.1: Example of the Pblocks for one TDC channel **(a)**, and detail of the Input Stage Pblock **(b)**

So for each TDC channel of the Acquisition System two Pblocks are used, one for the TDL (see Figure 4.1a), and another inside of the TDL Pblock for the Input Stage Module (see Figure 4.1b). The placement of the sixteen channels was done by trial and error until an optimal distribution of the channels was found, this distribution ensures the best performance for each channel and an even distribution of the resource utilization, allowing for other modules to be implemented.

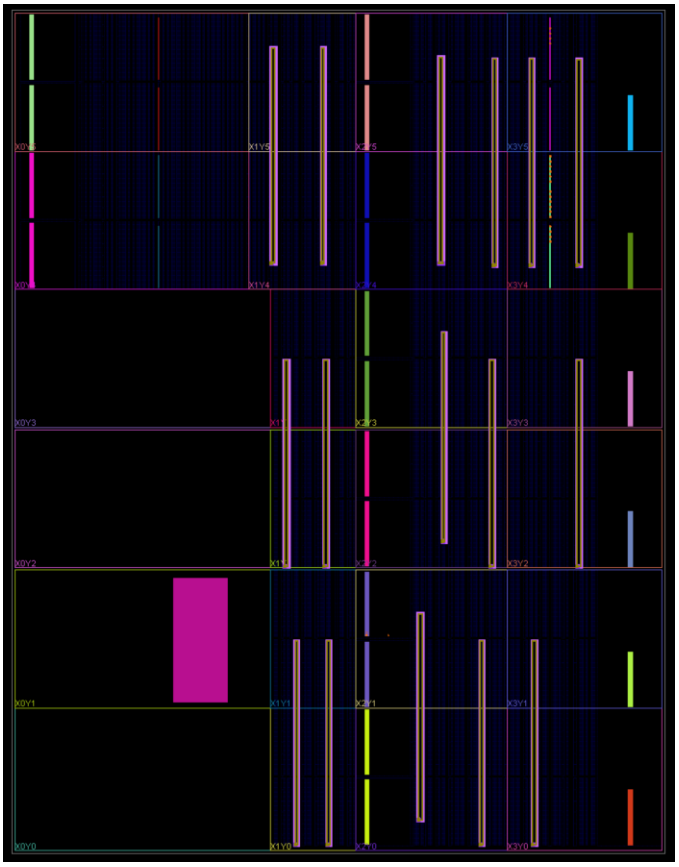


Figure 4.2: Distribution of the sixteen Pblocks of the channel of the Acquisition System in the FPGA

4.5 Conclusion

This chapter described the process of implementing the Acquisition System. It gave an overview of how every module was implemented and gave emphasis to the implementation specificities of each module.

Chapter 5

Tests and Results

In this chapter, the tests, and results that demonstrate the performance of the Acquisition System are presented. In order to do so, the performance of each channel of the system must be assessed, which is done by applying the tests presented in Section 3.6, which are used to determine the linearity and precision of the channel.

Furthermore, the influence of the number of TDC channels implemented in the FPGA is examined in order to study if the decrease of resource utilization associated with the reduction of the number of channels raises the performance of the implemented TDC channels. The objective of this study is to conclude if the multiple TDC channel Acquisition System is feasible and if there is an optimal number of implemented channels, that balances individual channel performance and resource utilization with the number of implemented channels.

5.1 Test Setup

In order to analyze the performance of each channel of the Acquisition System, a Test Setup was built. The setup has the objective of generating the Hit signals for the TDC. The signals are generated by the arbitrary function generator AFG1022 of Tektronix (see Figure 5.1). The Hit signal generated by the AFG1022 is a square wave signal outputted by both output channels of the generator with a 3.3 MHz frequency, this frequency was selected to avoid correlation with the system clock of the TDC.

The generated signal is connected to the board via a BNC to Alligator Cable connected to jumper wires, which are connected to the PMOD inputs of the board. Although sixteen channels are implemented, only two Hit signals are used to reduce the noise and signal interference associated with the use of several Hit signals, and also to reduce the problems associated with the different routing of the signals. The two Hit

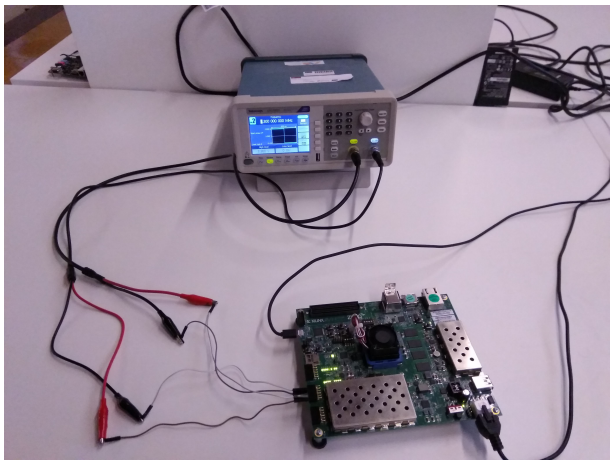


Figure 5.1: Test Setup.

signals used are then distributed uniformly between the inputs of the various channels of the Acquisition System.

5.2 Results

In order to test the influence of the number of channels implemented in the individual performance of each channel, five implementations with a different number of implemented channels were tested. They have sixteen, eight, four, two, and one implemented channels, respectively. The channels and their Pblocks are always placed in the same position on the board, reducing the effect of the changes of placement in the performance of the TDCs. It is also essential to take into account the resource utilization not only in its effect on the performance of the TDC channels but also to see what resources are still available to be used by other modules besides the Acquisition System if needed.

As the goal of this dissertation is to implement an Acquisition System with sixteen TDC channels, this implementation will be considered as the Target implementation, and it will be used as the reference point in terms of performance comparison with other implementations. For each implementation, the bin width of the taps of every channel is analyzed, later the precision of all channels is considered and examined. Then the resource utilization for each implementation is analyzed. Afterward, the results of the channels with the best and the worst performance of each implementation are presented. In this case, the performance of the channels is measured by their precision. For those channels, the bin width and the linearity for both the Start and Stop are presented, and the precision of the channels analyzed.

The performance and bin widths of the delay line vary depending on the propagation of a rise or fall edge signal due to the physical properties of the hardware. So it is necessary to analyze the line both for the Start and the Stop of the Hit signal.

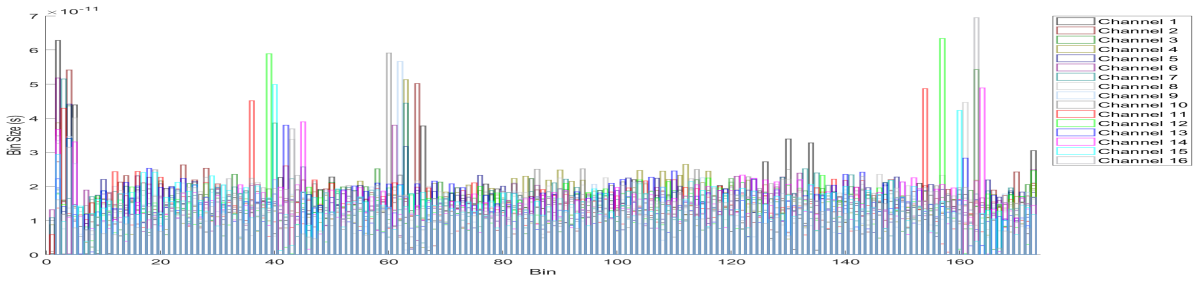
As seen in Section 4.1.3, the size of the ideal calibrated Bin is 15.2 ps, and the number of Bins in the calibrated delay line is 173.

The Results of the tests of each channel of the various implementations that are not displayed in the respective Section can be found in Appendix A.

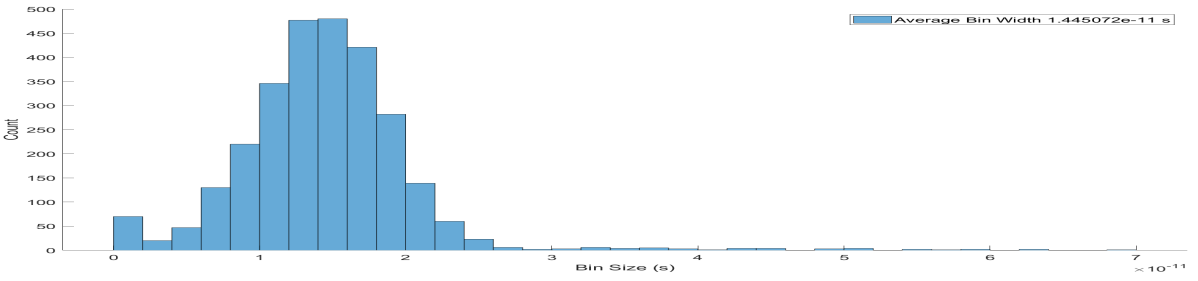
5.2.1 Results for 16-Channel Implementation

The implementation of an Acquisition System with sixteen channels is the goal of this dissertation, so the results of the current implementation must be thoroughly analyzed.

As it can be seen in Figure 5.2b, in the Start Line the majority of the Bins have a width very close to the ideal bin size of 15.2 ps, and the average Bin width of the Start is 14.45 ps which is very close to the ideal bin size. However in every channel there is at least one Bin which has a width that is substantially bigger than the ideal width (see Figure 5.2a), being the widest bin in the tap 163 of channel 16 with the width of 69.5 ps. The average number of bins with a width superior to 30 ps, which can be considered ultra-wide bins, is 2.81.



(a)



(b)

Figure 5.2: Measured Bin Width **(a)** for each channel and **(b)** the bin width Histogram of the Start of all the bins of the 16-Channel Implementation.

These very large Bins create the problem of very small Bins or even zero-width Bin, which were approached in Section 3.3.3, that are responsible for deteriorating the performance of the line, there is an average of 30.4 Bins with a width smaller than 10 ps in each channel.

The results of the Stop Line (see Figure 5.3) are very similar to the ones of the Start Line, although it can be seen that the width of the same Bin is different between the Start and Stop. However, the average Bin width is the same in both cases (see Figure 5.3b). Furthermore, there still are some ultra-wide Bins in the Stop line however they are in smaller number than in the Start line (see Figure 5.3a), with the largest tap being the second Bin of the first channel, with a size of 61.45 ps, and an average of 3.06 ultra-wide bins per channel, a slightly higher average than in the Start. Even though the size of the ultra-wide Bin is smaller, the average number of zero Bins in the Stop Line is slightly higher than in the Start Line, with an average of 32.94 Bins with less than 10 ps.

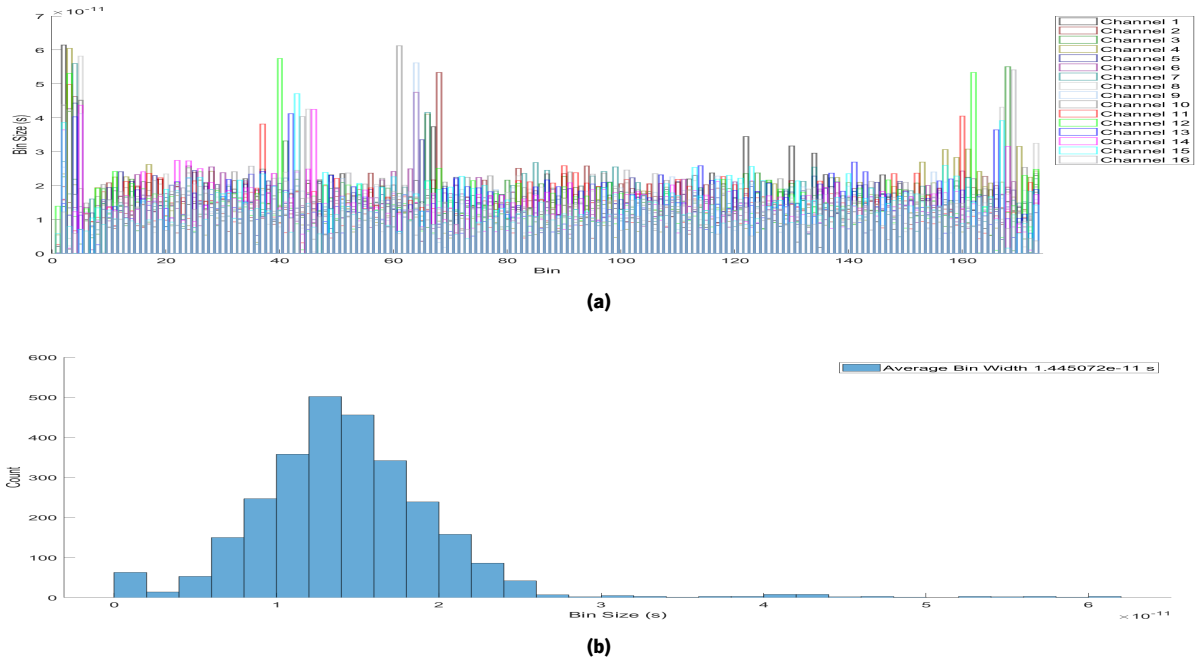


Figure 5.3: Measured Bin Width **(a)** for each channel and **(b)** the bin width Histogram of the Stop of all the bins of the 16-Channel Implementation.

In Figure 5.4a, the measurement of the signal used to perform the Code Density Test is shown. The Hit signal had a frequency of 3.3 MHz, but only the high part of the signal is measured, meaning that the Hit signal has a size of 151.5 ns. The majority of the measurements made by the different channels are similar to the real time of the signals, with an average measurement value of 151.34 ns. Nonetheless, there is a significant number of measurements that differ from the expected value, by 2.5 ns, which is one system clock period. This can be explained by the Coarse Counter Metastability error, seen in Section

3.3.4, which adds or subtracts one clock cycle to the measured coarse value. In spite of this Metastability error, which alters the measurement value and deteriorates the precision of the measurements, it varies between 1.27 ns in channel 1, and 349.8 ps in channel 13, the average precision of the channels is 731.1 ps.

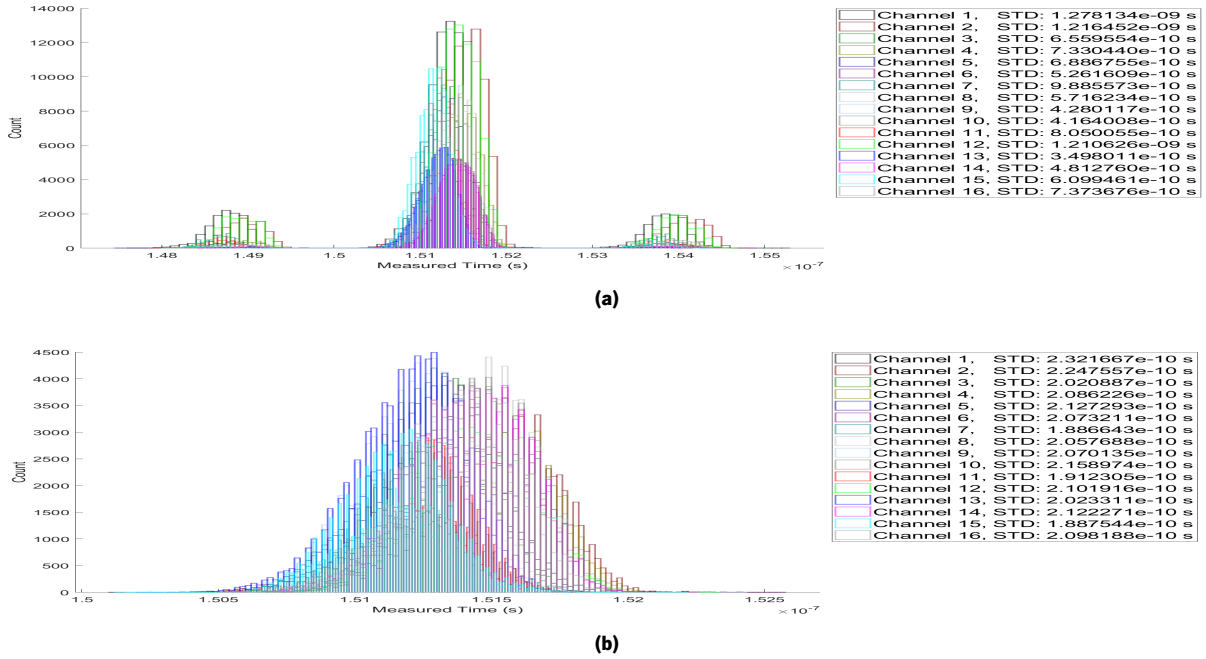


Figure 5.4: Measurement Histogram and Precision, of all channels of the 16-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

The performance of each channel is greatly dependent on the effect of the Metastability error, so it is difficult to compare the different channels due to its effect. In order to to enable the comparison of the channels only the measurements that do not present the Metastability error are taken into account, to make it possible to analyze the influence of the calibrated line in the measurement (see Figure 5.4b). If only these measurements were considered, the precision of the Acquisition System would be far superior, with precision varying between 232.17 ps and 188.66 ps, with an average precision of 207.47 ps.

5.2.1.1 Resource Utilization

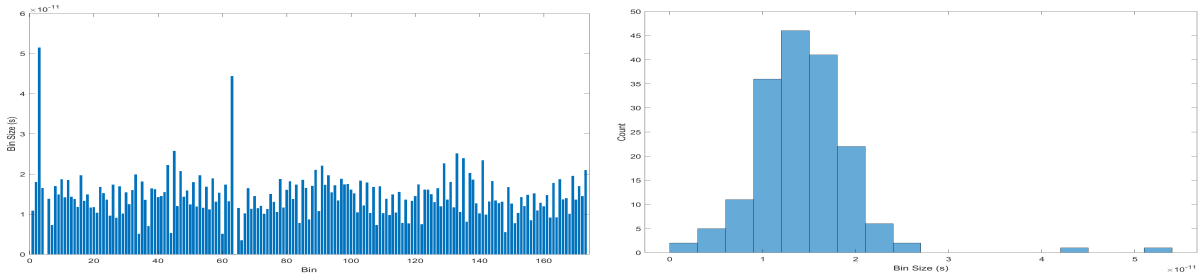
As seen in Table 5.1, even with sixteen channels implemented, the resources utilized are still low, and there are still enough available resources for implementing other peripherals that could be needed. The resource with the highest utilization is the LUT, where 28.59 percent where used, all other resources report a utilization under 20 percent.

Resource	Utilization	Available	Utilization
LUT	65872	230400	28.59
LUTRAM	523	191760	0.51
FF	44567	460800	9.67
BRAM	48	312	15.38
IO	6	360	1.67
BUFG	20	544	3.68
MMCM	1	8	12.50

Table 5.1: Resource Utilization of the 16 Channel Implementation

5.2.1.2 Channel with the best performance

It is now necessary to study the best performing channel more accurately in order to understand what is the best-case scenario possible, and what advantages it might have in terms of linearity and precision.

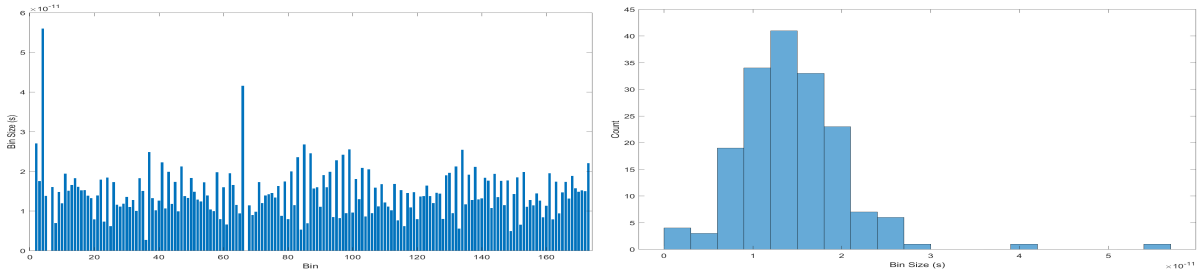


(a) **(b)**

Figure 5.5: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Start of the Best Channel of the 16-Channel Implementation.

Firstly, the width of the bins of the channel for the Start are studied, in Figure 5.5b is seen that the majority of the bins have a width very close to the ideal value with an average bin width of 14.45 ps. Furthermore, the number of ultra-wide bins is below average, and the most significant bin has a width of 51 ps in Tap 3 of the line. The existence of almost no ultra-wide bins contributes to the existence of only 24 taps with a size smaller than 10 ps, which is below average.

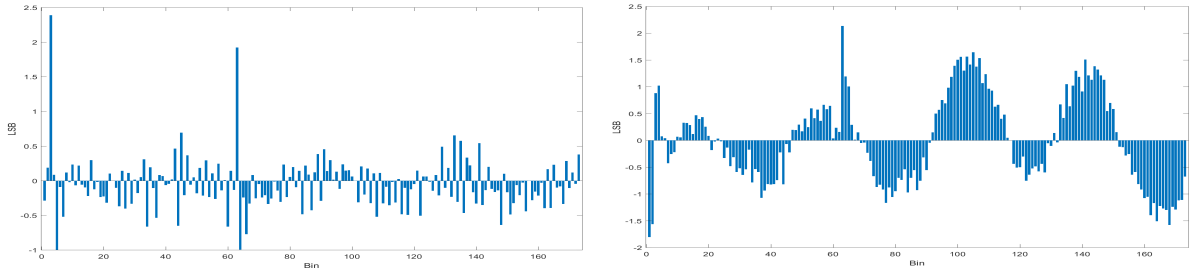
Next off, it is necessary to analyze the bin widths of the Stop of the channel the average bin width is equal to the Start, the widest bin, in this case, is Tap 4 with a width of 56 ps (see Figure 5.6a), which is slightly larger than the widest bin of the Start, and there are only 2 ultra-wide bins. However, the number of bins with less than 10 ps of width is considerably larger than in the Start, and it is even above the average, with 36 bins (see Figure 5.6b). This result is consistent with the results of the measurements of all channels of this implementation, where the average of zero-width bins is higher in the Stop Line that in the Start Line.



(a) **(b)**

Figure 5.6: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Stop of the Best Channel of the 16-Channel Implementation.

After examining the bin widths of both the Start and Stop lines, it is now possible to proceed with the study of the linearity of the delay line. First, the linearity of the Start line is studied. The maximum DNL of the delay line is 2.39 LSB, and the DNL average is close to zero with the value of -0.043 LSB (see Figure 5.7a). In terms of the Integral nonlinearity of the delay line, the positive maximum is 2.13 LSB, and the negative maximum is -1.80 LSB (see Figure 5.7b).

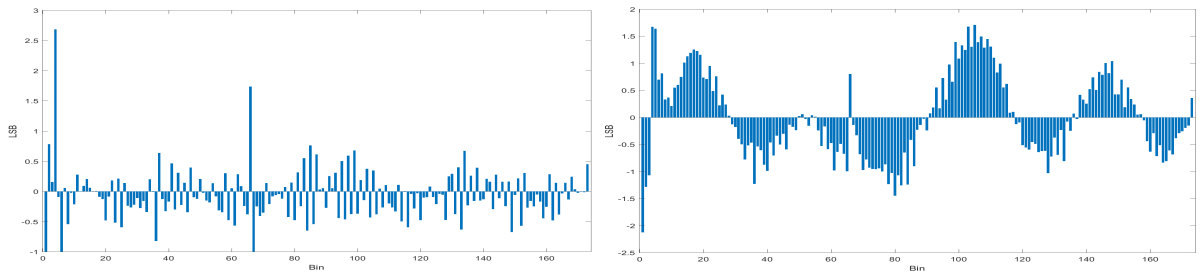


(a) **(b)**

Figure 5.7: DNL **(a)** and INL **(b)** results of the Code Density Test of the Start Line of the the Best Channel of the 16-Channel Implementation.

Afterward, the linearity of the Stop Line must be examined. The maximum DNL of the Stop Line is 2.68 LSB, with an average non-linearity equal to the Start Line (see Figure 5.8a). While in the INL of the line, the negative maximum is -2.12 LSB and the positive maximum of 1.71 LSB.

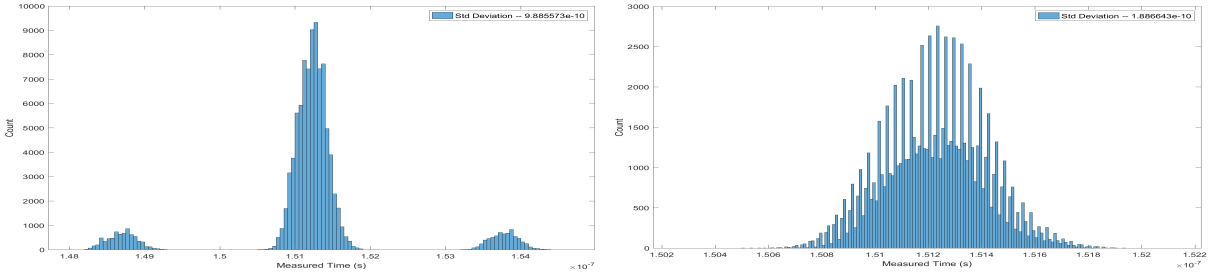
Finally, the precision of the channel is going to be analyzed, both with the influence of the Metastability error to study its influence in the precision of the measurement and without it to focus on the effect of the non-linearities of the delay line in the precision of the measurement and to study the performance of the channel. As it can be seen in Figure 5.9, the Coarse Metastability error significantly deteriorates the precision of the measurements, the precision with the error is 988.57 ps (see Figure 5.9a), while the one without the error is far more precise with a precision of 188.64 ps. This means that if the Coarse Counter



(a) Bin Size Start. **(b)** Bin Size Start.

Figure 5.8: DNL **(a)** and INL **(b)** results of the Code Density Test of the Stop Line of the the Best Channel of the 16-Channel Implementation.

Metastability error is corrected, it is possible to obtain much more precise TDC measurements that the ones currently available.



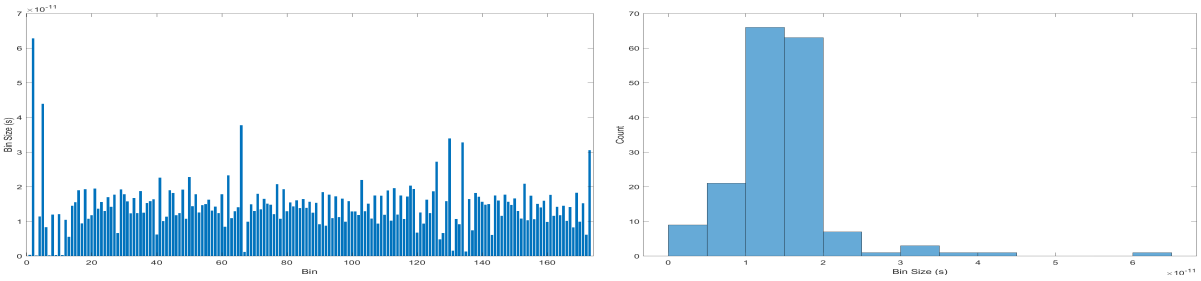
(a) **(b)**

Figure 5.9: Measurement Histogram and Precision of the Best Channel of 16-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

5.2.1.3 Channel with the worst performance

It is equally important to analyze the worst-performing channel, to understand how its poor performance affects the precision and reliability of the measurements made by the Acquisition System in the worst case possible.

The bin width of the Start is analyzed in the first place (see Figure 5.10), the vast majority of the bins have a width near the ideal size (see Figure 5.10b), with an average size of 14.45 ps. The widest tap is the second tap with a size of 62.82 ps, with a bigger size than the widest tap of the highest performance channel. The number of ultra-wide bins is also higher with 6 bins. The number of bins with a size smaller than 10 ps is of 30, which is inside the average, and as seen in Figure 5.10a the first bins of the line have a smaller width than the ones in the middle of the line. This distribution will influence the linearity of the line as it will be further analyzed.

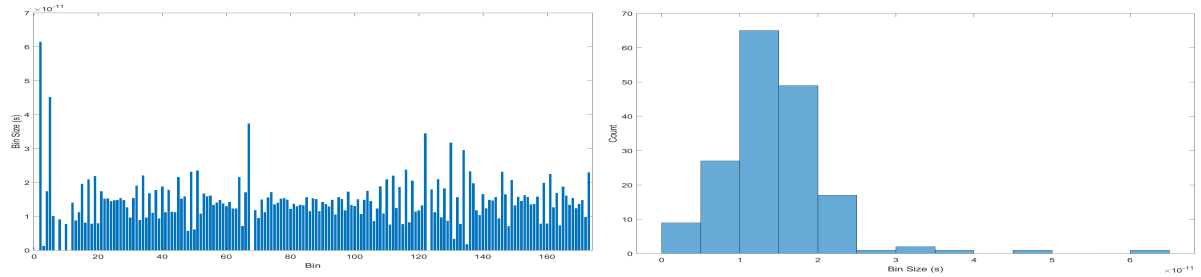


(a) Bin Size Start. **(b)** Bin Size Start.

Figure 5.10: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Start of the Worst Channel of the 16-Channel Implementation.

In the Stop Line, the bin width distribution substantially changes, nevertheless the average bin size is the same. The widest bin, in this case, is the second bin with a width of 61.45 ps (see Figure 5.11a), which is superior to the widest bin of the Start Line. Furthermore the number of ultra-wide bins is also higher with 5 bins. The number of bins with a size smaller than 10 ps is 36 (see Figure 5.11b), which is higher than the average number.

The linearity of this channel can be observed in Figure 5.12, where the non-linearities of the Start Line are displayed. The Differential nonlinearity of the taps of the line are comparable with the value of the best performing channel. The largest DNL of the current channel is 3.13 LSB (see Figure 5.12a), which is a higher value of DNL than in the best channel.

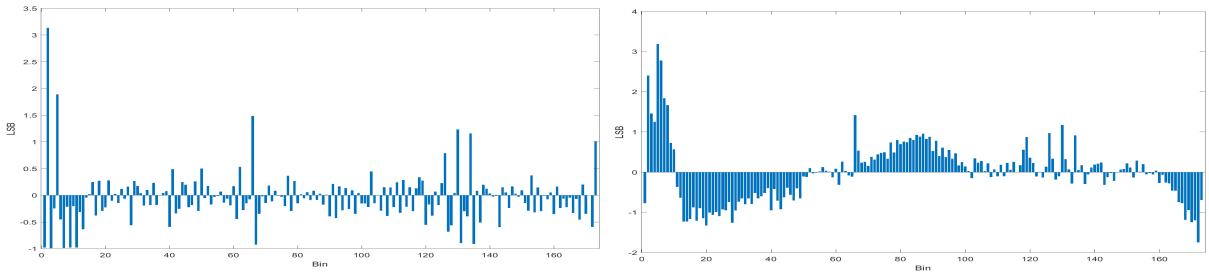


(a) **(b)**

Figure 5.11: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Stop of the Worst Channel of the 16-Channel Implementation.

However, the INL of the current channel, which has a positive maximum of 3.18 LSB and a negative maximum of -1.75 LSB (see Figure 5.12b), is higher than the best performing channel of the current implementation.

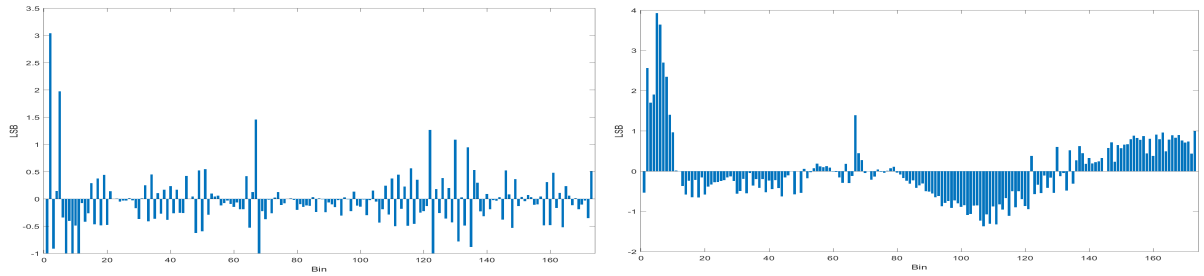
In the Stop Line (see Figure 5.13), the poor performance of the channel is also visible, even though it manifests itself differently. The maximum DNL of the Stop Line is 3.04 LSB (see Figure 5.13a), which is superior to the maximum value of the best performing channel. The INL of this channel has a positive



(a) **(b)**

Figure 5.12: DNL **(a)** and INL **(b)** results of the Code Density Test of the Start Line of the the Worst Channel of the 16-Channel Implementation.

maximum of 3.93 LSB and a negative maximum of -1.37 LSB (seen in Figure 5.13b), which is a significantly higher value when compared with Stop Line of the best performing channel.



(a) **(b)**

Figure 5.13: DNL **(a)** and INL **(b)** results of the Code Density Test of the Stop Line of the the Worst Channel of the 16-Channel Implementation.

Lastly, the precision of the channel must be studied, both with the influence of the Metastability error and without it (see Figure 5.14). The measurements of this channel with the Metastability error have a precision of 1.27 ns. While if the error is not taken into account, it becomes more precise with a performance value of 232.17 ps. The influence of the Coarse Metastability error in the performance observed in this channel shows how its influence can deteriorate the precision of the channel and the importance of developing a Synchronizer that mitigates the error

5.2.2 Results for 8 channel implementation

Now that the 16-Channel Implementation was analyzed, it is possible to analyze and compare the other implementations with it, in order to see the influence of the implemented number of channels in the overall performance of the Acquisition System. The first implementation to be compared is the 8-Channel Implementation.

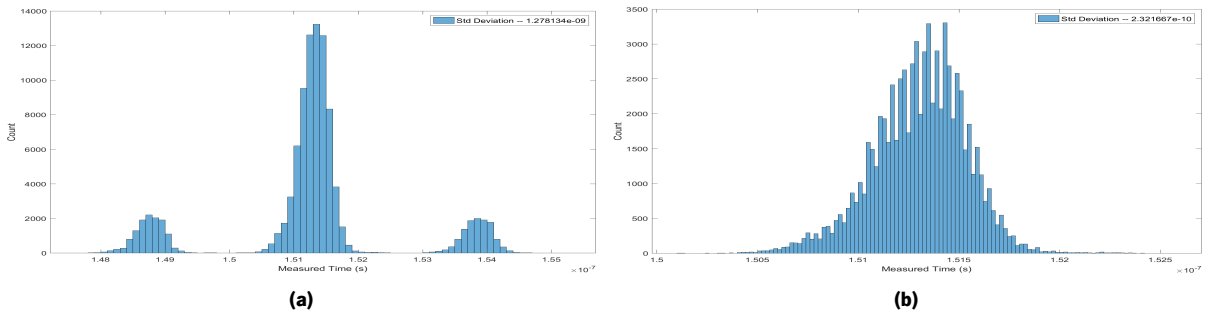


Figure 5.14: Measurement Histogram and Precision of the Worst Channel of 16-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

The vast majority of the Bins of the Start Line have a width very close to the ideal size of 15.2 ps (see Figure 5.15b), the average size of the taps is 14.45 ps which is equal to the average size of the 16-Channel implementation. As in the 16-Channel in every channel, there are at least one or more ultra-wide bins (see Figure 5.15a), with the widest one being tap 67 of channel 1 with 71.15 ps, which is slightly bigger than the one of the 16-Channel implementation, and the average number of ultra-wide bins is also higher with 3.25 bins per channel.

Furthermore, the problem with zero-width or very small bins is more significant in this implementation, which has an average of 35.25 bins with less than 10 ps per channel. It is a higher average than the 16-Channel Implementation, which in comparison leads to worse performance.

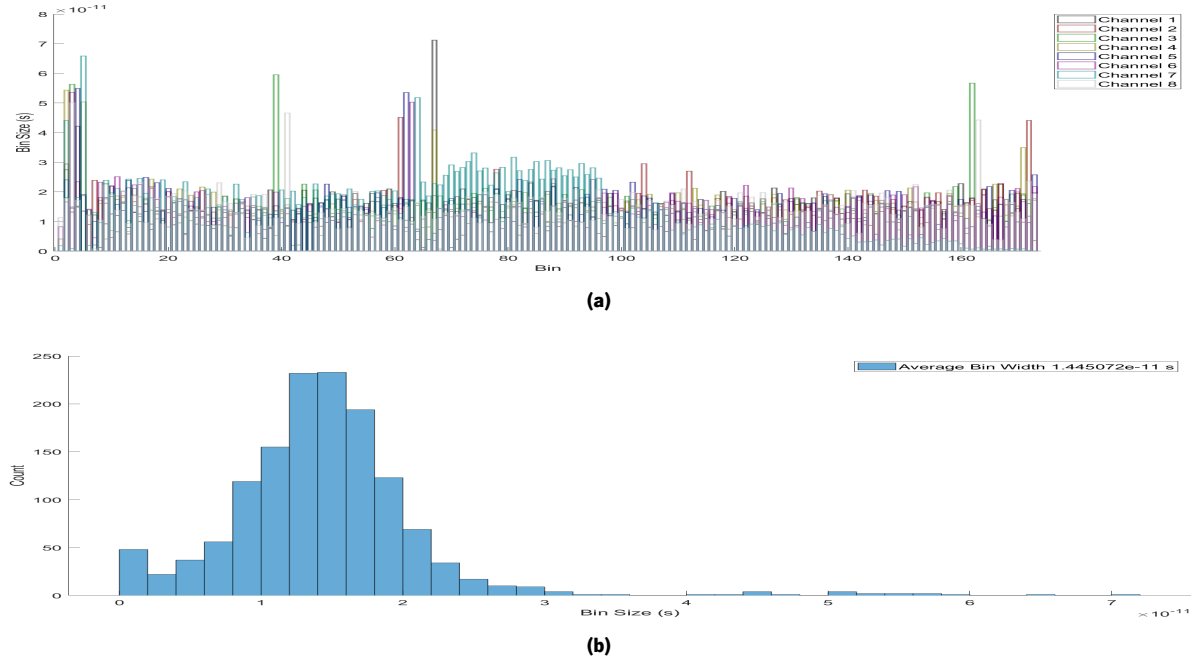
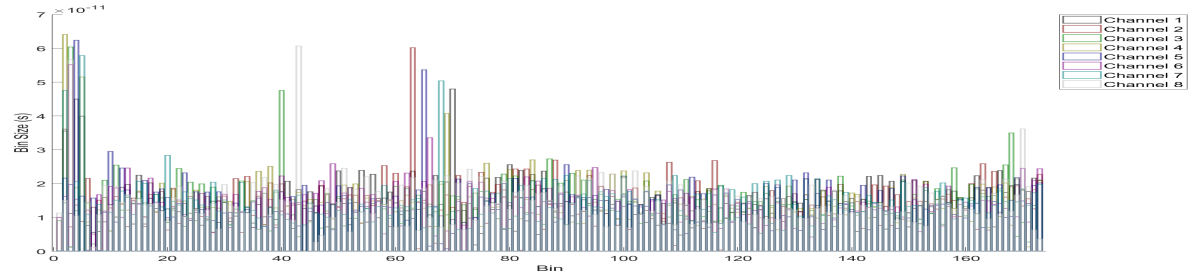
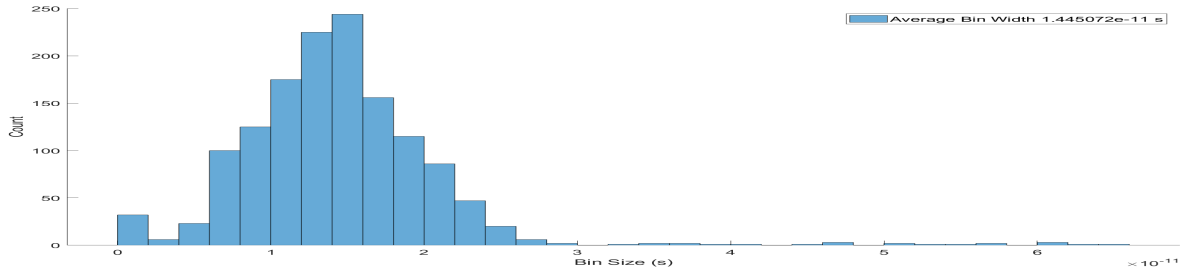


Figure 5.15: Measured Bin Width **(a)** for each channel and **(b)** the bin width Histogram of the Start of all the bins of the 8-Channel Implementation.



(a) Bin Size Stop.



(b) [Histogram Bin Size Stop.

Figure 5.16: Measured Bin Width **(a)** for each channel and **(b)** the bin width Histogram of the Stop of all the bins of the 8-Channel Implementation.

In the Stop Line, the bin width distribution is similar to the Start Line, the majority of Bins has width close to the ideal (see Figure 5.16b), and the average width is also of 14.45 ps. In conformity to what was observed in the 16-Channel Implementation, the widest Bin of all channels is smaller than the one of the Start Line, with a value of 64.10 ps in the tap 2 of channel 4. The average number of ultra-wide bins is also smaller than both the Start Line and the Stop Line of the 16-Channel Implementation with 2.75 bins per channel.

Moreover, the average number of bins with a width smaller than 10 ps is 35.75 per channel, which is higher than in the 16-Channel Implementation but is similar to the value for the Start Line.

The measurement precision of this implementation is displayed in Figure 5.17. The Hit used was identical to the one used in the 16-Channel Implementation. Similar to the last implementation, the majority of measurements are close to the real size of the Hit, except on Channel 7, where the non-linearities of the Line deeply affect the measurement. The average measurement is 151.29 ns. In Figure 5.17a, it can be seen the effect of the Coarse Counter Metastability in the measurements, where there are two other aggregations of measurements, one with more 2.5 ns than expected and another with less 2.5 ns. The precision of the measurements with the Metastability error varies from 929.02 ps to 378.70 ps and an average value of 594.55 ps, meaning that the precision of the measurement done in these channels with the Metastability error is better than the one made by the 16-Channel implementation.

Finally, the performance of the channels is examined, this value varies from the 343.09 ps of channel 7 to the 199.73 ps of channel 1, with an average precision of 229.08 ps, meaning that the precision of measurement of the current implementation is worse than the 16-Channel Implementation.

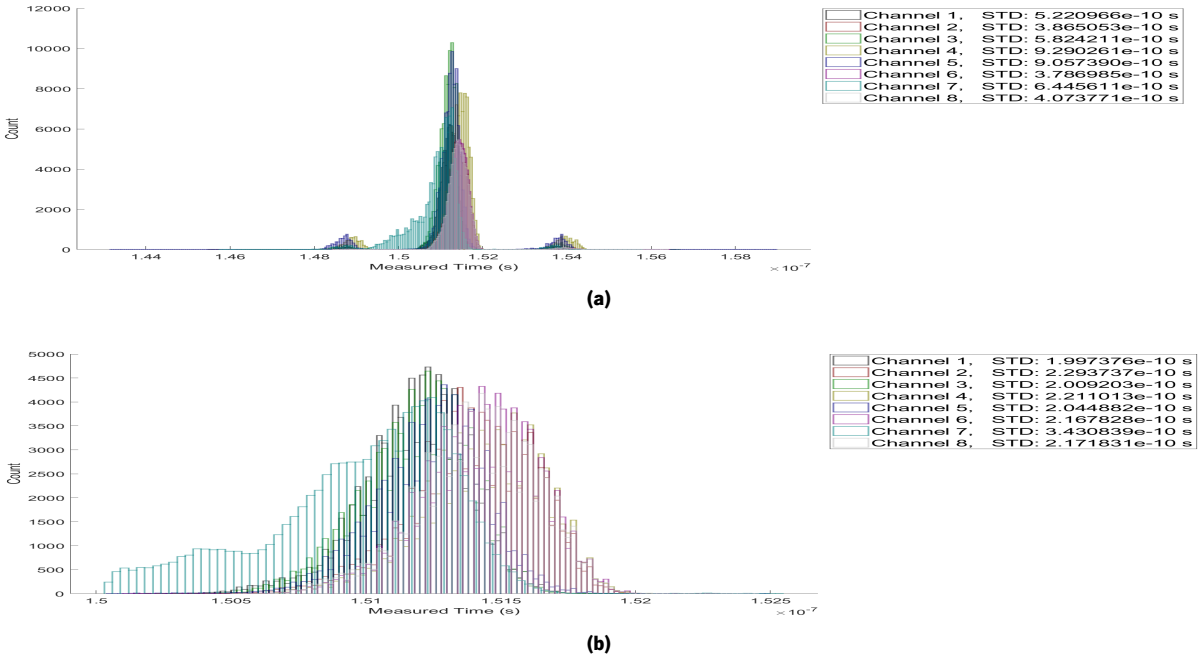


Figure 5.17: Measurement Histogram and Precision, of all channels of the 8-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

5.2.2.1 Resource Utilization

As seen in Table 5.2, with eight channels implemented, the resources utilized the use of LUTs, FFs, and BRAMs diminished while the utilization of the other resources remains equal. The resource with the highest utilization is the LUT, where 14.84 percent where used, all other resources report a utilization under 12.50 percent.

Resource	Utilization	Available	Utilization
LUT	34202	230400	14.84
LUTRAM	523	191760	0.51
FF	23843	460800	5.17
BRAM	24	312	7.69
IO	6	360	1.67
BUFG	12	544	2.21
MMCM	1	8	12.50

Table 5.2: Resource Utilization of the 8 Channel Implementation

5.2.2.2 Channel with the best performance

The best performing channel must be studied and compared with the best channel of the 16-Channel Implementation, to see if there is any influence of the number of implemented channels in the performance of the best-case scenario.

In the first place, the Bin width distribution of the Start Line is going to be examined (see Figure 5.18). The average width of the Start Line is 14.45 ps, which is equal to the value of the 16-Channel Implementation (see Figure 5.18b). Additionally, the current channel has 31 taps under 10 ps of width, which is below the average of the current implementation, although superior to the number for the Start line 16-Channel implementation. The number of ultra-wide bins is reduced and below the implementation average, and the biggest tap has a size of 71.12 ps (see Figure 5.18a).

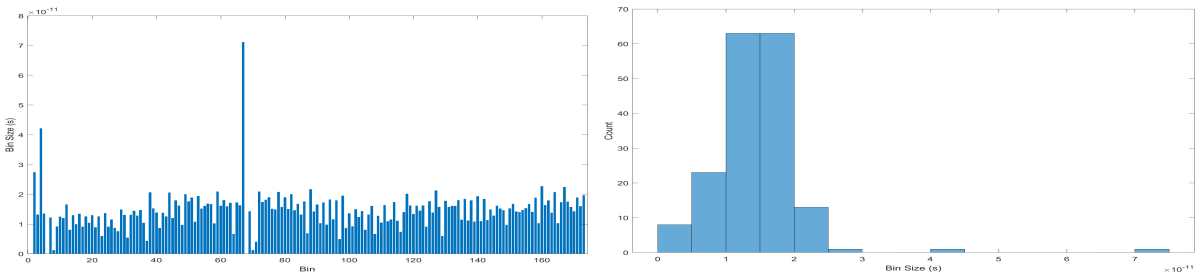
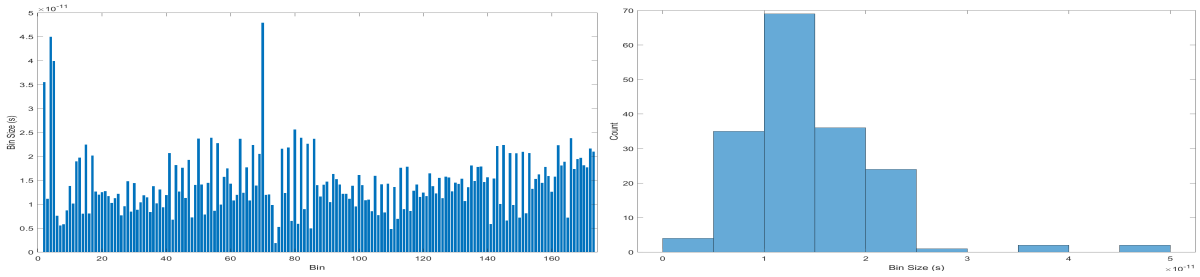


Figure 5.18: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Start of the Best Channel of the 8-Channel Implementation.

Subsequently, the Bin width distribution of the Stop Line is evaluated (see Figure 5.19). The average width of the bins is equal to the values of the Start Line, as it was expected. The widest bin of the line has a size of 47.92 ps, which is considerably smaller than the Start, contrasting with what happens in the 16-Channel Implementation, and it is also smaller than the widest value of the bin of the Stop on the 16-Channel Implementation. The number of ultra-wide bins is 4. Likewise to what happened in the 16-Channel implementation, the number of taps with less than 10 ps increases in the Stop Line with 39 taps (see Figure 5.19b), which is a little higher than the average value for the current implementation.

The maximum value of the DNL is 3.68 LSB (see Figure 5.20a), the average value of the DNL is equal to the value of the 16-Channel Implementation as expected since the average width of the taps in both cases is equal. The Integral nonlinearity of the Line has a positive maximum of 5.29 LSB and the negative maximum of -3.63 LSB a(see Figure 5.20b).

Afterward, the linearity of the Stop Line is analyzed. In terms of Differential nonlinearity, the maximum value is 2.15 LSB (see Figure 5.21a), which is smaller than the maximum DNL of both the Start Line and

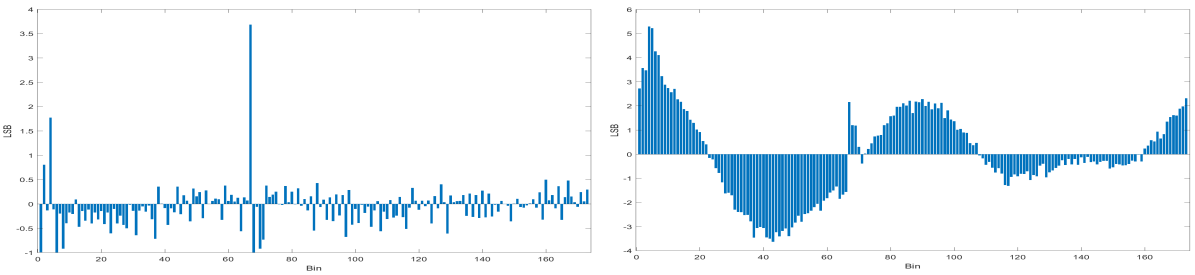


(a) **(b)**

Figure 5.19: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Stop of the Best Channel of the 8-Channel Implementation.

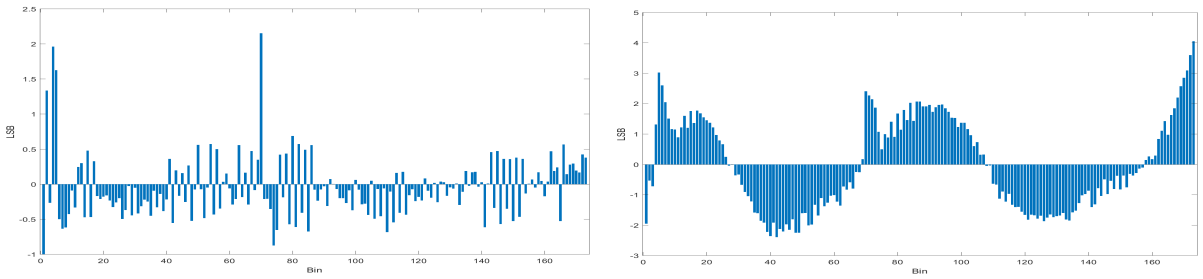
the Stop Line of the 16-Channel Implementation, and the average DNL is equal to the one seen in the other lines already analyzed. In terms of Integral nonlinearity, the positive maximum is of 4.05 LSB, and the negative maximum is of -2.40 LSB. The linearity of both the Start and Stop Line is considerably worse than the ones of the 16-Channel Implementation.

To conclude the analysis of this channel, the measurement precision is studied. First off, the precision of the channel under the effect of the Metastability error is studied. The precision of the channel is of 522.10 ps (see Figure 5.23a). However, if the influence Metastability error is removed, then the precision increments to 199.73 ps (see Figure 5.22b). Compared with the 16-Channel Implementation, it can be seen that the influence of the Metastability error is less noticeable in the current case due to having a better precision, however, when this error is eliminated, the channel of the 16-Channel implementation proves to be more precise with a better than the current channel. So it can be concluded that the reduction of the number of implemented channels by half without adjusting the placement of the PBlocks did not improve the performance of the channels. Therefore, to achieve better performance from reducing the number of channels, it is necessary to analyze and change the placement of the PBlocks in a way that ensures the best performance of the channels in the current implementation.



(a) **(b)**

Figure 5.20: DNL **(a)** and INL **(b)** results of the Code Density Test of the Start Line of the the Best Channel of the 8-Channel Implementation.



(a) Bin Size Start. **(b)** Bin Size Start.

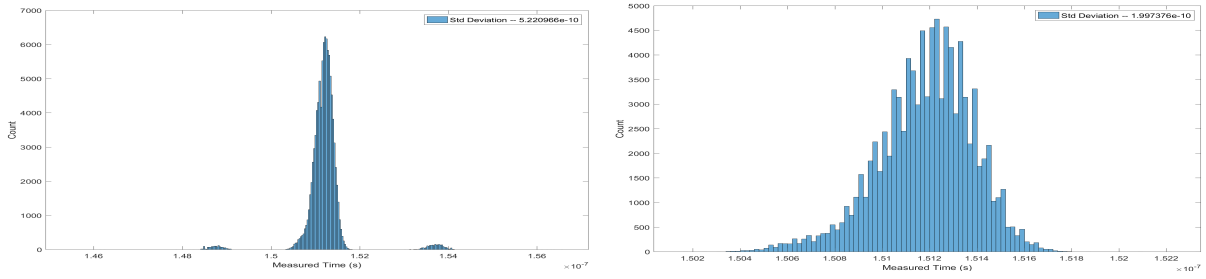
Figure 5.21: DNL **(a)** and INL **(b)** results of the Code Density Test of the Stop Line of the the Best Channel of the 8-Channel Implementation.

5.2.2.3 Channel with the worst performance

Now it is necessary to study the performance of the channel with the least performance, to conclude if the reduction of the number of channels elevates the performance of the worst performing channel.

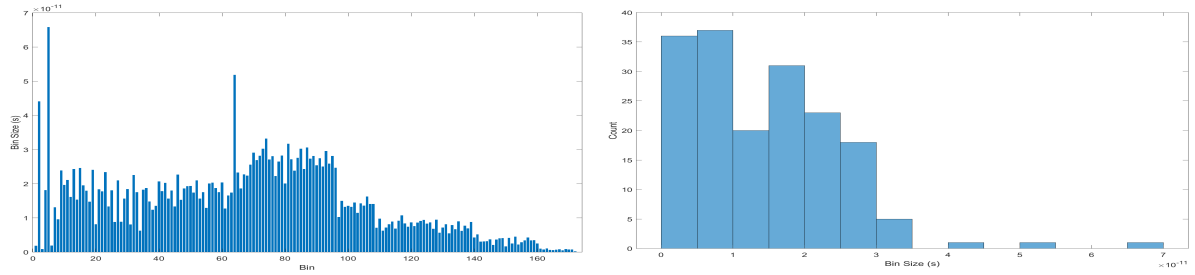
The Start Line Bin width distribution will now be analyzed. As seen in Figure 5.23b, there is a significant number of Bins that have a size under 10 ps, contrarily to what happens in all other channels analyzed until now. The number of small bins is 73, which is substantially higher than the average, and it is the cause of the poor performance of the measurements of the channel. The number of ultra-wide bins is 8, which is higher than in the other cases and could explain the higher number of small bins. The widest bin having 65.85 ps (see Figure 5.23a).

Contrarily to the Start Line, the Bin width distribution of the Stop Line is similar to the ones seen in the other channels. The vast majority of bins as a width close to ideal, it has a below average number of bins under 10 ps, with only 31 Bins (see Figure 5.24b), which means that the poor performance of this channel is limited to the Start and not something connected to the delay Line. The widest bin of the Stop is 57.85 ps (see Figure 5.24a), which is smaller than the widest Bin of the Stop Line of the worst performing



(a) **(b)**

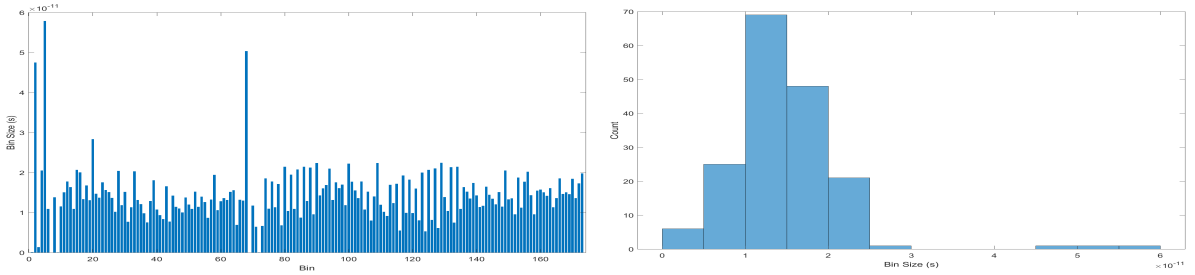
Figure 5.22: Measurement Histogram and Precision of the Best Channel of 8-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.



(a) **(b)**

Figure 5.23: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Start of the Worst Channel of the 8-Channel Implementation.

channel of the 16-Channel Implementation, and the number of ultra-wide bins is the same as the Stop Line of the best channel.

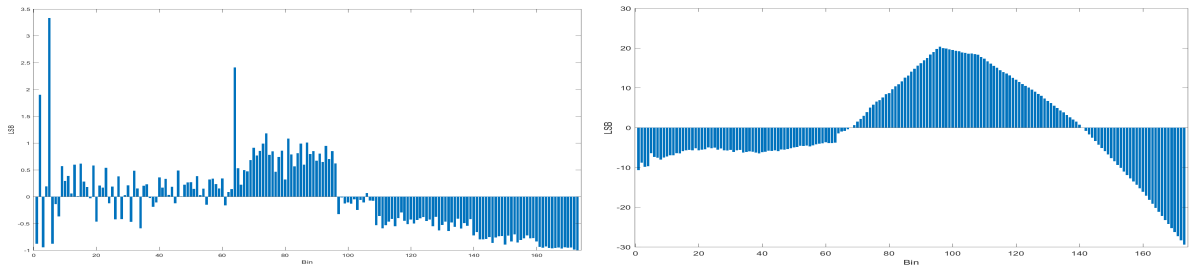


(a) **(b)**

Figure 5.24: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Stop of the Worst Channel of the 8-Channel Implementation.

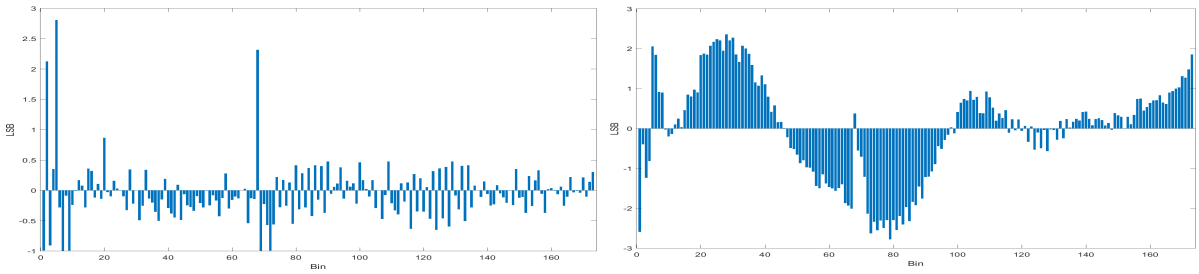
The linearity of the Start Line of the channel can be seen in Figure 5.25, where it can be observed that the high number of small bins of the channel reduces the linearity of the channel. The maximum DNL of the current Start Line is 3.33 LSB (see Figure 5.25a). However, it is in the INL of the channel that the poor linearity of the channel is more noticeable. The value of the INL varies from a positive maximum of 20.60 LSB and a negative maximum of -29.40 LSB (see Figure 5.25b). These values are far superior to the ones seen in the 16-Channel Implementation and reflect the poor bin width distribution of the delay Line. It can be concluded that the measurements of the Start Line are too nonlinear to be reliable.

Contrasting with the results of the Start Line, the linearity of the Stop Line is better than the performance of the Stop Line of the worst channel of the 16-Channel Implementation. Its linearity values are even comparable to the linearity values of the Stop Line of the best channel of the current implementation. The maximum value of the DNL of this channel is 2.80 LSB (see Figure 5.26a). Furthermore, the INL varies from the positive maximum of 2.36 LSB and the negative maximum of -2.78 LSB (see Figure 5.26b).



(a) **(b)**

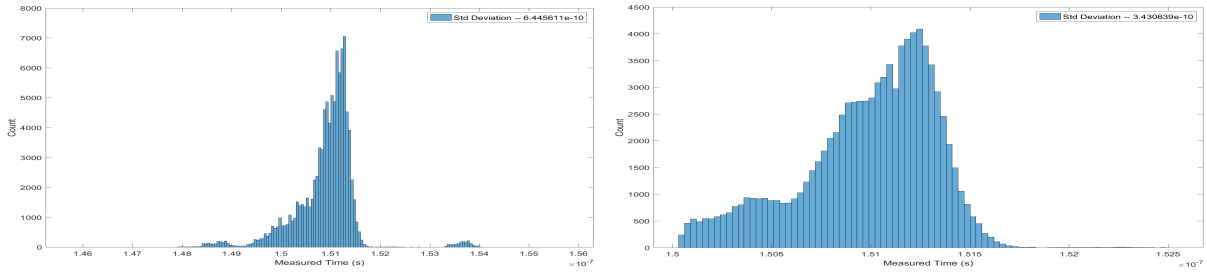
Figure 5.25: DNL **(a)** and INL **(b)** results of the Code Density Test of the Start Line of the the Worst Channel of the 8-Channel Implementation.



(a) **(b)**

Figure 5.26: DNL **(a)** and INL **(b)** results of the Code Density Test of the Stop Line of the the Worst Channel of the 8-Channel Implementation.

In terms of precision, the Metastability error does not influence this channel as much as it did in the 16-Channel Implementation. The measurement precision has a value of 644.56 ps (see Figure 5.27a), which is more precise that the worst channel of the 16-Channel Implementation. However, when the influence of the Coarse error is not taken into account, the precision falls short of the 16-Channel Implementation, with a precision of 343.08 ps (see Figure 5.27b). Reflecting the poor performance of the measurements of this channel, even though the Stop Line has a good performance, the non-linearities of the Start Line deteriorate the precision to much.



(a) All channels measurements and precision. **(b)**

Figure 5.27: Measurement Histogram and Precision of the Worst Channel of 16-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

5.2.3 Results for 4 channel implementation

In order to further test the influence of the number of implemented channels in the performance of the Acquisition System, the 4-Channel implementation will now be analyzed.

As in the other implementations, firstly, the distribution of the Bin width of the Start Line of every channel is studied. As expected, the majority of the Bins have a size close to the ideal bin size, and the average bin size is the same as the ones registered in the other implementations (see Figure 5.28b). There is an average of 3.75 ultra-wide bins, which is a higher value than the average of the 16-Channel Implementation, and the widest bin of the current implementation has a size of 75.38 ps in Bin 2 of channel 2 (see Figure 5.28a).

Additionally, the average number of bin with a size smaller than 10 ps is 32 per channel (see Figure 5.28b). Although the value is close to the average value of the 16-Channel Implementation, it is still higher, which means it could deteriorate the performance of the Line.

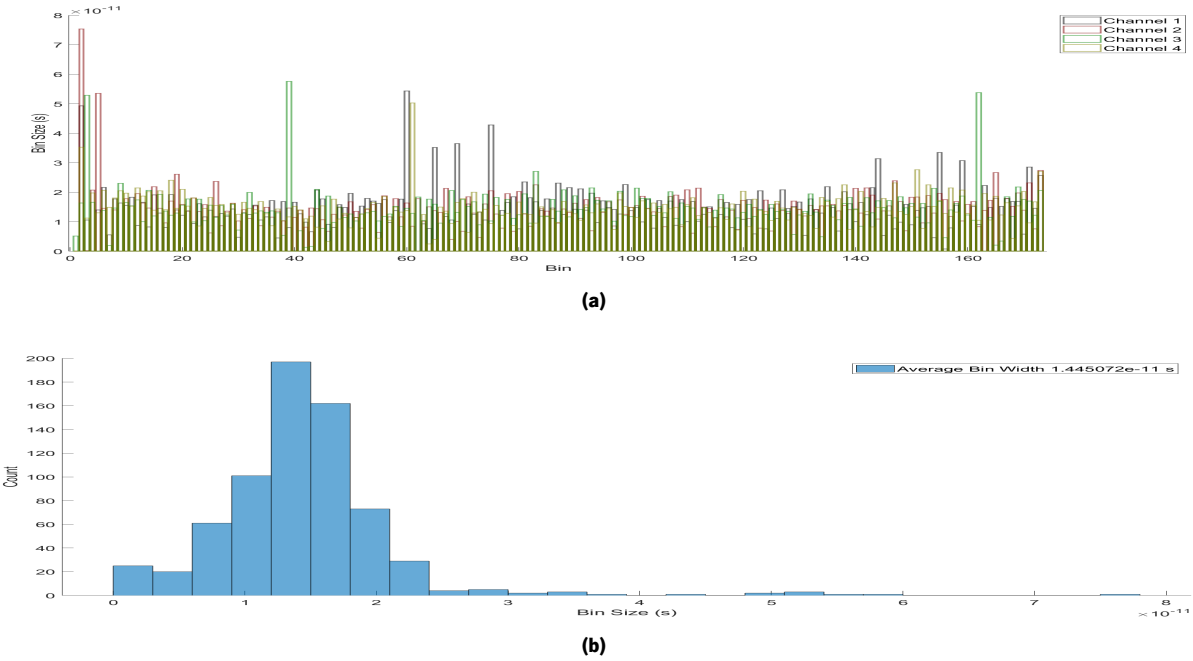


Figure 5.28: Measured Bin Width **(a)** for each channel and **(b)** the bin width Histogram of the Start of all the bins of the 4-Channel Implementation.

Analyzing the Bin width distribution of the Stop Line, it can be seen that the Bin width distribution does not differ much from the Start Line, the large part of the Bins have a width close to ideal, and the average size of the Taps is equal to the Start (see Figure 5.29b). Moreover, there is an average of 3.25 ultra-wide bins per channel, the most substantial Bin in every channel of the Stop Line has the size of 65.10 ps in

the second Tap of the first channel (see Figure 5.29a), which is smaller than the widest one of the Start Line.

Nonetheless, and in Line with what happened in the other implementations, the average number of Taps with less than 10 ps of size is higher in the Stop Line, with a value of 36.75.

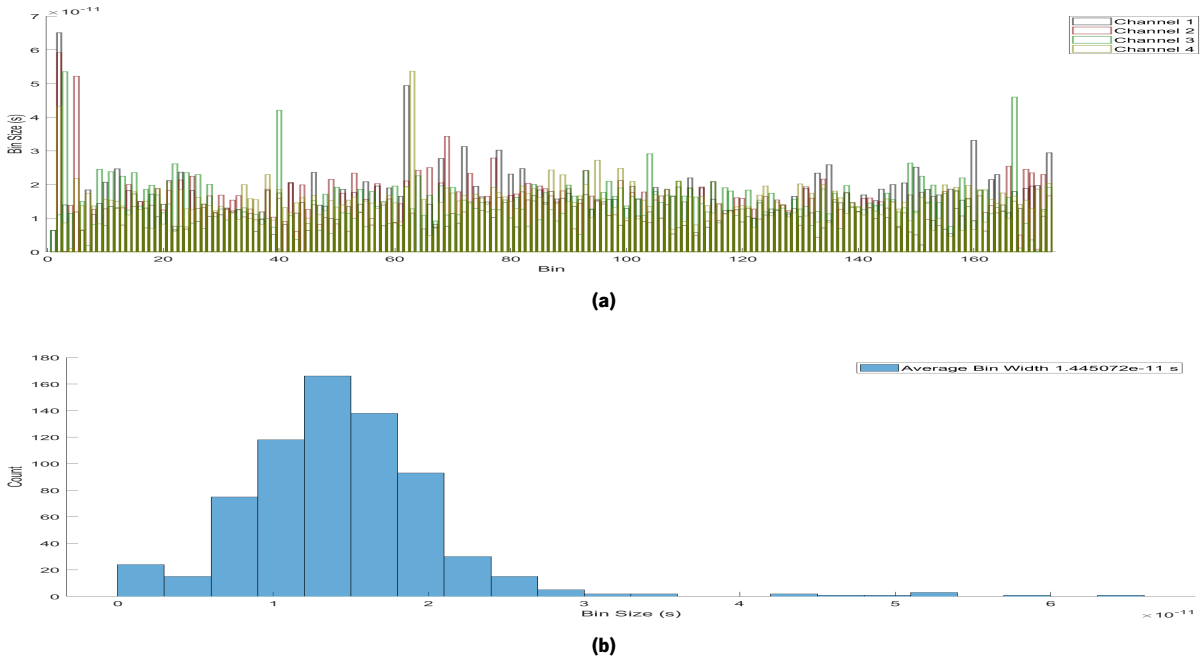


Figure 5.29: Measured Bin Width **(a)** for each channel and **(b)** the bin width Histogram of the Stop of all the bins of the 4-Channel Implementation.

The results of the measurements and their precision, with and without the Coarse Metastability error, are displayed in Figure 5.30. As in the 16-Channel Implementation, the most measurements are close to the real size of the Hit signals, with an average measurement value of 151.35 ns, which is very close to the average measurement value obtained by the 16-Channel Implementation. The precision of the measurement ranges from 1.10 ns to 359.10 ps, the average precision is of 682.54 ps. However, if the Coarse error is removed, the precision of the measurements ranges from 219.04 ps to 191.44 ps, with an average value of 206.91 ps, meaning that the average precision of this implementation is better than the 16-Channel Implementation.

5.2.3.1 Resource Utilization

As seen in Table 5.3, the reduction of the number of implements channels diminishes the resource utilization of LUTs, BRAMs, and FFs. The resource with the highest utilization is the MCMM, where 12.5 percent where used, all other resources report a utilization under 8 percent.

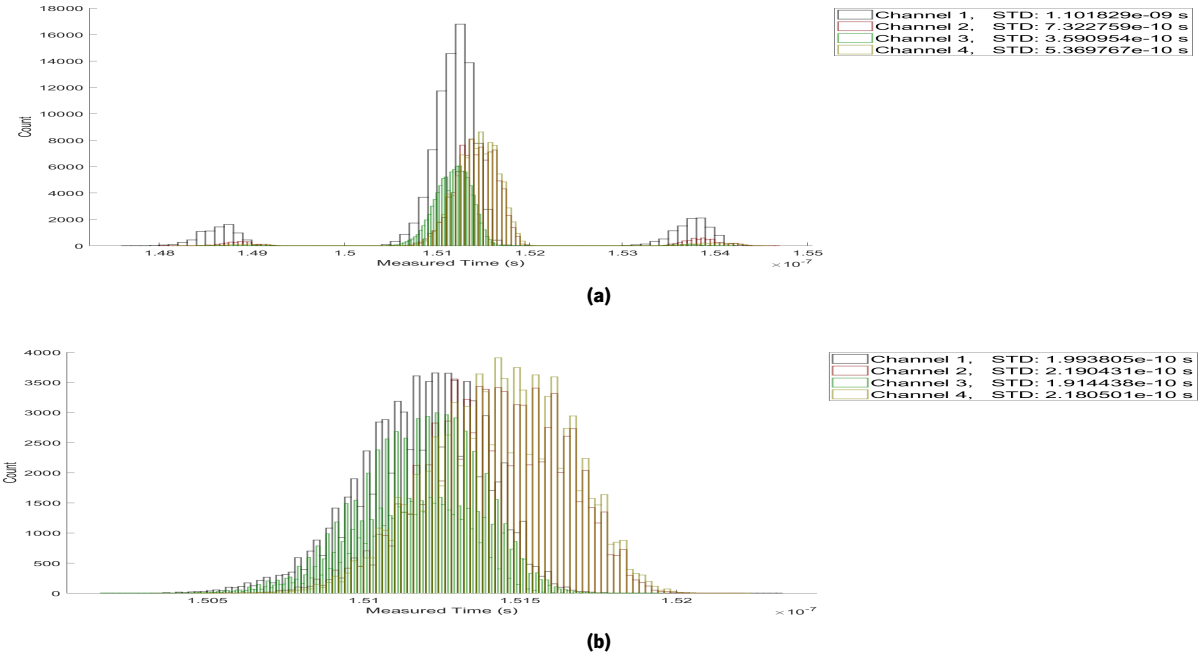


Figure 5.30: Measurement Histogram and Precision, of all channels of the 4-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

Resource	Utilization	Available	Utilization
LUT	18.419	230400	7.99
LUTRAM	523	191760	0.51
FF	13485	460800	2.93
BRAM	12	312	3.85
IO	6	360	1.67
BUFG	8	544	1.47
MMCM	1	8	12.50

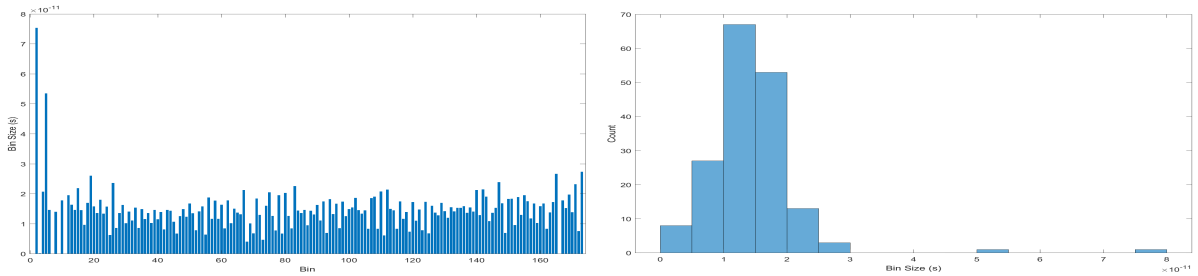
Table 5.3: Resource Utilization of the 4 Channel Implementation

5.2.3.2 Channel with the best performance

The decrease in the number of channels implemented influences the performance of all implemented channels. Therefore it is necessary to analyze the best performing channel and compare it to the best channel from the 16-Channel Implementation to examine the influence of this decrease in the best-case scenario of the current implementation.

The first thing to be analyzed is the Bin width distribution of the Start Line (see Figure 5.31). The average width value is, as expected, of 14.45 ps. Furthermore, there are 25 Bin with a size below 10 ps (see Figure 5.31b), which is a value below the average of this implementation, and close to the value registered in the Start Line of the 16-Channel Implementation. The widest bin of the line has a value of

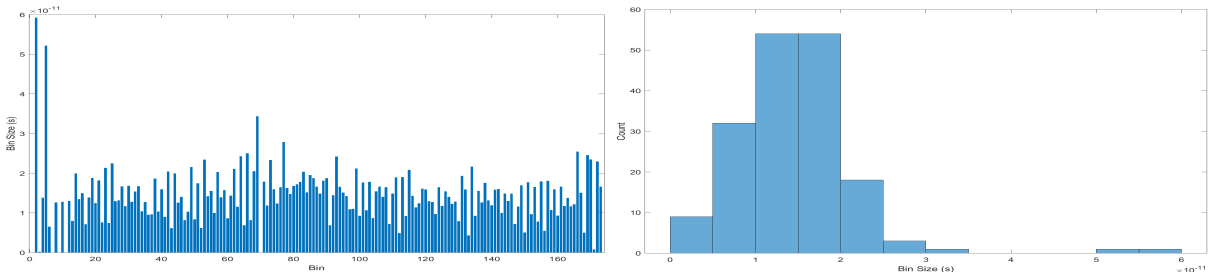
57.60 ps (see Figure 5.31a), which is bigger than the widest bin of the Start Line of the best channel of the 16-Channel Implementation. Furthermore, the number of ultra-wide bins is 3, in comparison with the average for the current implementation, it is slightly lower.



(a) **(b)**

Figure 5.31: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Start of the Best Channel of the 4-Channel Implementation.

Subsequently, the Stop Line of the channel must be studied (see Figure 5.32). Firstly the average size of the Bins is equal to the one registered until now in the other cases (see Figure 5.32b). The widest bin of the Stop Line has a value of 53.53 ps (see Figure 5.32a), which is smaller than the widest bin of the Start Line and the widest bin of the Stop of the same channel of the 16-Channel Implementation. The number of ultra-wide bins is 3, a smaller number than in the Start line. The number of Taps with a size smaller than 10 ps is even higher than in the Start Line, with 38 Bins, which is higher than the average value for the current implementation.

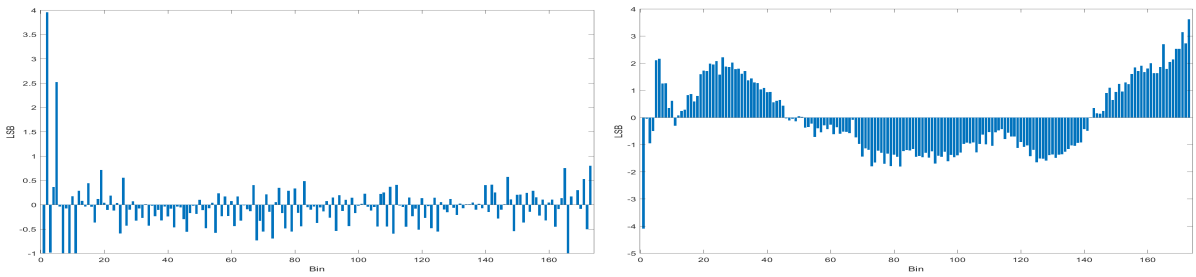


(a) **(b)**

Figure 5.32: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Stop of the Best Channel of the 4-Channel Implementation.

After examining the Bin width distribution of both the Start and Stop Line, the linearity of the Start Line is analyzed. In terms of DNL, the maximum value is of 2.79 LSB, and an average value equal to the 16-Channel implementation (see Figure 5.33a). The INL has a positive maximum of 2.70 LSB and a negative maximum -2.23 LSB (see Figure 5.33b), meaning that the non-linearity of the Start Line is higher

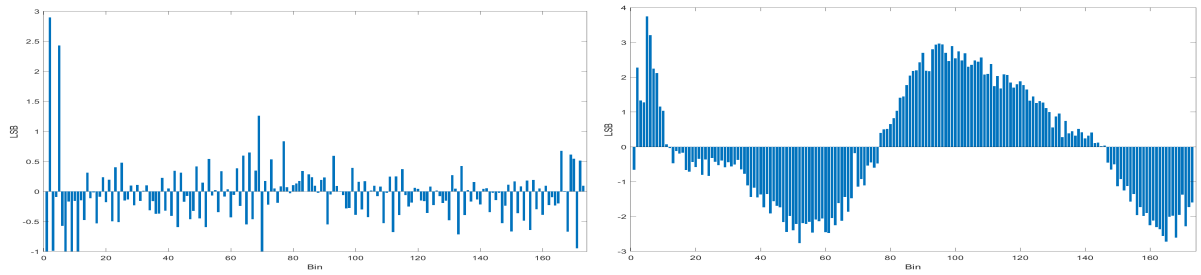
than what was observed in the 16-Channel Implementation. So it can be concluded that the reduction of the number of channels did not increment the linearity of the Start Line.



(a) **(b)**

Figure 5.33: DNL **(a)** and INL **(b)** results of the Code Density Test of the Start Line of the the Best Channel of the 4-Channel Implementation.

Next, the linearity of the Stop Line is examined, the maximum Differential nonlinearity has a value of 2.52 LSB, which is smaller than the maximum DNL of the Stop Line of the best performing channel of the 16-Channel Implementation. The INL in the Stop Line has a positive maximum of 3.82 LSB and a negative maximum of -3.43 LSB, which is a higher non-linearity that the best performing channel of the 16-Channel Implementation.

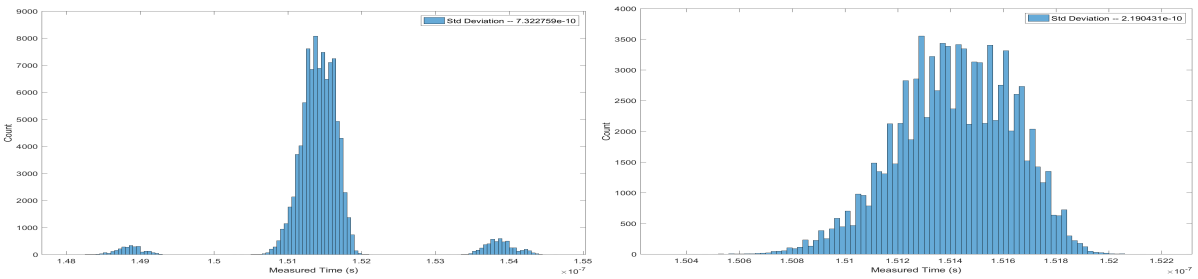


(a) Bin Size Start. **(b)** Bin Size Start.

Figure 5.34: DNL **(a)** and INL **(b)** results of the Code Density Test of the Stop Line of the the Best Channel of the 4-Channel Implementation.

After analyzing the Bin width distribution and the non-linearities of both lines, the precision of the measurements of the channel can finally be examined. The measurement precision of the current channel with the Coarse error is 359.10 ps, which is the worst precision of the current implementation and is not far from the worst precision of the 16-Channel Implementation. However, if the Coarse error is ignored, the precision of the current channel ascends to a value 191.44 ps, which is close to the precision of the best performing channel of the 16-Channel Implementation. So it can be concluded that even though the linearities of the delay line of the channel are considerably worse, the precision of the measurements is not necessarily affected by it. Thus, the Metastability error must not only affect the Coarse Counter

measurement but also affect the acquisition of the Thermometer Code in the TDL. The Metastability error causes the Store Stage of the TDL to occasionally acquire the status of the delay line a clock after the Start or Stop of the signal, meaning it captures an invalid Thermometer Code, which has the effect of lowering the value of the LSB from 15.2 ps to a 14.45 ps.



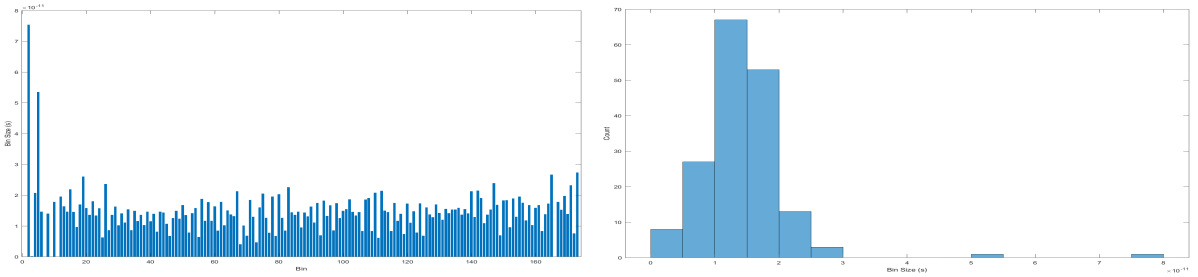
(a) All channels measurements and precision. **(b)**

Figure 5.35: Measurement Histogram and Precision of the Best Channel of 4-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

5.2.3.3 Channel with worst performance

Now the influence of the reduction of the number of implemented channels in the performance of the worst performing channel must be analyzed.

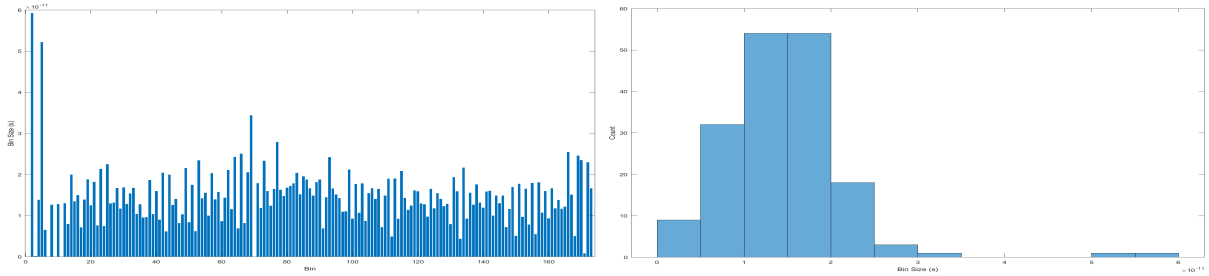
The Start Line Bin width distribution will now be examined. The vast majority of the Bins has a width close to the ideal value, and the average size is 14.45 ps as expected. The widest Bin of the Line has a size of 75.38 ps (see Figure 5.36a), which is bigger than the size of the widest Tap of the worst channel of the 16-Channel Implementation. Moreover, the line has 2 ultra-wide bins, a below average value. The number of Taps with less than 10 ps is of 35 taps (see Figure 5.36b), which is close to the average for the implementation.



(a) Bin Size Start. **(b)** Bin Size Start.

Figure 5.36: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Start of the Worst Channel of the 4-Channel Implementation.

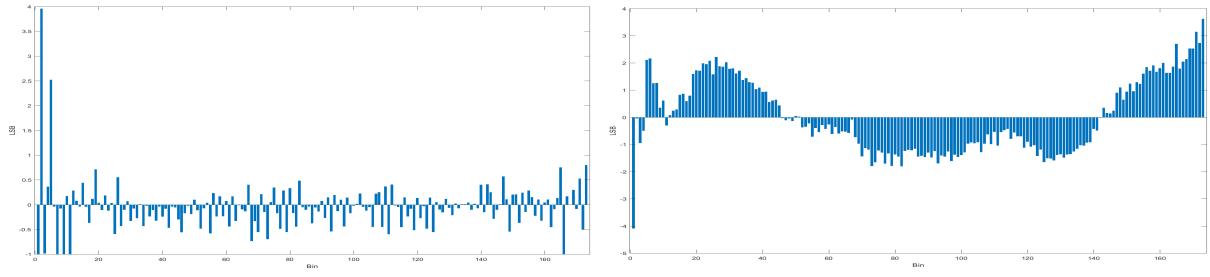
The Tap size distribution of the Stop Line is similar to what was observed in the other channels, with the majority of values being close to the ideal value. However, the number of small bins is higher than the average value for the implementation with 41 Taps below 10 ps (see Figure 5.37b), but it is close to the value of the 16-Channel Implementation. Furthermore, the widest bin of the Stop Line has a size of 59.22 ps (see Figure 5.37a), which is very close to the value of the 16-Channel Implementation, and the number of ultra-wide bins is below average.



(a) **(b)**

Figure 5.37: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Stop of the Worst Channel of the 4-Channel Implementation.

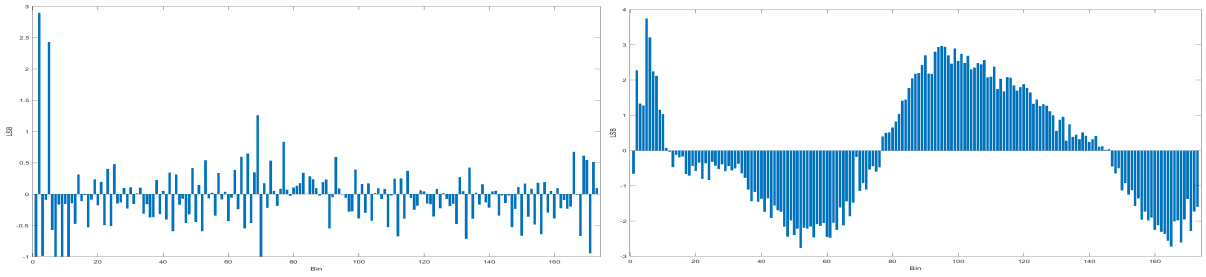
After analyzing the Tap size distribution, the non-linearities of the channel must be examined. Firstly the focus is on the non-linearities of the Start Line. The maximum DNL of the Start Line is of 3.96 LSB (see Figure 5.38a), it is considerably higher than the value for the 16-Channel Implementation. Moreover, the INL of the Start Line varies on a range between 3.62 LSB and -4.10 LSB. Even though, the non-linearities of the current Start Line are greater than the ones observed in the best performing channel, if they are compared with the non-linearities of the worst channel of the 16-Channel Implementation, it can be concluded that the reduction of the number of channels, in this case, had an influence in enhancing the linearity of the Start Line of the worst channel.



(a) **(b)**

Figure 5.38: DNL **(a)** and INL **(b)** results of the Code Density Test of the Start Line of the the Worst Channel of the 4-Channel Implementation.

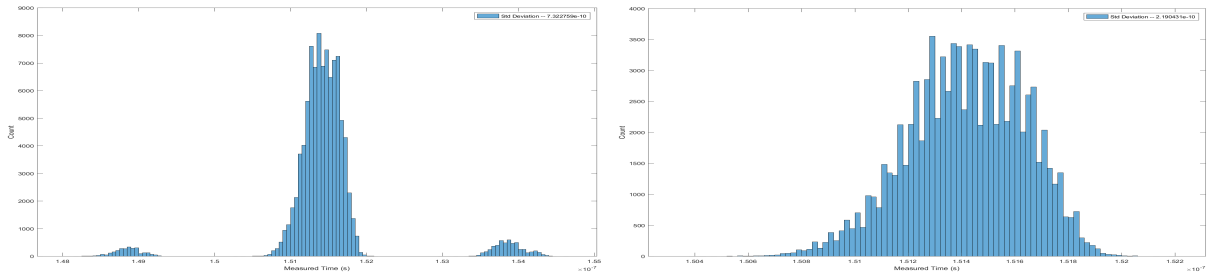
Afterward, the non-linearities of the Stop Line must be studied. The Stop Line has a maximum Differential nonlinearity of 2.90 LSB (see Figure 5.39a). This value is slightly higher than the one of the worst performing channel of the 16-Channel Implementation. The range of the Integral nonlinearity varies from the positive maximum of 3.74 LSB to the negative maximum of -2.77 LSB (see Figure 5.39b). Contrarily to what happened in the Start Line, the reduction of the number of channels did not improve the linearity of the Stop Line comparing with the 16-Channel Implementation, meaning that there is no sustained evidence on the benefits of diminishing the number of channels in order to achieve better linearity.



(a) **(b)**

Figure 5.39: DNL **(a)** and INL **(b)** results of the Code Density Test of the Stop Line of the the Worst Channel of the 4-Channel Implementation.

Regarding the precision of the channel, it can be concluded that the influence of the Coarse Metastability error, although being noticeable it is not as great as the influence on the best performing channel of the current implementation, the measurement precision of this channel is of 732.27 ps (see Figure 5.40a), which is better than the 1.10 ns of the best channel. When the influence of the Coarse error is ignored, there is an enhancement of the precision of the current channel to 219.04 ps (see Figure 5.40b). Nevertheless, this is less precise than the best performing channel of the current implementation and also less precise that the worst performing channel of the 16-Channel implementation, meaning that there is no real advantage of reducing the number of implement channels in terms of measurement precision.



(a) **(b)**

Figure 5.40: Measurement Histogram and Precision of the Worst Channel of 4-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

5.2.4 Results for 2 channel implementation

So that it may be possible to thoroughly analyze the effect of the number of implemented channels in the performance of the Acquisition System, the 2-Channel Implementation will be studied.

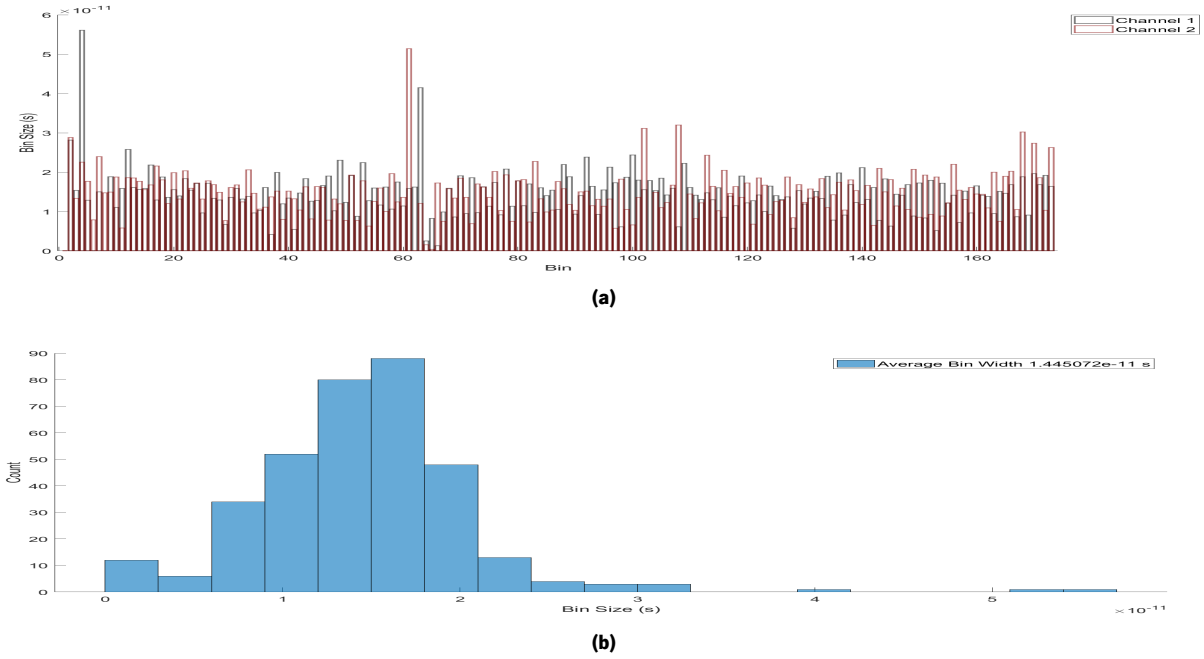


Figure 5.41: Measured Bin Width **(a)** for each channel and **(b)** the bin width Histogram of the Start of all the bins of the 2-Channel Implementation.

The first aspect to be analyzed is the distribution of the Tap Size of the Start line of all the implemented channels. The majority of Taps have a size close to the ideal bin size, with the average size being the expected value. The widest tap of the implementation has a size of 56.10 ps in Tap 4 of channel 1 (see Figure 5.41a), and the average number of ultra-wide bins per channel is 3. The average number of bins below the 10 ps size is 34, which is a little higher than the average of the 16-Channel Implementation (see Figure 5.41b).

Next, the Bin width distribution of the Stop line of every channel is examined. The average number of bins with less than 10 ps of size is 38 Bins (see Figure 5.42b). The widest bin is the same that in the Start line, however, the value for the Stop line is higher than for the Start with a size of 59.15 ps (see Figure 5.42a), and the average number of ultra-wide bins is 2 bins per channel.

Finally, the precision of the measurements of the Acquisition System for the current implementation is examined. Likewise to the 16-Channel Implementation, the vast majority of measurements are close to the real size of the Hit, the average measurement value of 151.33 ns which is close to the average measurement value of the 16-Channel Implementation. Furthermore, the average precision of the system

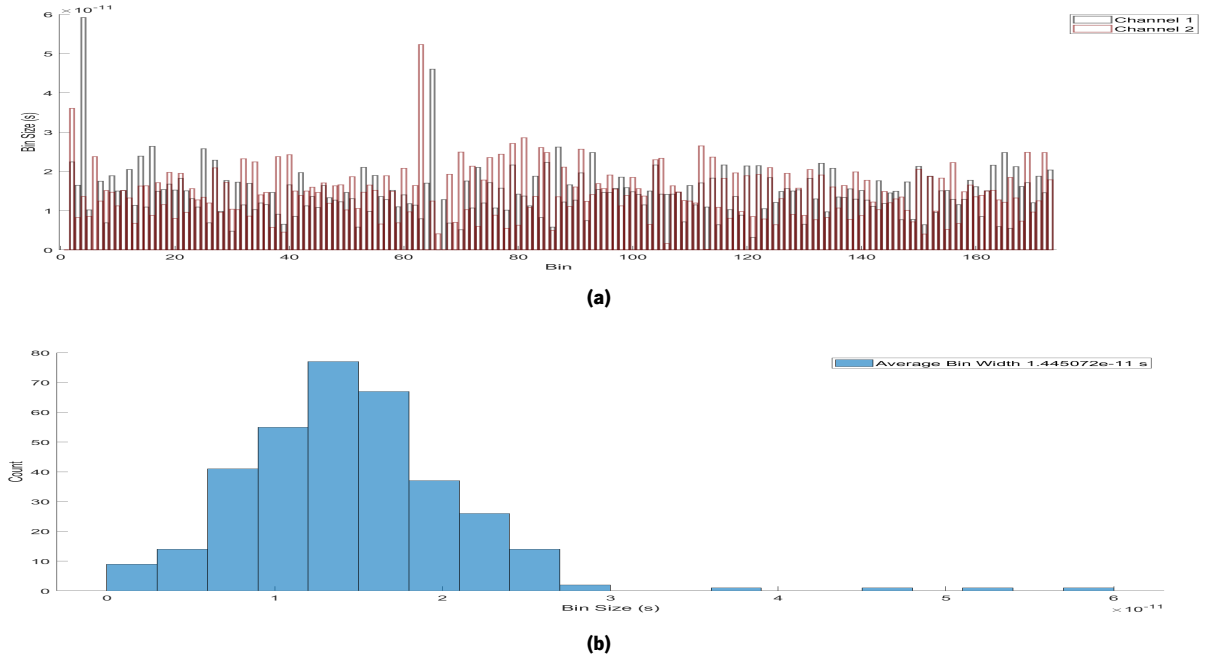


Figure 5.42: Measured Bin Width **(a)** for each channel and **(b)** the bin width Histogram of the Stop of all the bins of the 2-Channel Implementation.

is of 410.36 ps with the effect of the Metastability error. If the Coarse error is not taken into account, the precision of the measurement is enhanced to an average precision of 210.157 ps.

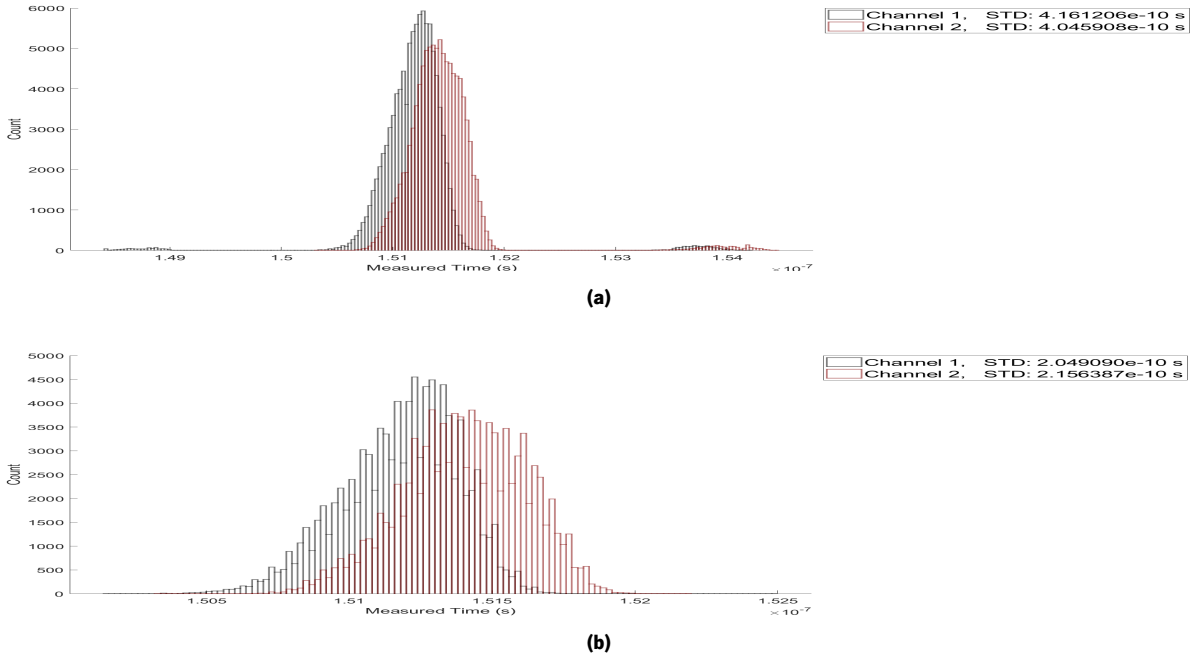


Figure 5.43: Measurement Histogram and Precision, of all channels of the 2-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

5.2.4.1 Resource Utilization

As seen in Table 5.4, the reduction of the number of implements channels diminishes the resource utilization of LUTs, BRAMs, and FFs. The resource with the highest utilization is the LUT, where 12.50 percent where used, all other resources report a utilization under 5 percent.

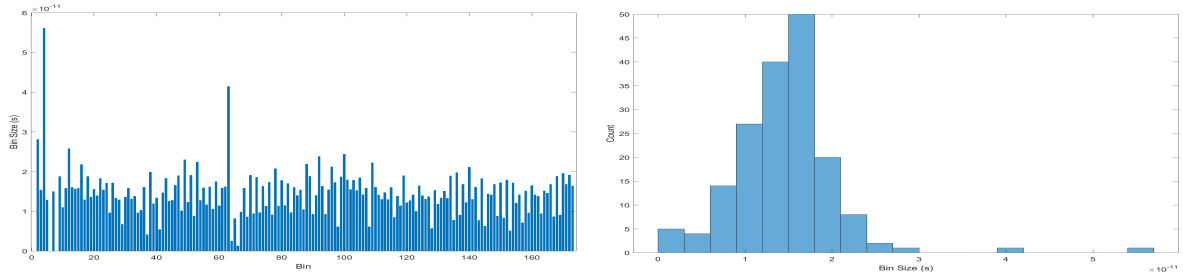
Resource	Utilization	Available	Utilization
LUT	10549	230400	4.58
LUTRAM	523	191760	0.51
FF	8295	460800	1.90
BRAM	48	312	15.38
IO	6	360	1.67
BUFG	6	544	3.68
MMCM	1	8	12.50

Table 5.4: Resource Utilization of the 2 Channel Implementation

5.2.4.2 Channel with the best performance

As there are only two channels in the current implementation, both must be studied and compared with the 16-Channel Implementation. The first channel to be studied is the more linear channel.

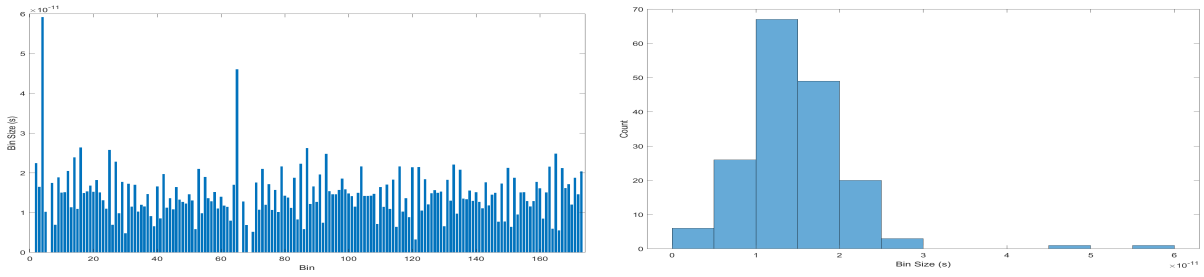
The distribution of the Bin widths of the Start line of the channel is displayed in Figure 5.44. The majority of Bins has a size close to the ideal. Furthermore, the widest Bin of this channel has a size of 56.10 ps (see Figure 5.44a), the number of ultra-wide bins is 2, and the number of small bins is 36 (see Figure 5.44b).



(a) **(b)**

Figure 5.44: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Start of the Best Channel of the 2-Channel Implementation.

The Bin width distribution of the Stop line is will now be examined. The majority of bins has a size close to ideal Bin width. The number of Bins with a size smaller than 10 ps is 32, the number of ultra-wide bins is 2 (see Figure 5.45b). The widest Bin of the line has a size of 59.50 ps (see Figure 5.45a).

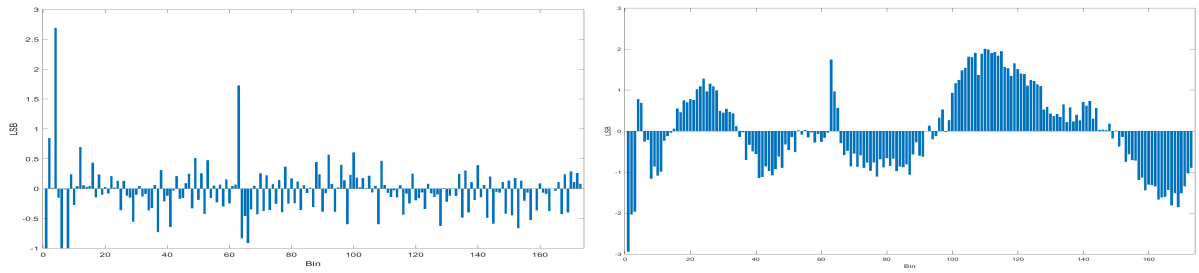


(a) **(b)**

Figure 5.45: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Stop of the Best Channel of the 2-Channel Implementation.

After examining the bin width distribution of both lines, it is now necessary to analyze the linearity of the channel. First, the linearity of the Start line is analyzed.

The maximum value of DNL of the Start line is 2.69 LSB (see Figure 5.46a), which, compared with the 16-Channel Implementation, has a higher value of non-linearity. In terms of INL, the non-linearities vary from 2.00 LSB to -2.94 LSB (see Figure 5.46b). In comparison with the 16-Channel Implementation, the current line has a bigger range of non-linearities values, although the positive maximum of the current channel is lower.

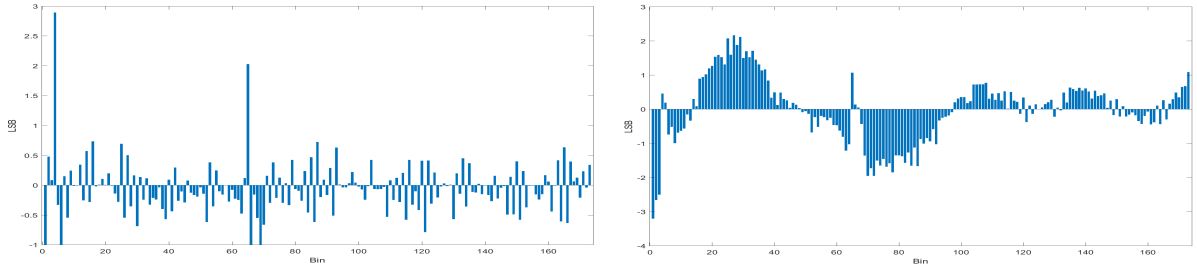


(a) **(b)**

Figure 5.46: DNL **(a)** and INL **(b)** results of the Code Density Test of the Start Line of the the Best Channel of the 2-Channel Implementation.

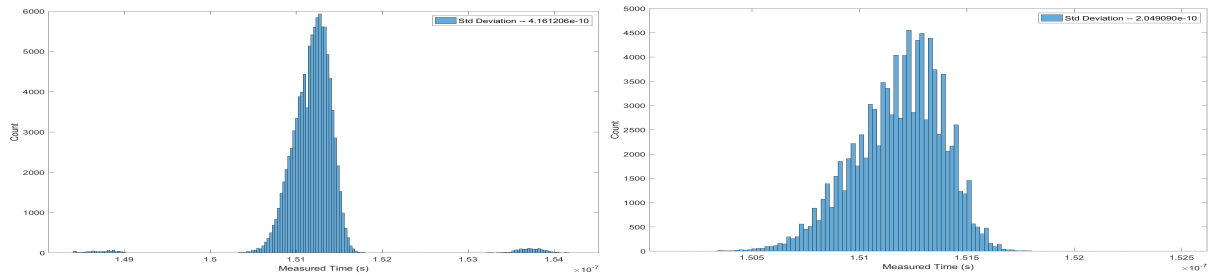
Second, the linearity of the Stop line is analyzed. The maximum DNL has a value of 2.89 LSB (see Figure 5.47a), which is a higher value that both the Start line of the current channel and the Stop line of the 16-Channel Implementation. The values of the Integral nonlinearity are in a range between 2.17 LSB and -3.20 LSB (see Figure 5.47b).

Finally, the precision of the channel is analyzed. If the Metastability error is taken into account, the precision of the channel is 416.12 ps (see Figure 5.48a). Comparing with the 16-Channel Implementation, it is far better precision. If the error is not taken into account, the precision is enhanced to 204.90 ps (see Figure 5.48b), which is less precise that the 16-Channel Implementation.



(a) **(b)**

Figure 5.47: DNL **(a)** and INL **(b)** results of the Code Density Test of the Stop Line of the the Best Channel of the 2-Channel Implementation.



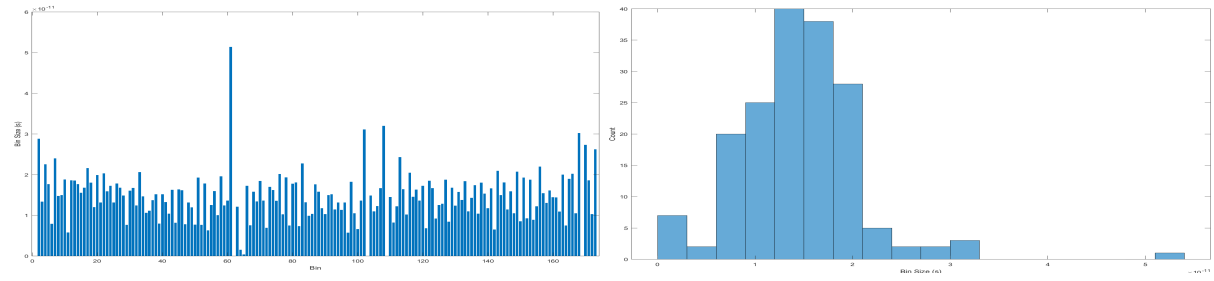
(a) **(b)**

Figure 5.48: Measurement Histogram and Precision of the Best Channel of 2-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

5.2.4.3 Channel with worst performance

After examining the most linear channel, the other channel must be analyzed to see if the reduction of the number of channels improves the performance of the worst channel of the implementation.

So firstly, the Start line Tap width distribution is studied. As seen in Figure 5.49b, the bulk of the Bins have a size close to the ideal. The widest Tap of the Start line has a size of 51.38 ps (see Figure 5.49a), with 4 ultra-wide bins, and the number of bins under 10 ps is 32 (see Figure 5.49b).

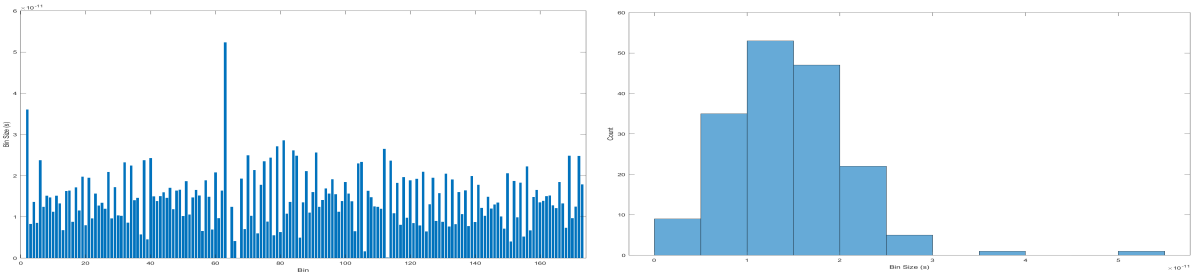


(a) **(b)**

Figure 5.49: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Start of the Worst Channel of the 2-Channel Implementation.

Then the Bin width distribution of the Stop line is analyzed. The distributions follow the regular pattern

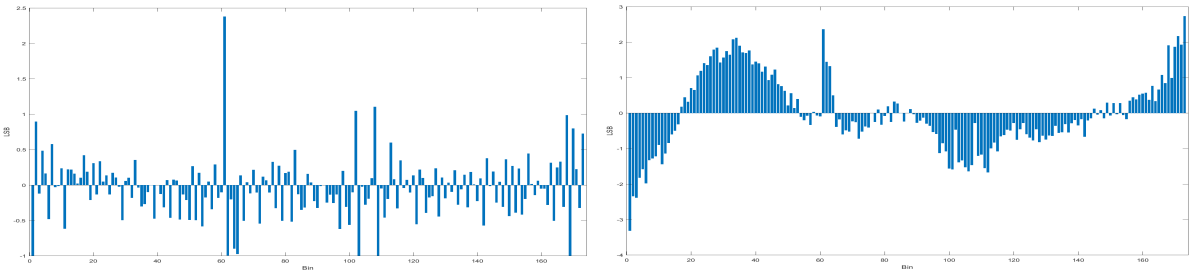
seen in the other channels. The widest Bin has a size of 52.30 ps (see Figure 5.50b), and 4 ultra-wide bins. The number of Bins smaller than 10 ps is 44 (see Figure 5.50a). Although the size of the widest Bin is smaller for the Stop line of the best channel, the number of small taps is far superior, which will have a negative effect on the performance.



(a) **(b)**

Figure 5.50: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Stop of the Worst Channel of the 2-Channel Implementation.

Now the linearity of the Start line is examined. The maximum DNL of the Start line has a value of 2.38 LSB (see Figure 5.51a), which is smaller than the DNL value of the best channel. However, in terms of INL, the range of the non-linearities is higher for this channel, varying from 2.73 LSB to -3.32 LSB (see Figure 5.51b).

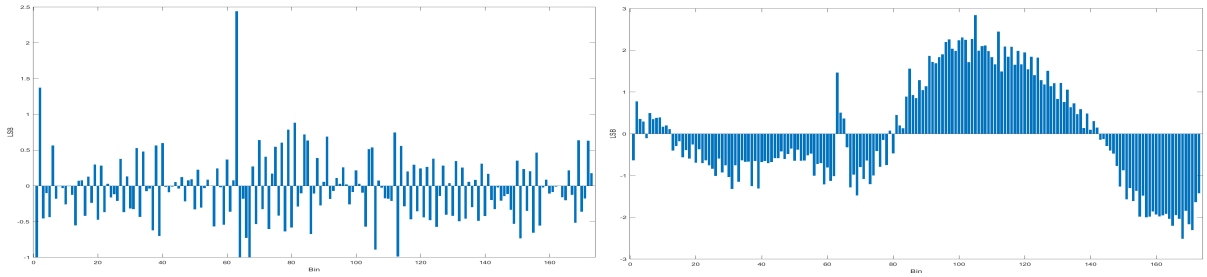


(a) **(b)**

Figure 5.51: DNL **(a)** and INL **(b)** results of the Code Density Test of the Start Line of the the Worst Channel of the 2-Channel Implementation.

Afterward, the linearity of the Stop line is examined. The maximum Differential nonlinearity has a value of 2.44 LSB (see Figure 5.52a), a value lower than the one from the best performing channels. Nevertheless, the range of the INL is bigger in this channel, with the value varying between -2.52 LSB and 2.84 LSB (see Figure 5.52b).

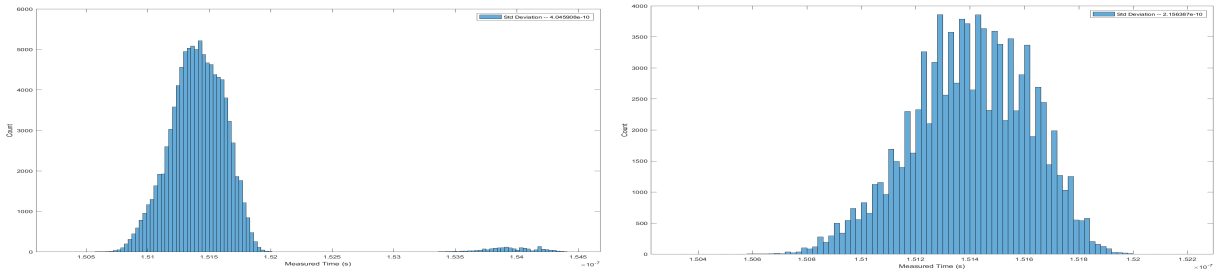
Finally, the precision of the channel is analyzed. If the Coarse error is taken into account, the precision of the current channel is better than the precision of the best performing channel, being significantly more precise that the worst performing channel of the 16-Channel Implementation, with a precision value of



(a) **(b)**

Figure 5.52: DNL **(a)** and INL **(b)** results of the Code Density Test of the Stop Line of the the Worst Channel of the 2-Channel Implementation.

404.59 ps (see Figure 5.53a). However, if the Coarse error is not taken into account, the precision is enhanced to 215.64 ps (see Figure 5.53b), which is worse than both the value of the best performing channel and the worst channel of the 16-Channel Implementation.



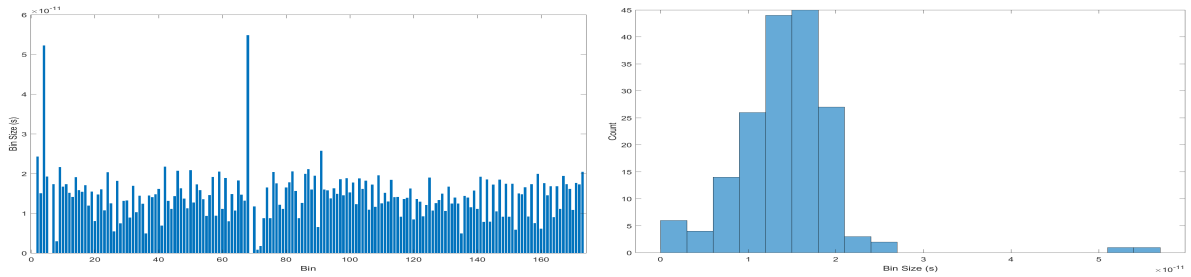
(a) **(b)**

Figure 5.53: Measurement Histogram and Precision of the Worst Channel of 2-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

5.2.5 Results for 1 channel implementation

The last implementation to be analyzed is of an Acquisition System with a single TDC channel, to study its performance.

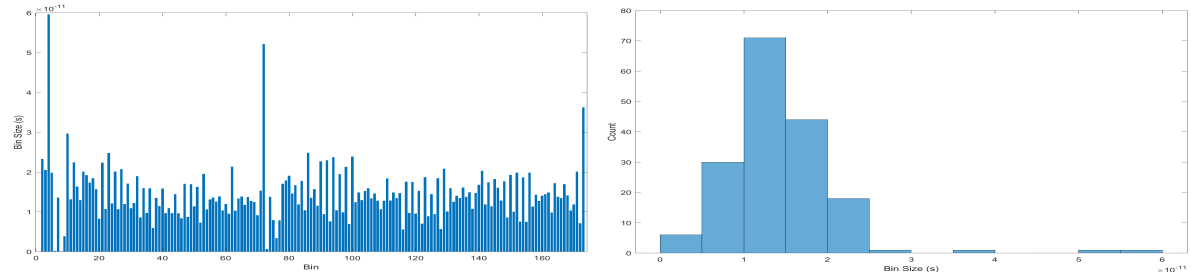
As in the other implementation, the first thing to be analyzed is the Bin width distribution of the Start Line. The average width, as expected, is of 14.45 ps as at every other channel. The maximum size of the Bins of the Start Line is 54.85 ps (see Figure 5.54a), which, compared with the best channel of the 16-Channel Implementation, is slightly bigger than the widest Bin of that implementation, and there are 2 ultra-wide bins. In terms of bins with less than 10 ps of size there are 32 bins in the Start Line (see Figure 5.54b), comparing with the average number per channel the for 16-Channel Implementation is a little higher.



(a) **(b)**

Figure 5.54: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Start of the Channel of the 1-Channel Implementation.

After analyzing the Start Line, now the Bin width distribution of the Stop is examined. The number of Bins with a width under 10 ps is 36, which is higher than the average number for the 16-Channel Implementation (see Figure 5.55b). The widest Bin of the Stop Line has a size of 59.58 ps (see Figure 5.55a), which is bigger than the widest bin of the best channel of the 16-Channel Implementation, furthermore the number of ultra-wide bins is 3.

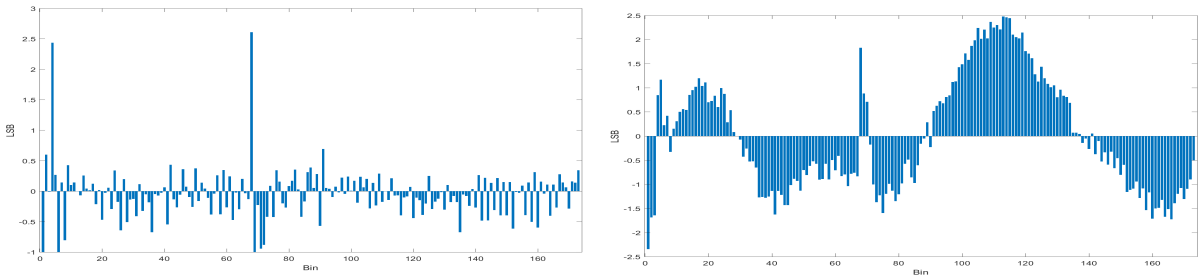


(a) **(b)**

Figure 5.55: Measured Bin Width **(a)** and **(b)** the bin width Histogram of the Stop of the Channel of the 1-Channel Implementation.

Now that the distribution of the bin width was examined, it is necessary to study the non-linearities of the Line. In the Start Line the maximum DNL is of 2.61 LSB, which is a higher value of DNL compared with the best channel of the 16-Channel Implementation. The INL of the Line ranges from 2.48 LSB to -2.34 LSB, which is a more significant non-linearity range that the one seen in the best channel of the 16-Channel Implementation.

In the Stop Line, the maximum Differential nonlinearity is of 2.91 LSB, which is superior to the best channel of the 16-Channel Implementation. The INL range of the Stop Line is between 2.21 LSB and -3.72 LSB. These non-linearities are far superior to the one seen in the Start Line of the 16-Channel Implementation.



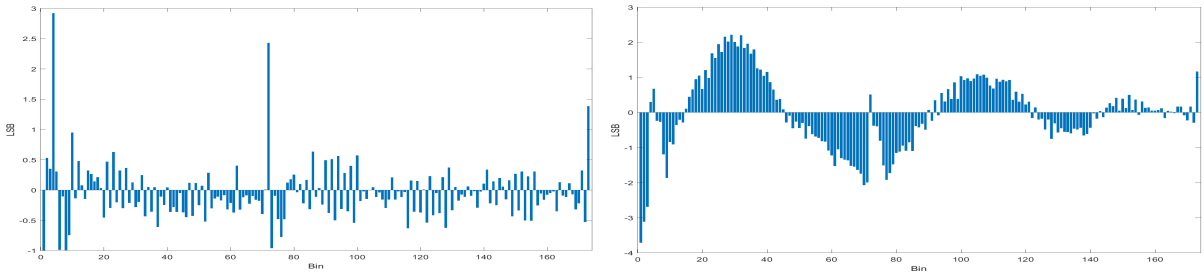
(a) **(b)**

Figure 5.56: DNL **(a)** and INL **(b)** results of the Code Density Test of the Start Line of the Channel of the 1-Channel Implementation.

Finally, the precision of the channel is examined. The first analysis takes into account the influence of the Metastability error in the measurements, the precision of measurements, in this case, is of 570.15 ps, meaning that the effect of the error is less noticeable compared with the best channel of the 16-Channel Implementation. When the Coarse error is not taken into account, the precision of the channel increases to 192.13 ps, which slightly lower than the precision of the best channel of the 16-Channel Implementation. So it can be concluded that even if the reduction of the number of implemented channel does not improve the precision and performance of the channels it also does not deteriorate it much meaning that if needed to reduce the resource utilization the number of channels can be reduced without the risk of deteriorating the measurements done by each channel.

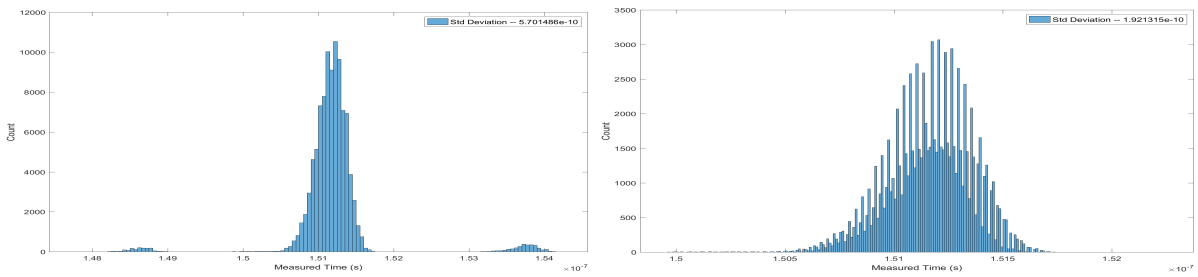
5.2.5.1 Resource Utilization

As seen in Table 5.1, the resource utilization of the Acquisition System when a single channel is implemented is very low. Therefore there is a huge quantity of resources available for implementing other modules.



(a) **(b)**

Figure 5.57: DNL **(a)** and INL **(b)** results of the Code Density Test of the Stop Line of the Channel of the 16-Channel Implementation.



(a) **(b)**

Figure 5.58: Measurement Histogram and Precision of the Channel of 1-Channel Implementation, of the time interval of 151.5 ns. All measurements **(a)** and measurements without the metastability error **(b)**.

Resource	Utilization	Available	Utilization
LUT	10549	230400	4.58
LUTRAM	523	191760	0.51
FF	8295	460800	1.80
BRAM	6	312	1.92
IO	6	360	1.67
BUFG	6	544	1.10
MMCM	1	8	12.50

Table 5.5: Resource Utilization of the 1 Channel Implementation

5.3 Conclusion

The summary of the performances of each of the implementations is displayed in Table 5.6, the average precision of all implementations does not differ much except in the 8-Channel Implementation, due to influence of the worst precision channel.

Implementation	Best precision	Average Precision	Worst Precision
16-Channel	188.66 ps	207.47 ps	232.17 ps
8-Channel	199.73 ps	229.08 ps	343.09 ps
4-Channel	191.44 ps	206.91 ps	219.04 ps
2-Channel	204.91 ps	210.18 ps	215.64 ps
1-Channel	192.13 ps		

Table 5.6: Precision Results of the various implementations

Chapter 6

Conclusions and Future Work

After examining the results of the tests that demonstrated the performance of the Acquisition System, and also after analyzing the influence of the number of implemented TDC channels, conclusions can be made about implementing the developed work and the implementation of TDCs in FPGA. Therefore this chapter has the goal of making an overview of the results obtained and of drawing conclusions from them identifying possible flaws in the current System and making suggestions on future improvements to the Acquisition System.

6.1 Conclusion

An Acquisition System for a LiDAR sensor was developed, it has sixteen TDC peripherals implemented, making it possible to read sixteen different input signals. The measurement precision of the implemented TDCs varies between 232.17 ps and 188.66 ps, with an average precision of 207.47 ps, if the Coarse Counter Metastability error is not taken into account.

The implemented TDC peripherals were based on the Tapped-Delay Line architecture, due to its simplicity and good resolution. However, in order to achieve good linearity on the TDL and measurement precision, a Calibration Module was developed. This Module is based on an online decimation calibration technique.

Furthermore, due to the Coarse Counter Metastability error, a Synchronizer based on the one proposed in [51] was implemented. Although the Synchronizer reduced the effect of the Metastability error in the measurements, it still has a deteriorating effect on the performance of the TDC channel. Analyzing the results the causes of this deterioration is noticeable in two different cases, first and the most influential one is the counting of an extra value or not reading one value, which creates an error of the measurement of more or less one clock cycle. The other case is connected to the acquisition of Thermometer Code, due

to the Metastability error the Store Stage of the TDL will sometimes acquire the status of the delay line a clock cycle after the Start or Stop of the signal, meaning that the signal will be captured in taps at the end of the chain that, even though they are part of the line, should never have value captured in them. This lowers the value of the LSB from the expected 15.2 ps to a 14.45 ps seen in the results, worsening the precision of the measurements.

The Synchronizer did not completely eliminate the Metastability error of the Coarse Counter. Therefore it is crucial to understand why this did not happen. The Synchronizer uses two Coarse Counters connected to two phased clocks, one with a phase of 72 degrees and another with a phase of 144 degrees, with the same frequency that the reference clock, to correct the values of the Coarse Counter (see Section 4.1.4). However, due to different routing between the phased clock generator and the counters of each TDC channel, the insertion delay of the clock is not always equal, meaning that the skew of the phase clocks is different from channel to channel. Furthermore, in extreme cases, the routing propagation time of the reference clock, compared with the phased clocks, can be so significant that one or even both clocks' phases can arrive before the reference clock, rendering the Synchronizer unusable. This problem grows with the increase of the number in implemented channels, as the routing of the clocks becomes more difficult. The only way to prevent such cases would be to manually routing the clock trees of the counters ensuring the elimination of these phase inversion scenarios. However, this is an arduous task, thus further research needs to be done regarding the Synchronizer to find a simpler, layout independent solution.

The TDC Peripherals developed, were designed with the goal of being easily integrable in every design, so the communication between the peripheral and the rest of the design can be done in two different ways, via AXI bus like in the current Acquisition System. Alternatively by connecting directly with the FIFO of the TDC, as the FIFO was developed to allow this type of communication and any application can use to the TDC by connecting to the READFIFO wire and the FIFODATA.

6.2 Future Work

The next steps in the development of the Acquisition System will be to develop and implement a Synchronizer that completely corrects the Metastability error, and to integrate the the Acquisition System with the LiDAR.

In order to correct the Metastability error, a new Synchronizer design can be developed (see Figure 6.1). It generates two signal that validate the values acquired by the TDL and the Coarse Counter at the

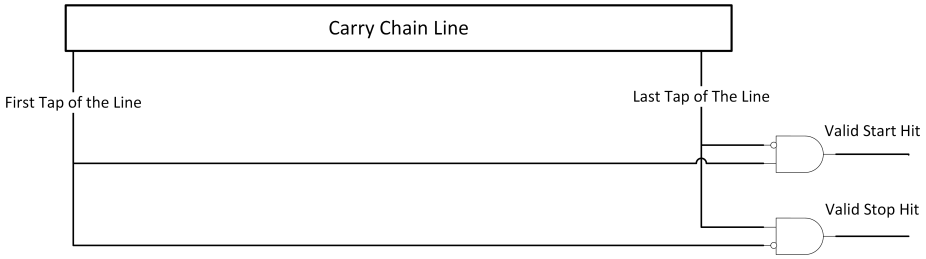


Figure 6.1: Design of the new Synchronizer that should correct the Metastability error.

Start and Stop of the Hit Signal. The generation of this validation signals is done by comparing the first and last bins of the TDL. The Valid Start Hit signal is only enabled when the first bin of the TDL is one and the last bin is zero, meaning that the a Start event occurred and therefore the Thermometer Code acquired by the sample Flip-Flops is valid. While the Valid Stop Hit signal is only enabled when the first bin of the TDL is zero and the last bin is one, which indicated that a Stop event occurred and thus the value of the Thermometer Code in the sample FFs is valid, and the value of the Coarse Counter can be acquired and stored.

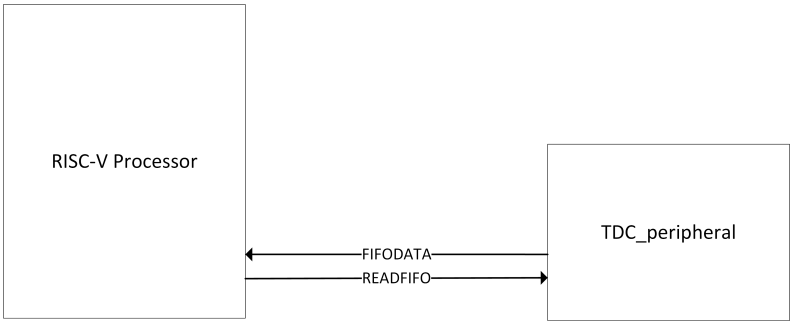


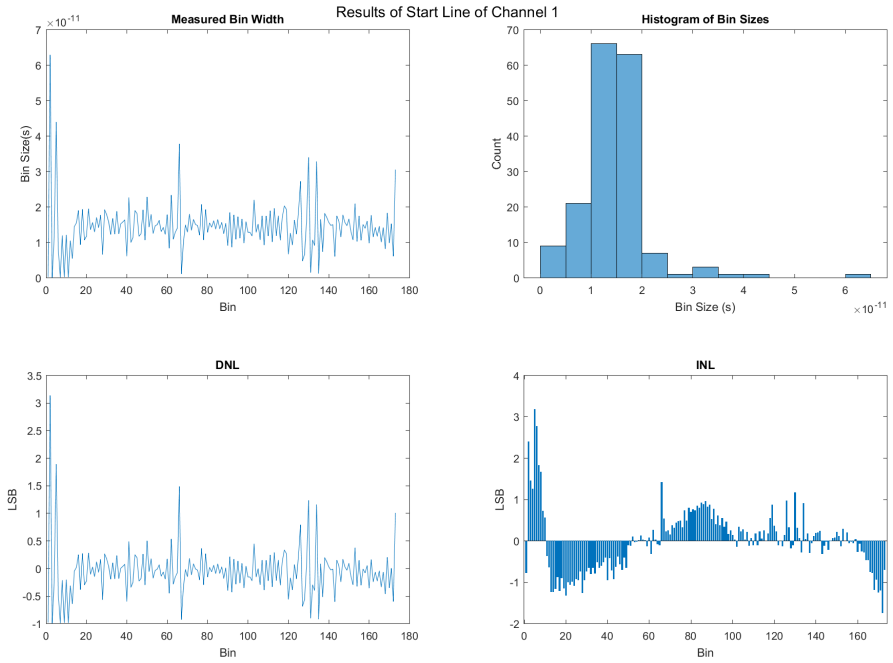
Figure 6.2: Integration of one TDC_Peripheral in a RISC-V

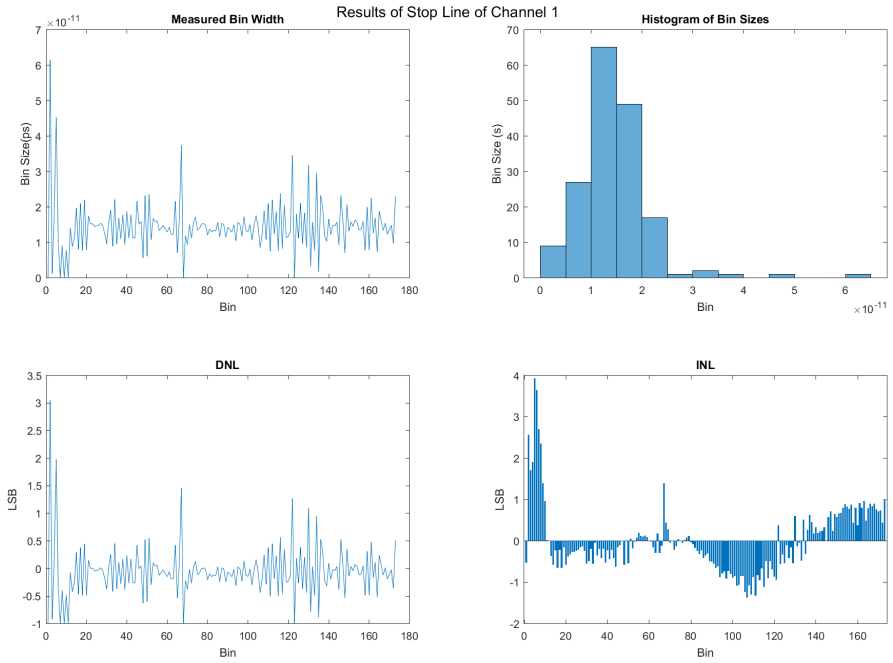
The integration of the Acquisition System in the LiDAR sensor will be done by integrating the TDC_Peripheral with the RISC-V processor used by the sensor. To do so, the processor must have a bus connected to the FIFODATA output of the TDC_Peripheral and a pin connected o the READFIFO signal to request a new value from the FIFO of the TDC (see Figure 6.2).

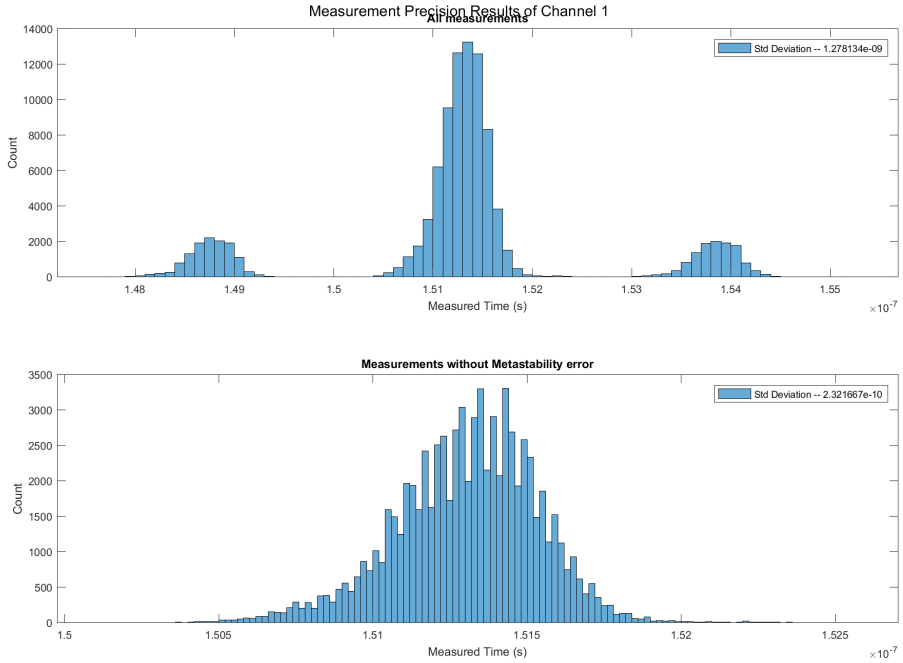
Appendix A

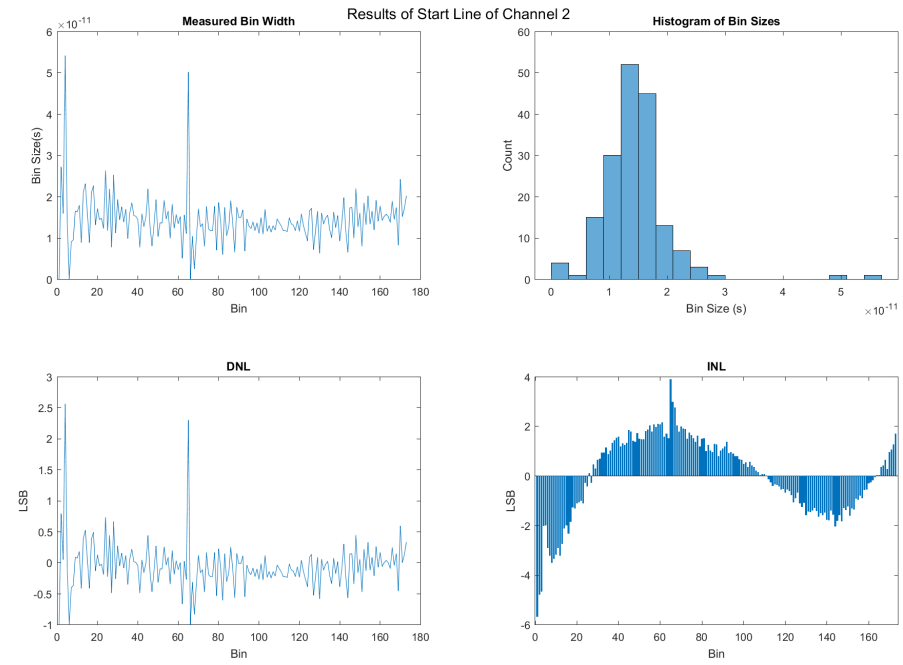
Results of the Tests of the various implementations

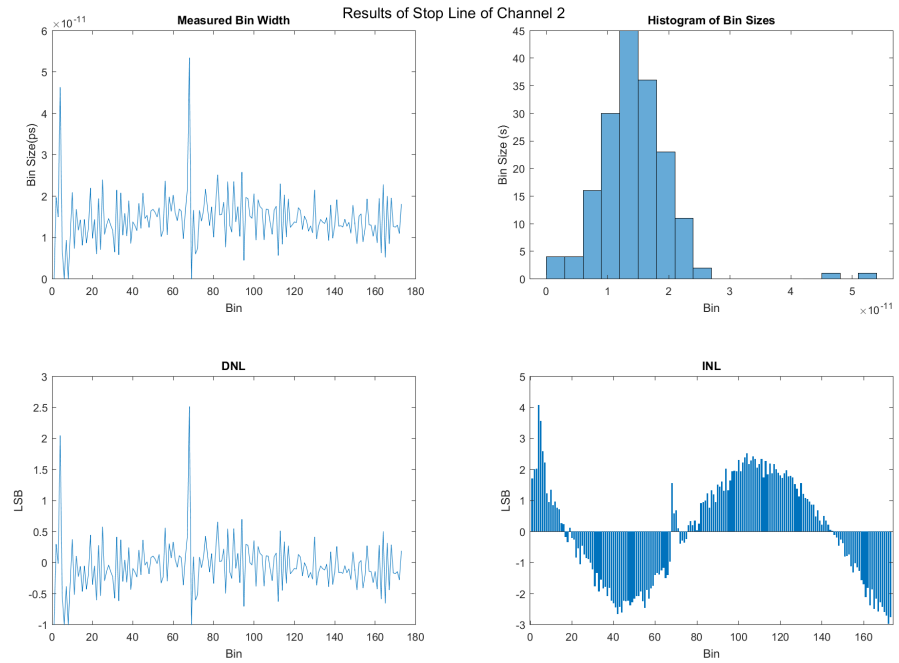
16-Channel Implementation

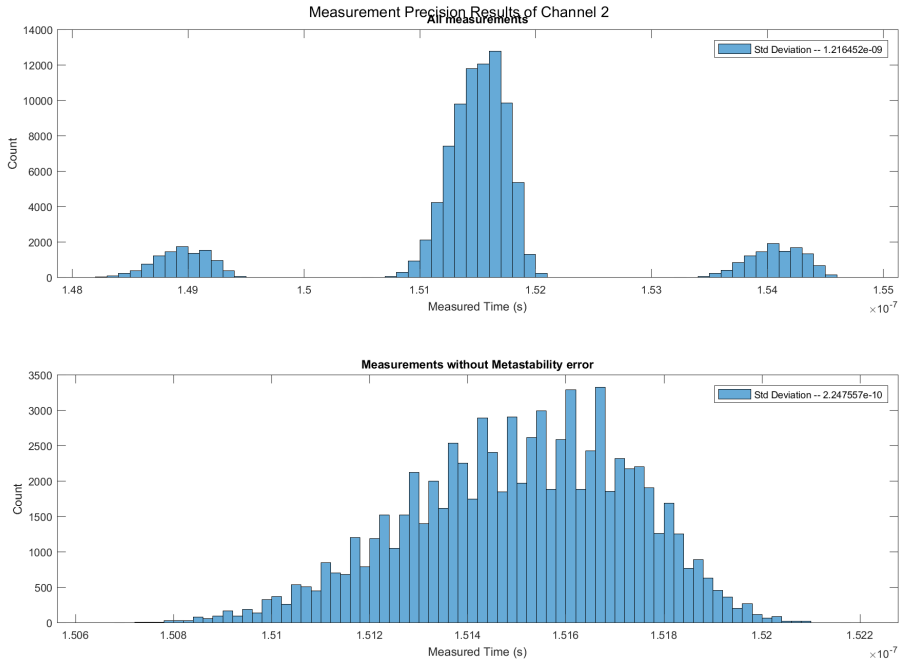


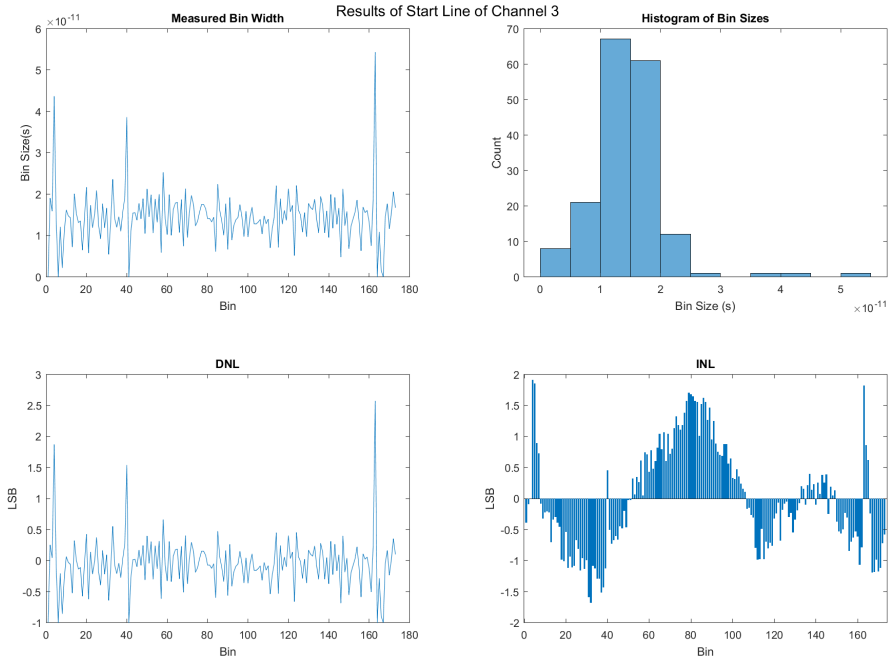


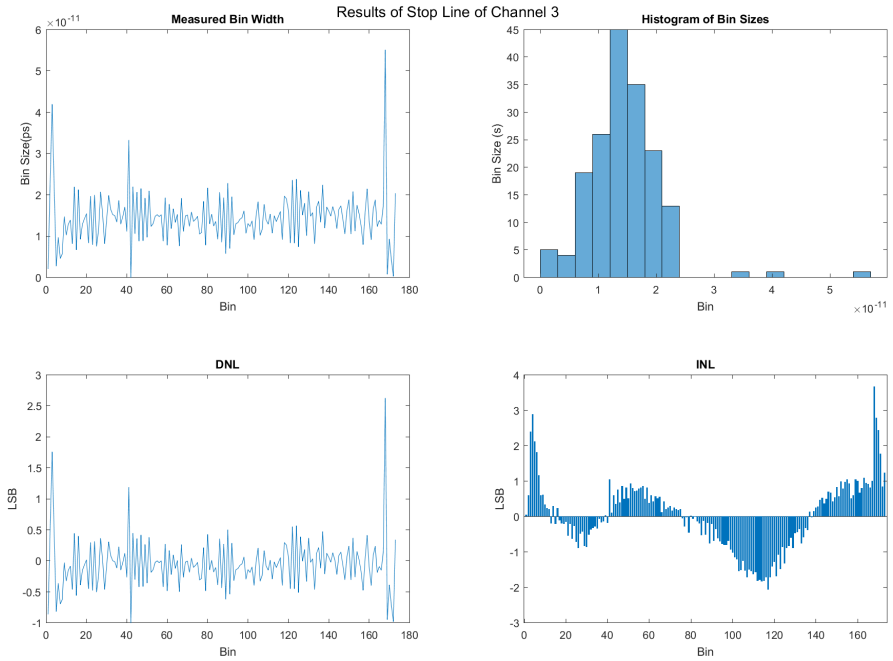

Measurement Precision Results of Channel 1
All measurements
Std Deviation -- 1.278134e-09
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.321667e-10
Count
Measured Time (s)
×10^-7

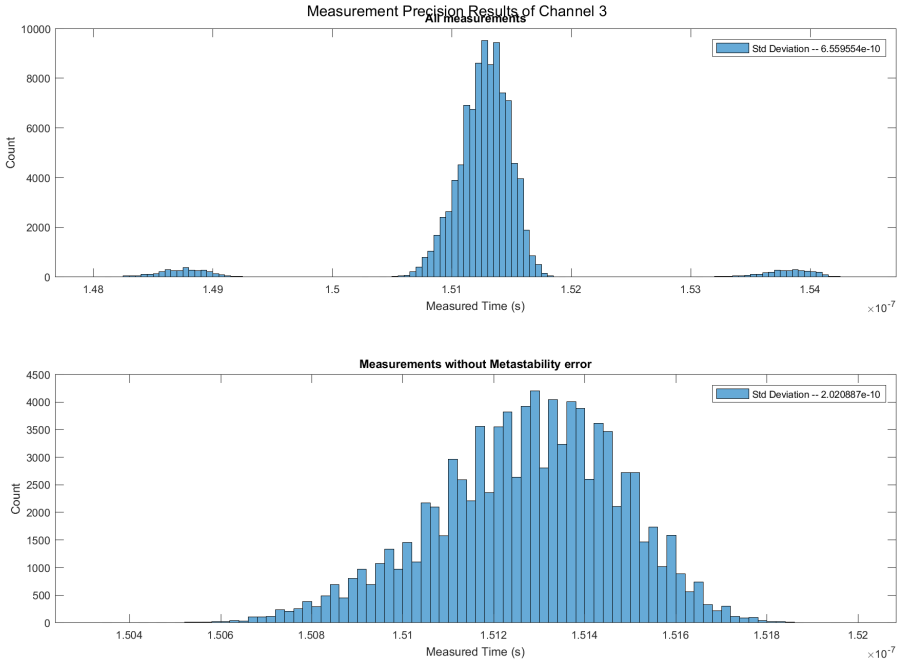

Results of Start Line of Channel 2
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

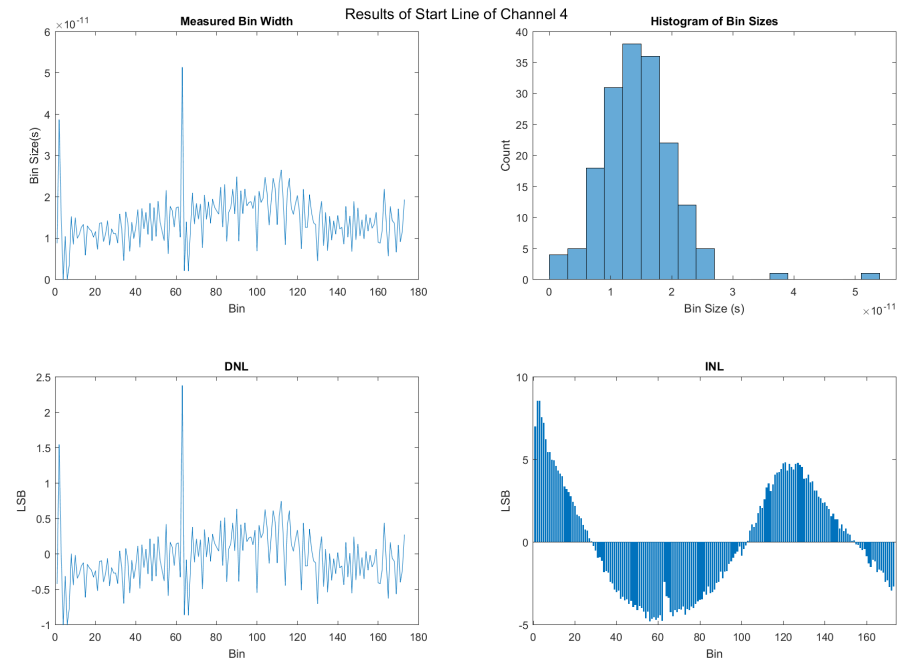

Results of Stop Line of Channel 2
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

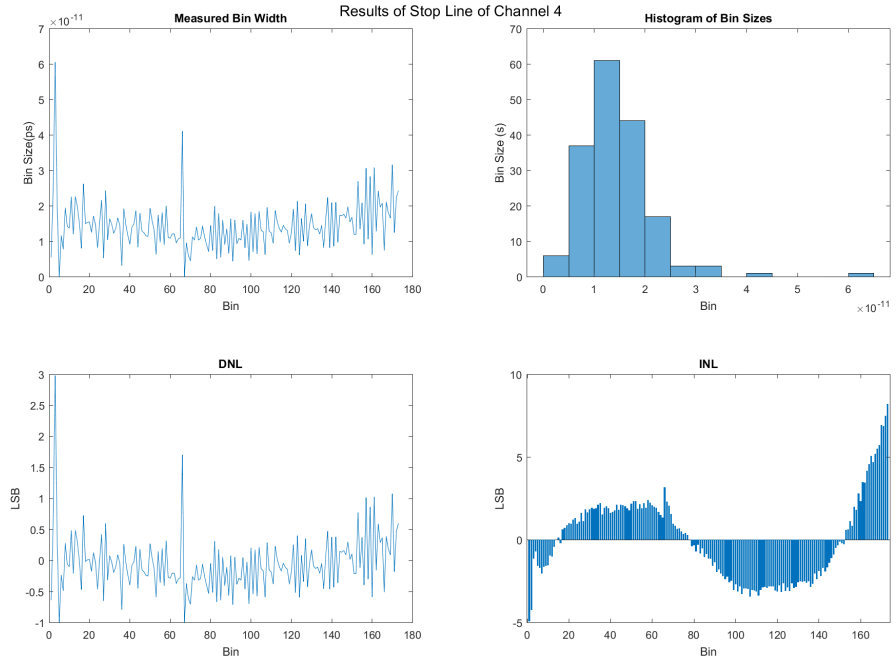

Measurement Precision Results of Channel 2
All measurements
Std Deviation -- 1.216452e-09
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.247557e-10
Count
Measured Time (s)
×10^-7

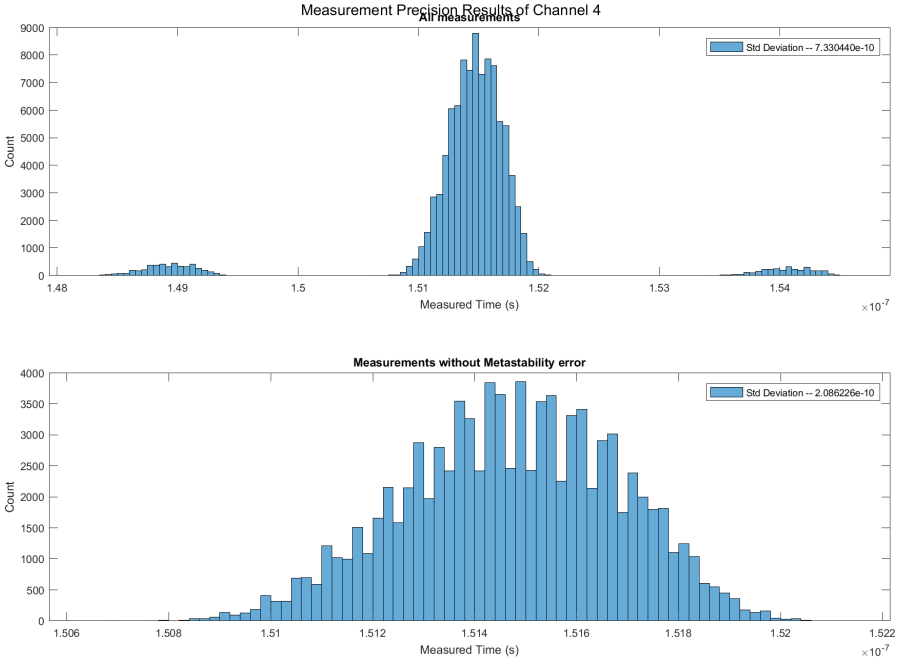

Results of Start Line of Channel 3
Measured Bin Width
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
DNL
LSB
Bin
INL
LSB
Bin

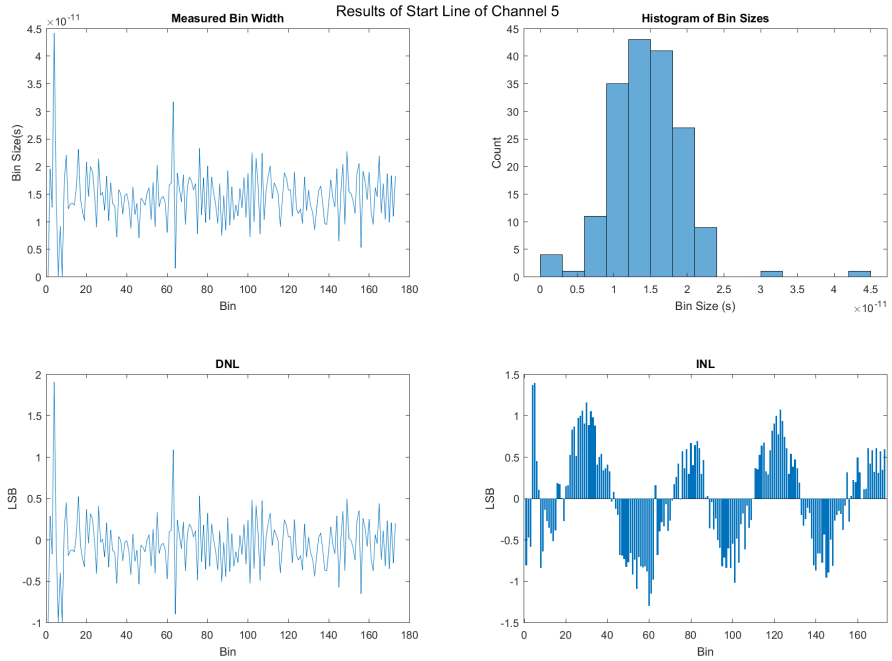

Results of Stop Line of Channel 3
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

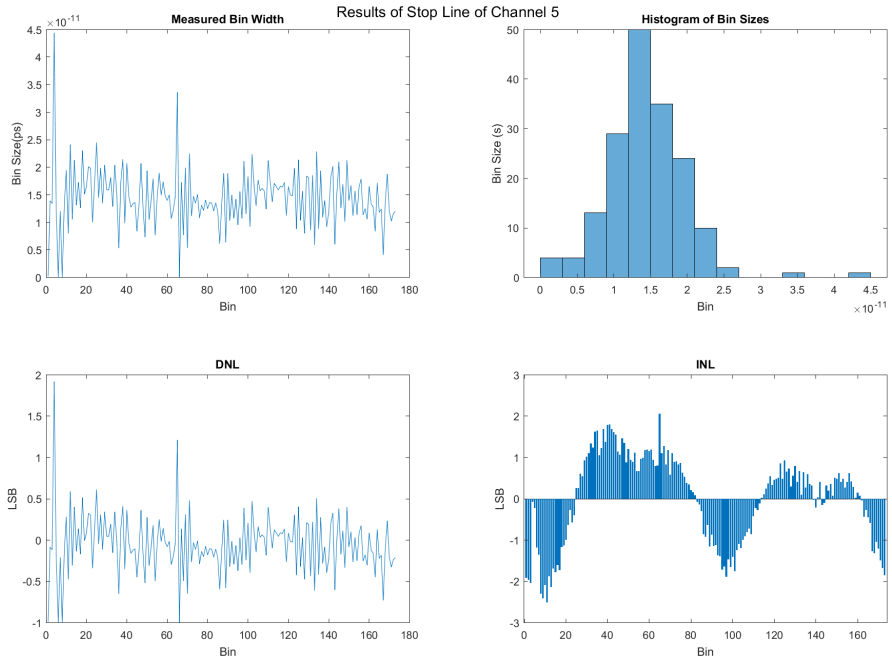

Measurement Precision Results of Channel 3
All measurements
Std Deviation -- 6.559554e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.020887e-10
Count
Measured Time (s)
×10^-7

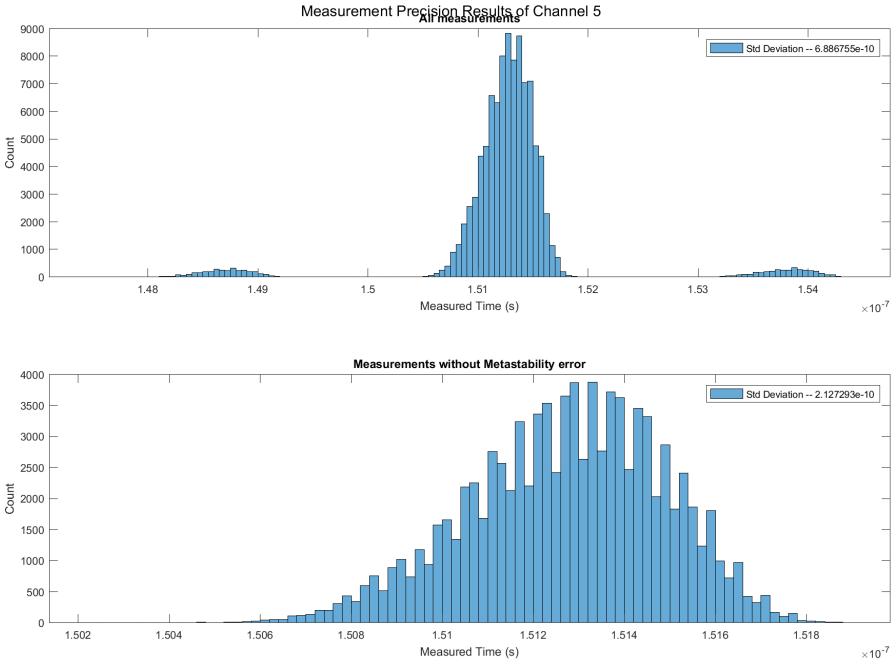

Results of Start Line of Channel 4
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

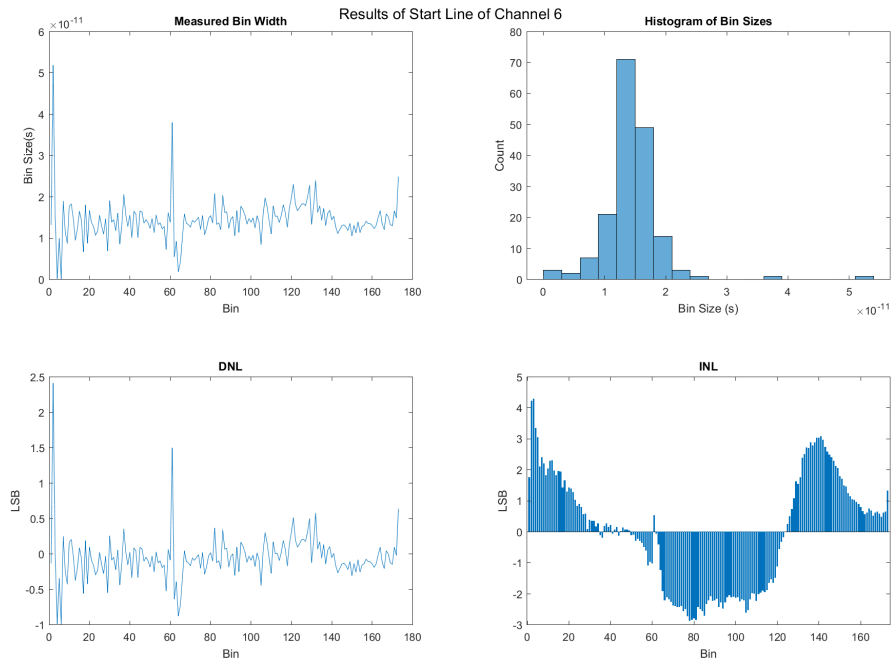

Results of Stop Line of Channel 4
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

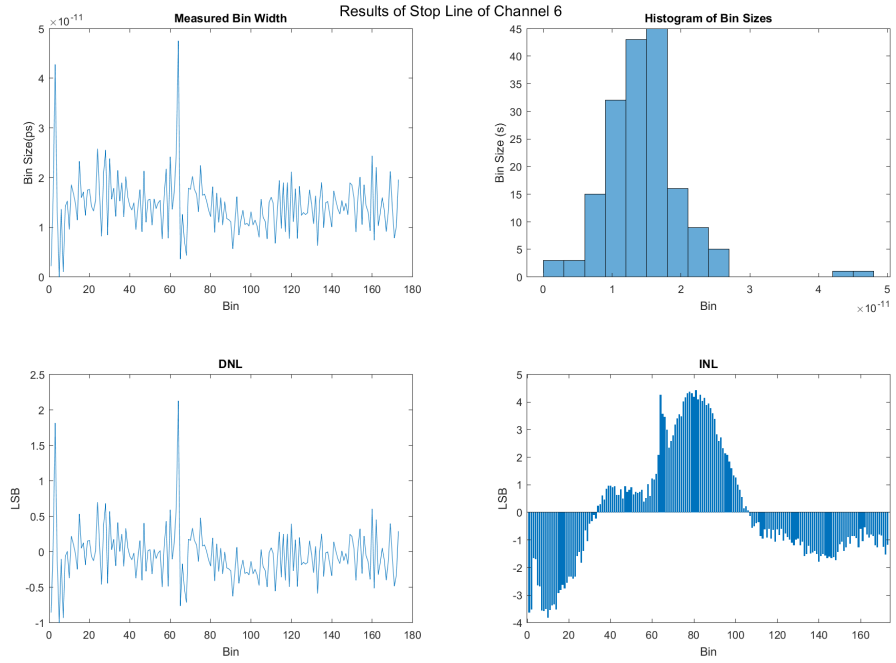

Measurement Precision Results of Channel 4
All measurements
Std Deviation -- 7.330440e-10
Count
Measured Time (s)
Measurements without Metastability error
Std Deviation -- 2.086226e-10
Count
Measured Time (s)

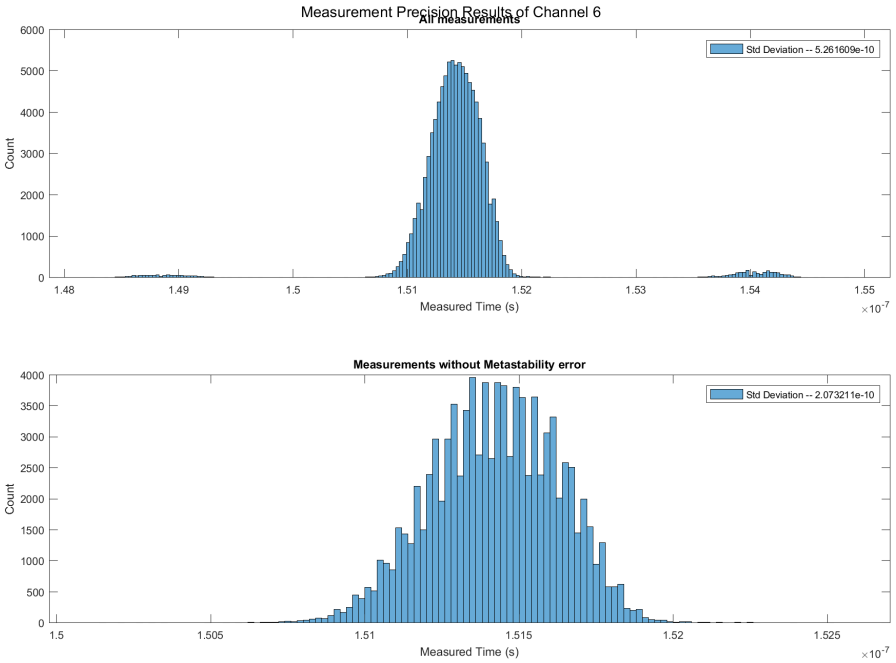

Results of Start Line of Channel 5
Measured Bin Width
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
DNL
LSB
Bin
INL
LSB
Bin

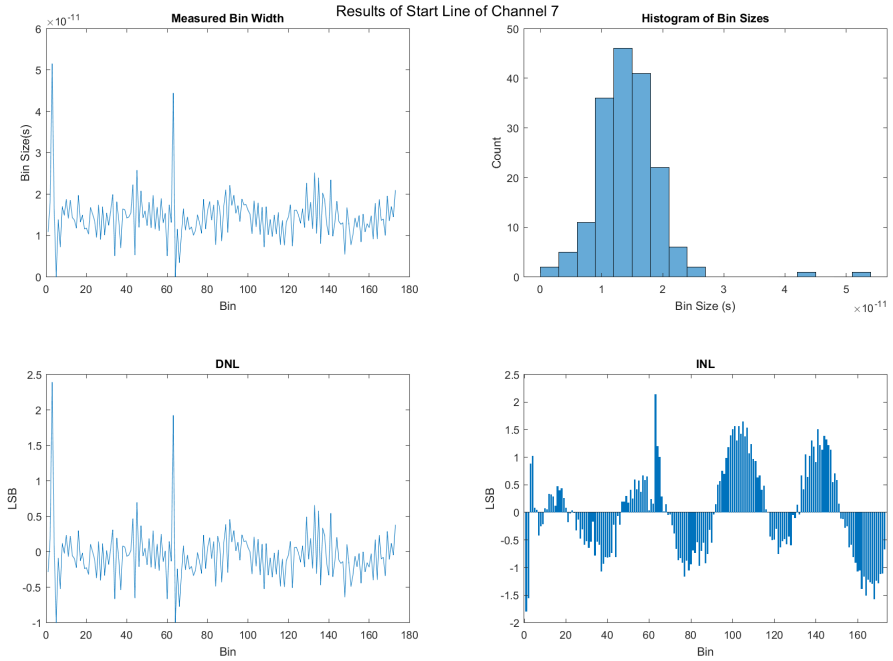

Results of Stop Line of Channel 5
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

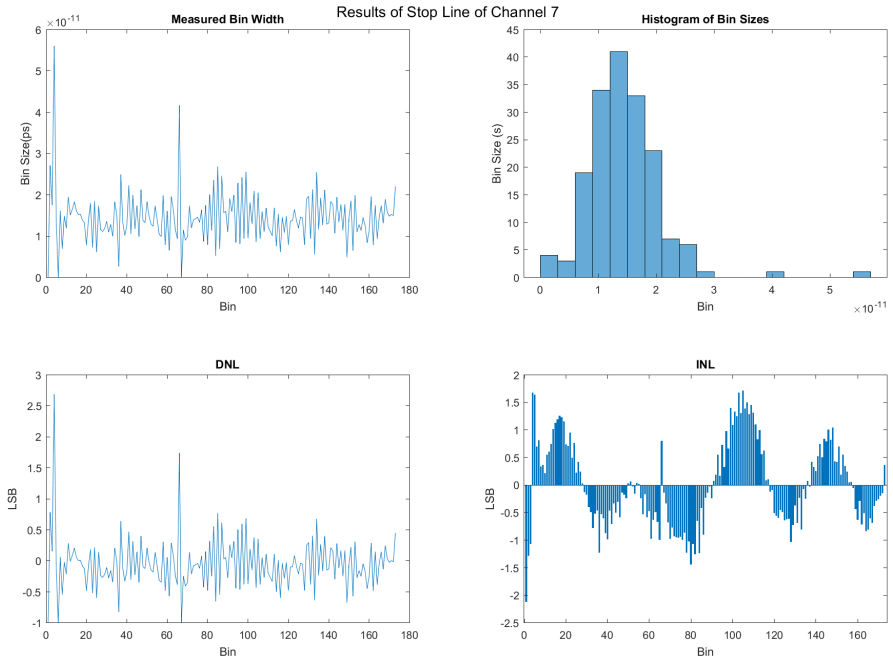

Measurement Precision Results of Channel 5
All measurements
Std Deviation -- 6.886755e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.127293e-10
Count
Measured Time (s)
×10^-7

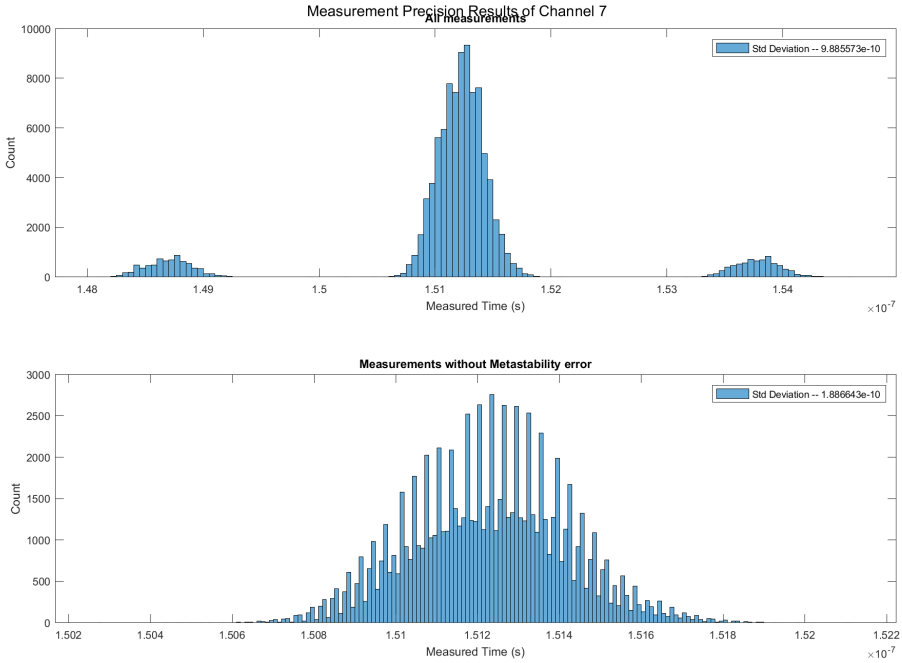

Results of Start Line of Channel 6
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

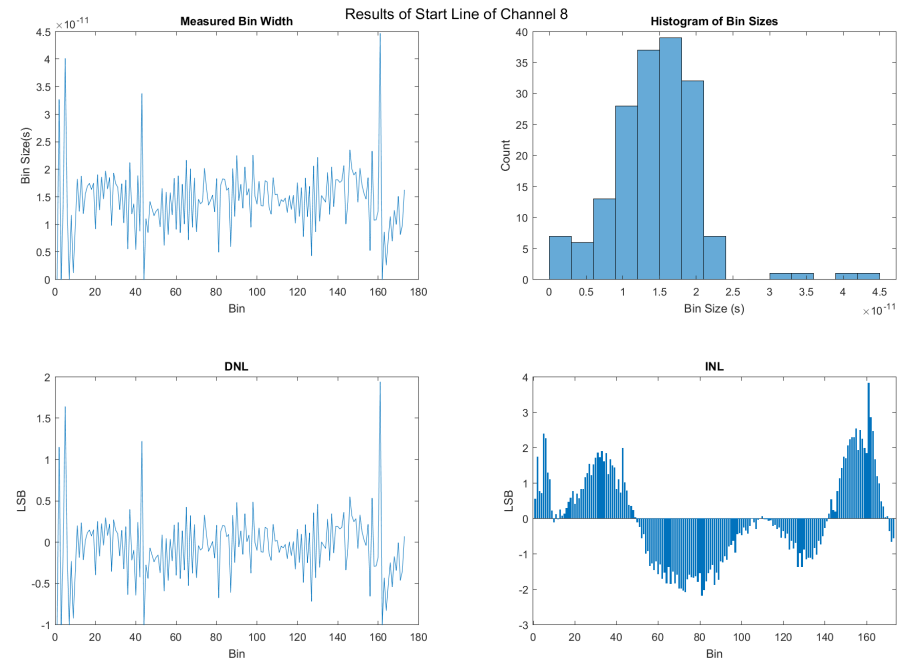

Results of Stop Line of Channel 6
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

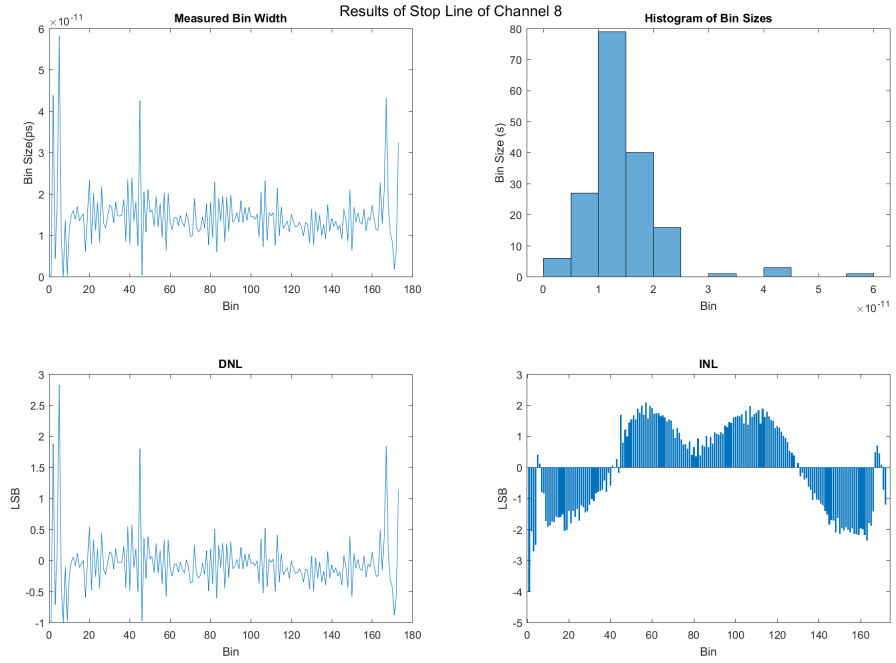

Measurement Precision Results of Channel 6
All measurements
Std Deviation -- 5.261609e-10
Count
Measured Time (s)
Measurements without Metastability error
Std Deviation -- 2.073211e-10
Count
Measured Time (s)

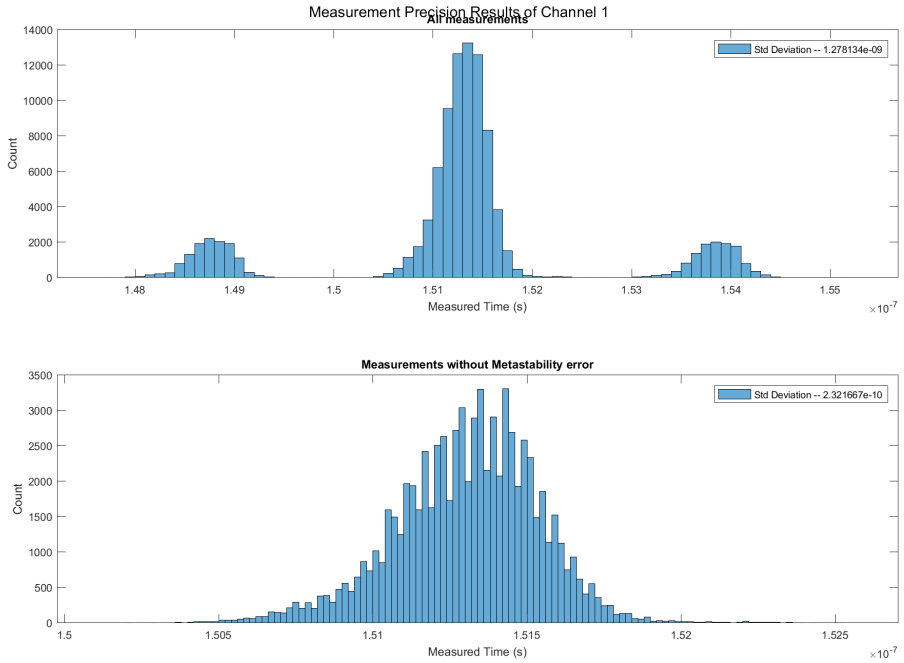

Results of Start Line of Channel 7
Measured Bin Width
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
DNL
LSB
Bin
INL
LSB
Bin

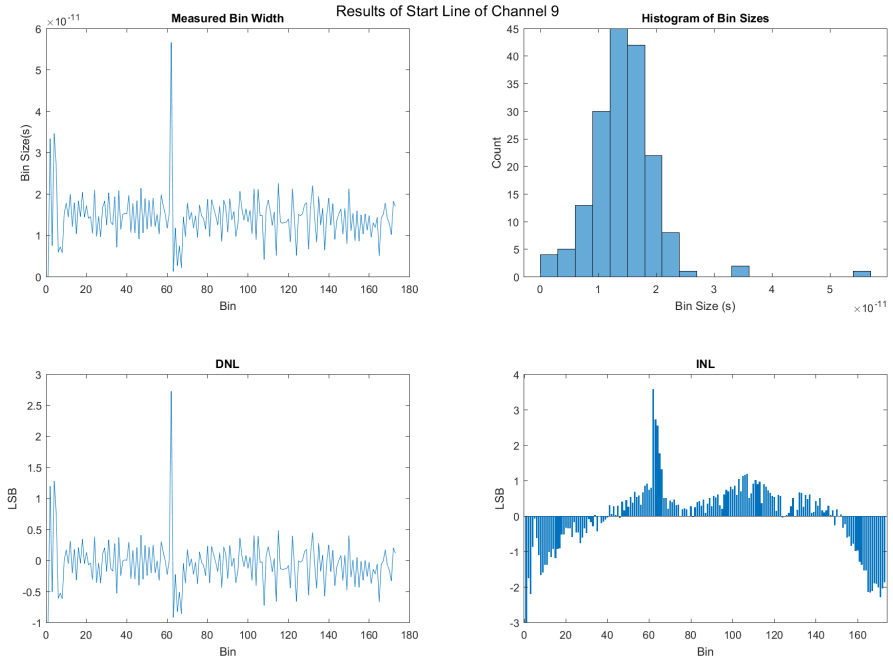

Results of Stop Line of Channel 7
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

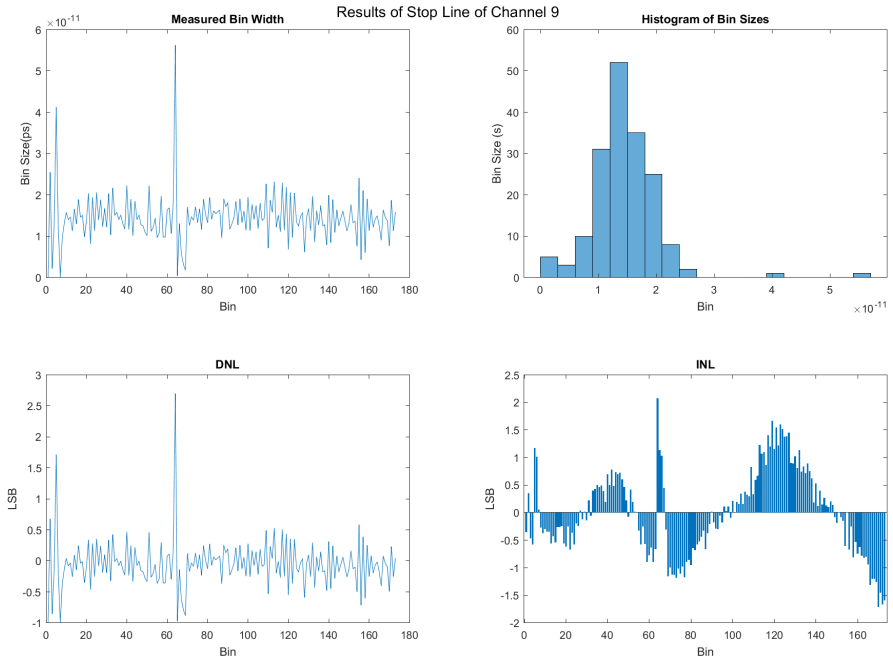

Measurement Precision Results of Channel 7
All measurements
Std Deviation -- 9.885573e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 1.886643e-10
Count
Measured Time (s)
×10^-7

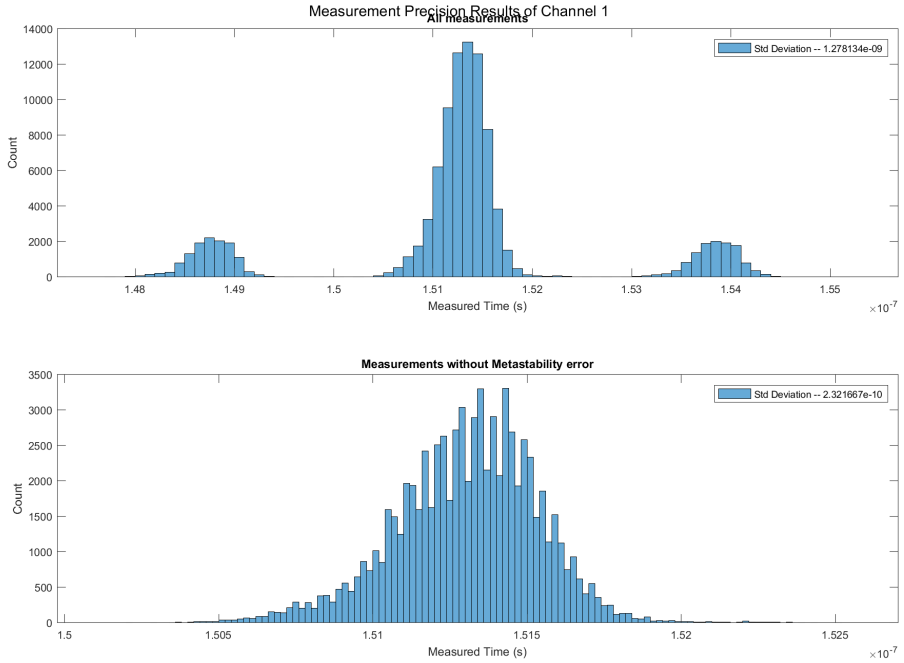

Results of Start Line of Channel 8
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

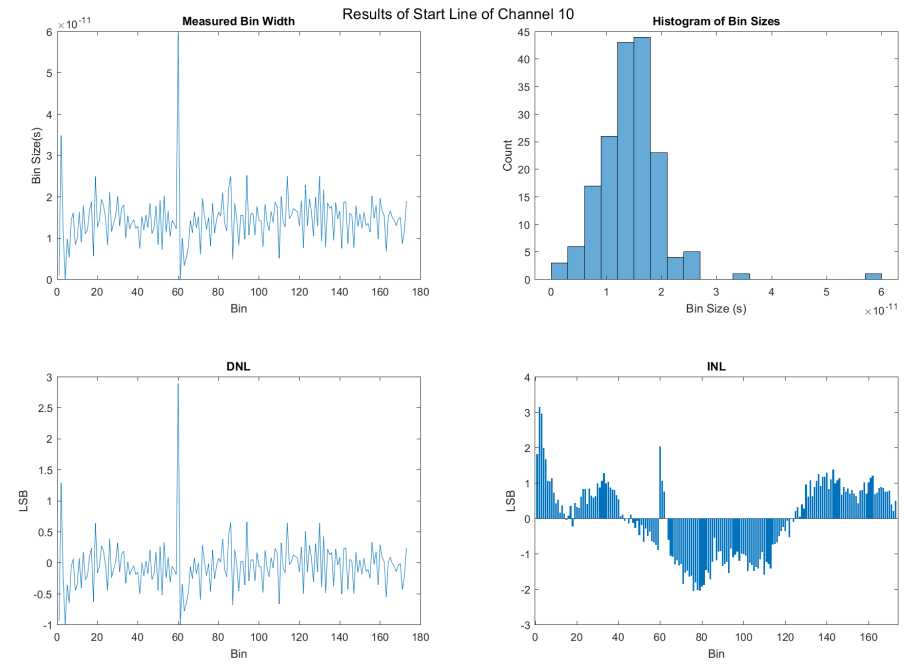

Results of Stop Line of Channel 8
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

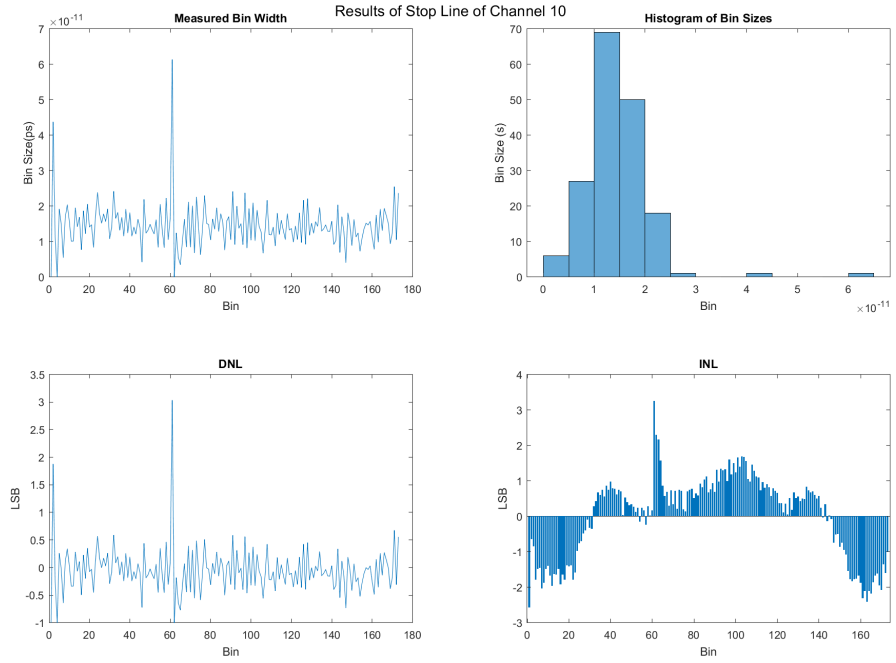

Measurement Precision Results of Channel 1
All measurements
Std Deviation -- 1.278134e-09
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.321667e-10
Count
Measured Time (s)
×10^-7

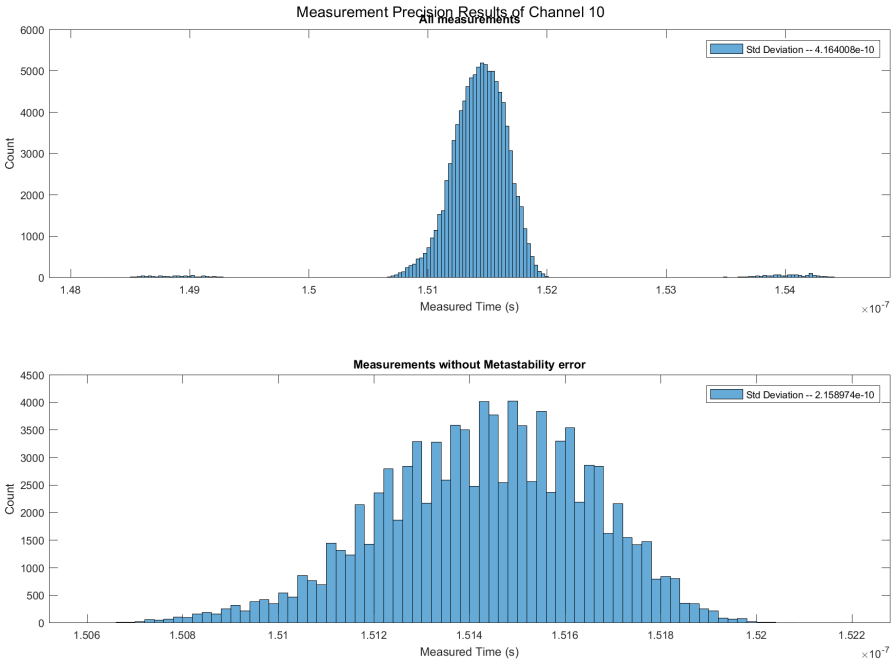

Results of Start Line of Channel 9
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

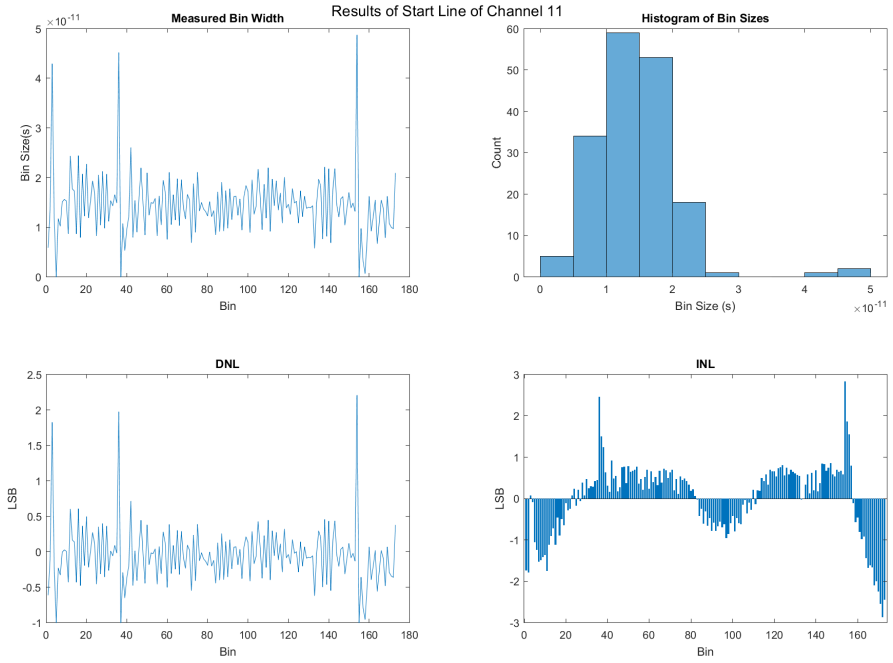

Results of Stop Line of Channel 9
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

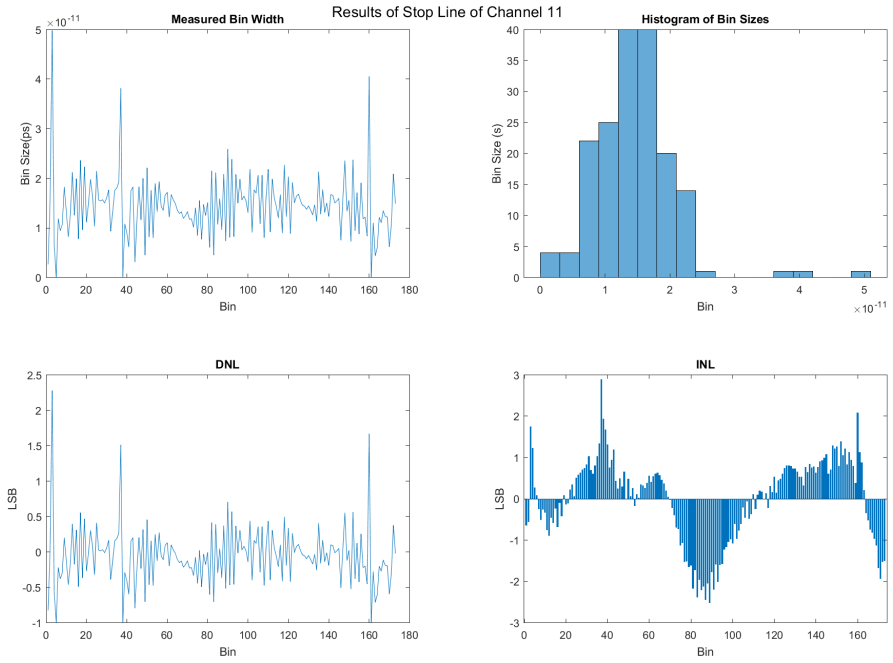

Measurement Precision Results of Channel 1
All measurements
Std Deviation -- 1.278134e-09
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.321667e-10
Count
Measured Time (s)
×10^-7

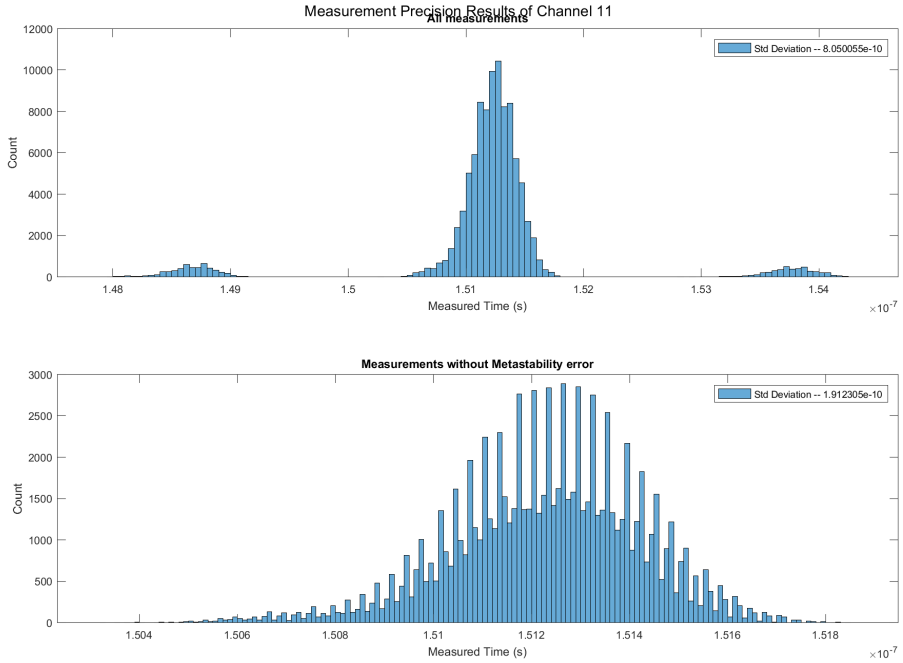

Results of Start Line of Channel 10
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

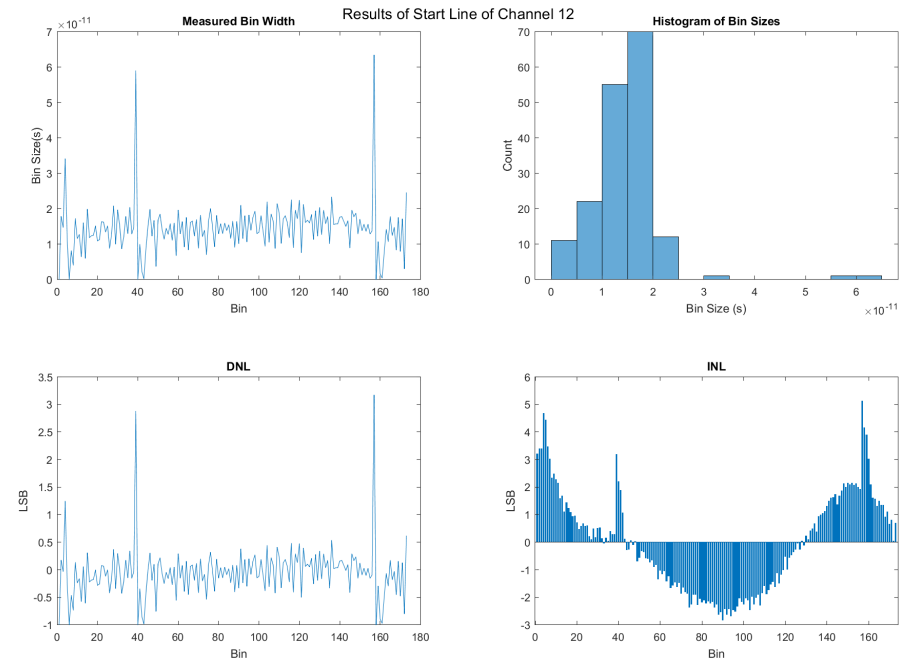

Results of Stop Line of Channel 10
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

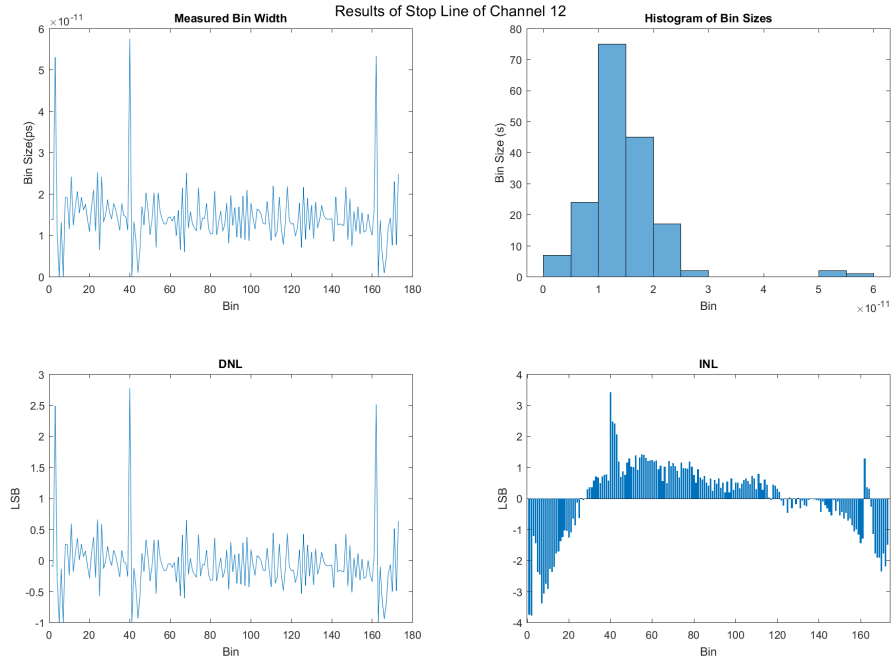

Measurement Precision Results of Channel 10
All measurements
Std Deviation -- 4.164008e-10
Count
Measured Time (s)
Measurements without Metastability error
Std Deviation -- 2.158974e-10
Count
Measured Time (s)

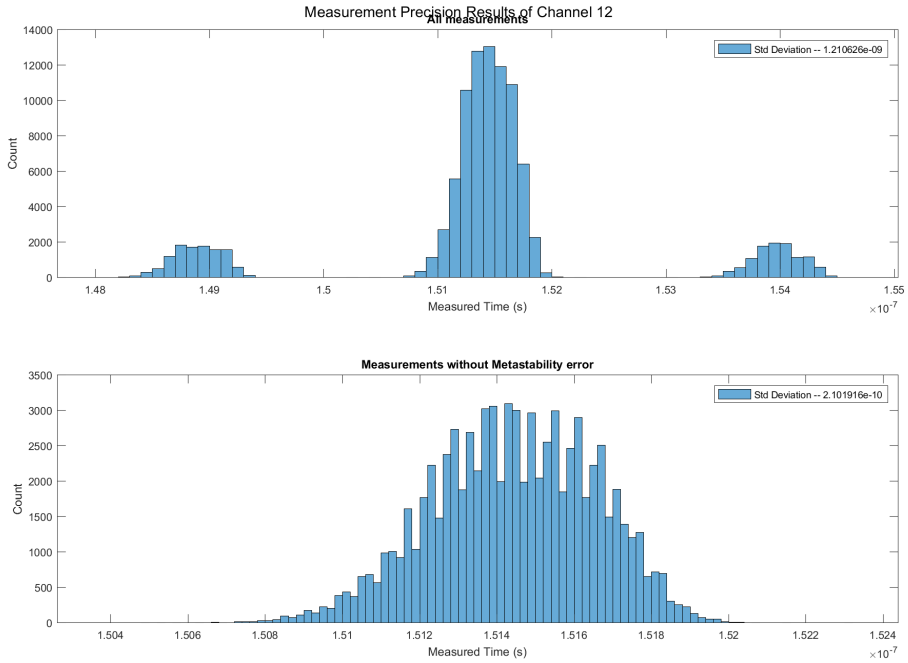

Results of Start Line of Channel 11
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

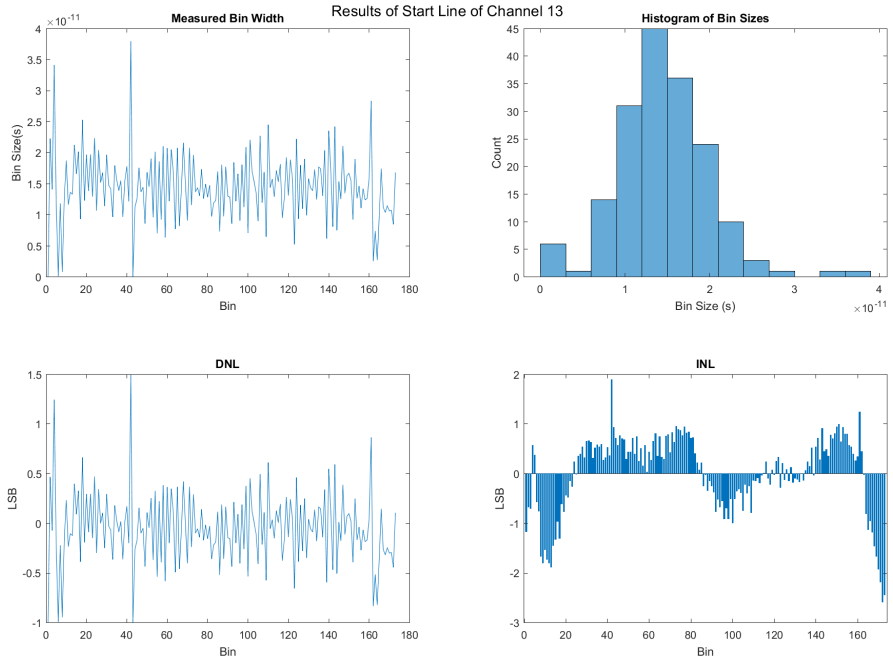

Results of Stop Line of Channel 11
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

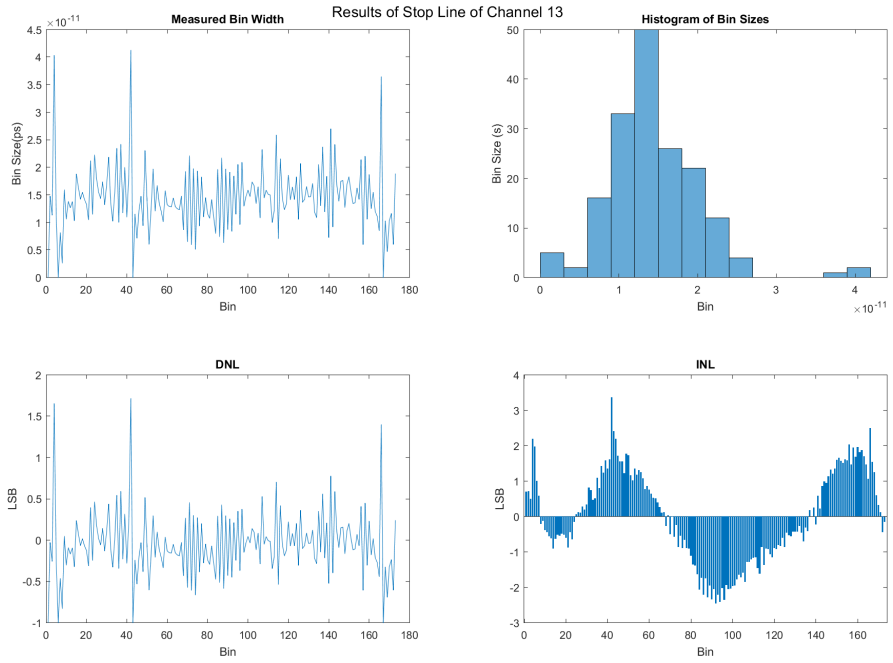

Measurement Precision Results of Channel 11
All measurements
Std Deviation -- 8.050055e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 1.912305e-10
Count
Measured Time (s)
×10^-7

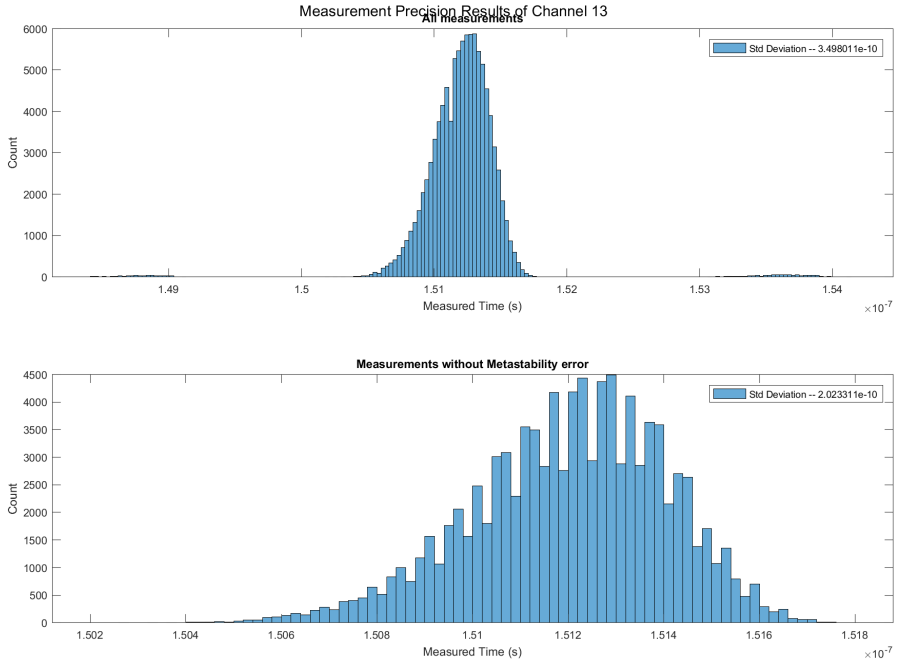

Results of Start Line of Channel 12
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

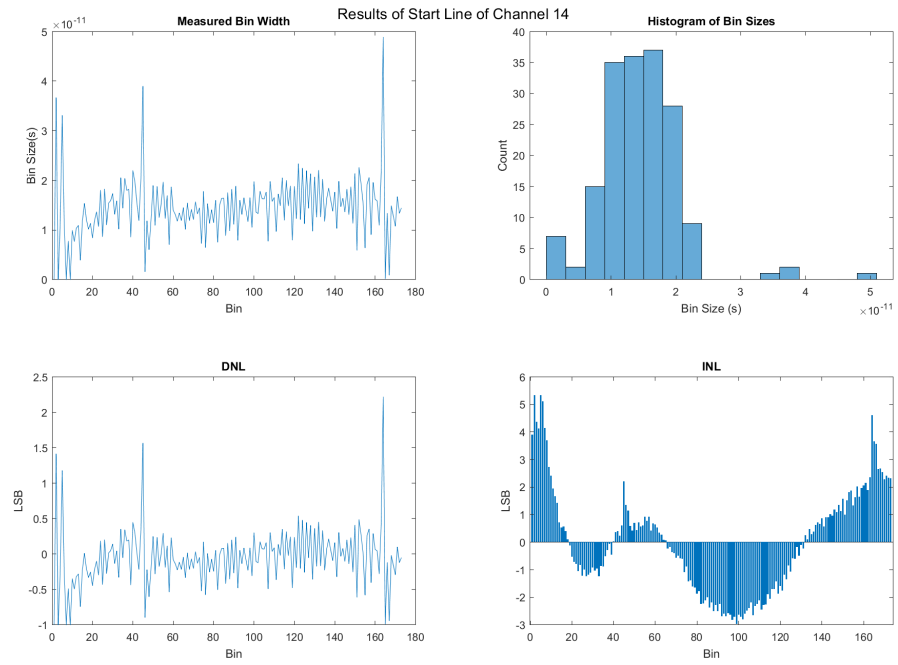

Results of Stop Line of Channel 12
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

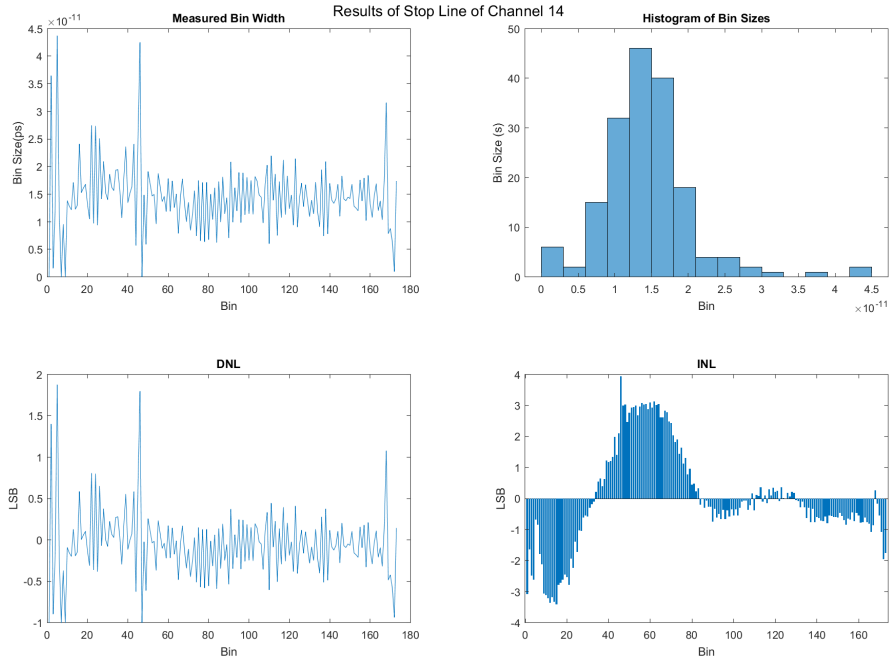

Measurement Precision Results of Channel 12
All measurements
Std Deviation -- 1.210626e-09
Count
Measured Time (s)
Measurements without Metastability error
Std Deviation -- 2.101916e-10
Count
Measured Time (s)

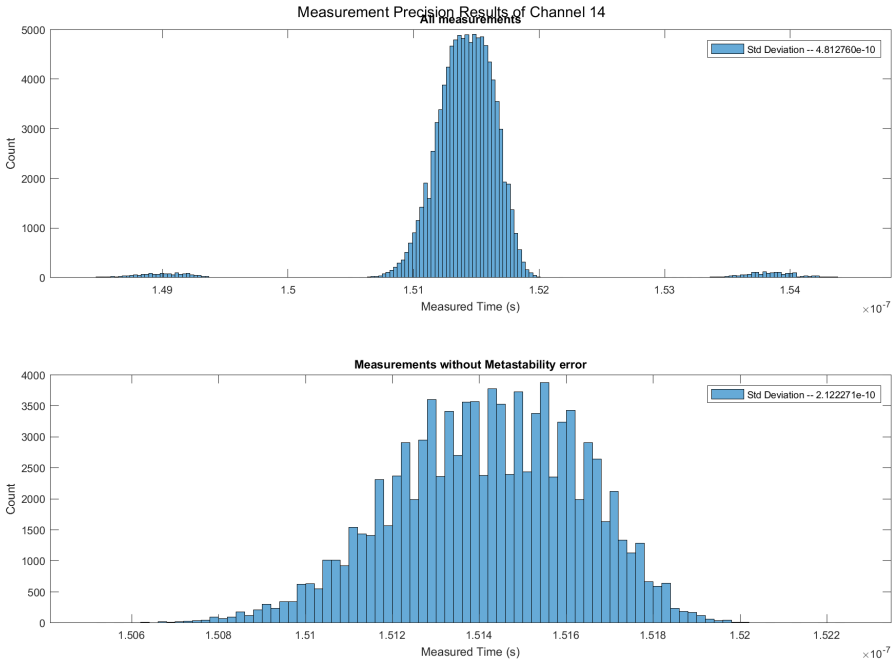

Results of Start Line of Channel 13
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

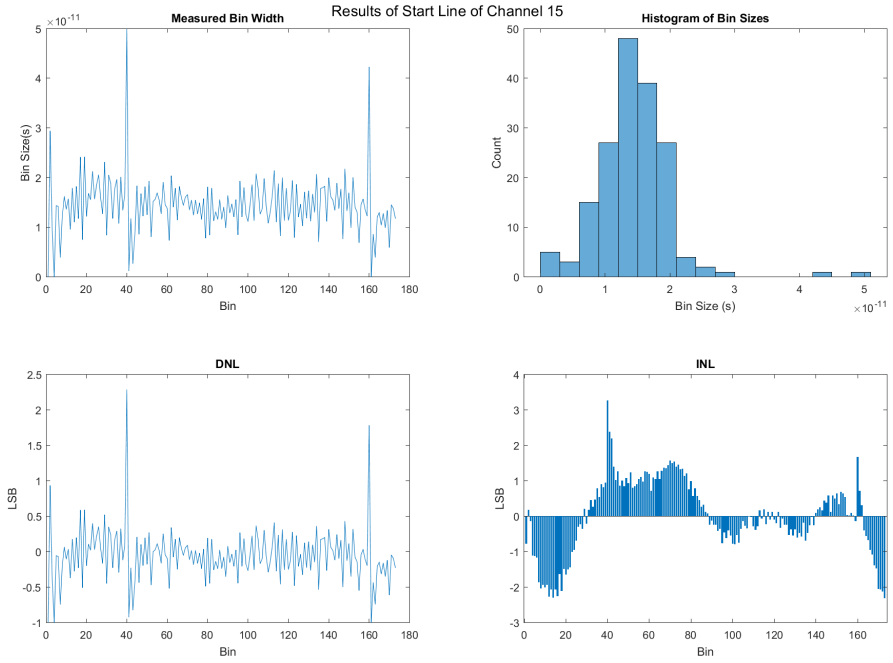

Results of Stop Line of Channel 13
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

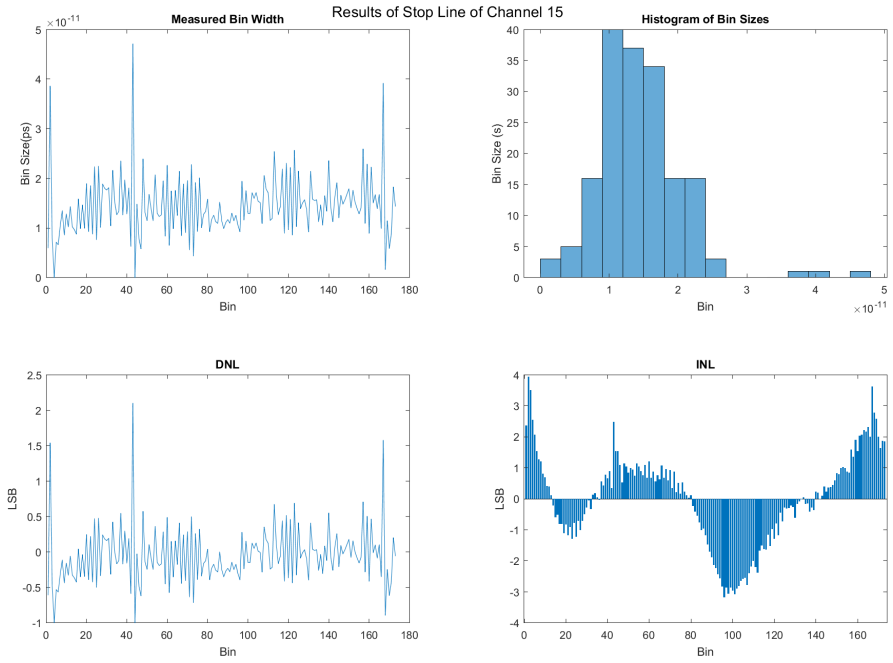

Measurement Precision Results of Channel 13
All measurements
Std Deviation -- 3.498011e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.023311e-10
Count
Measured Time (s)
×10^-7

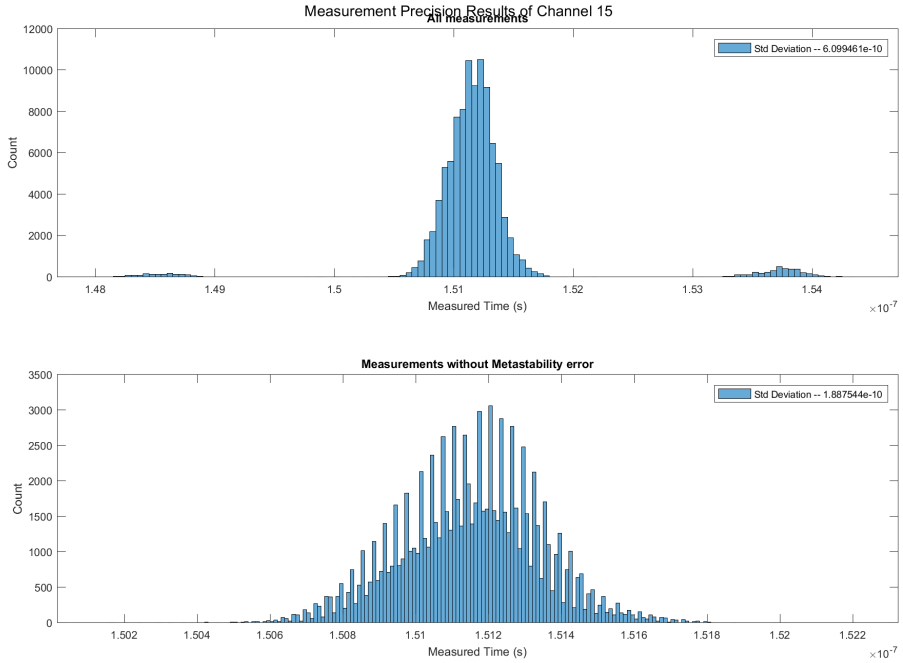

Results of Start Line of Channel 14
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

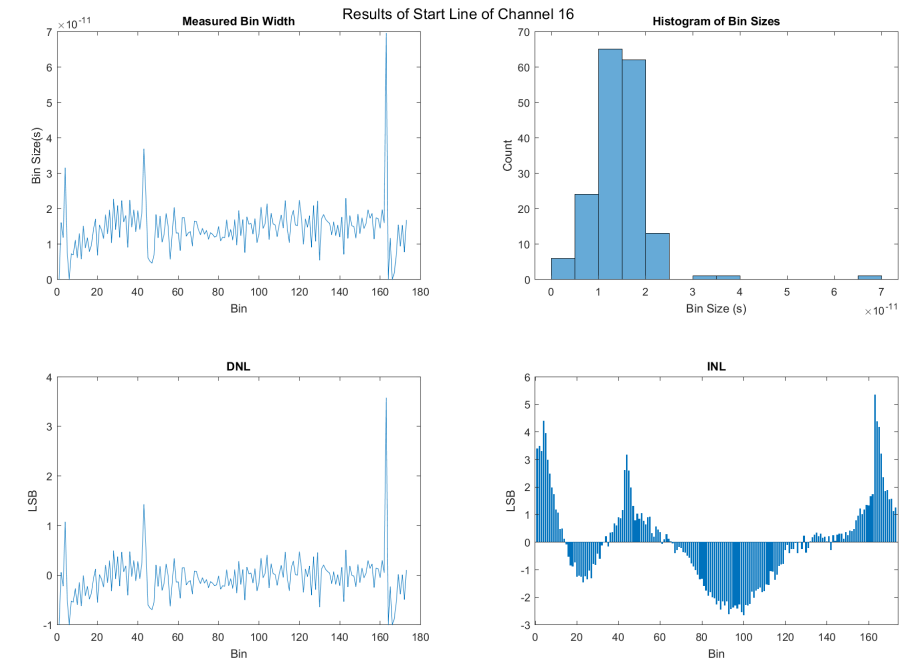

Results of Stop Line of Channel 14
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

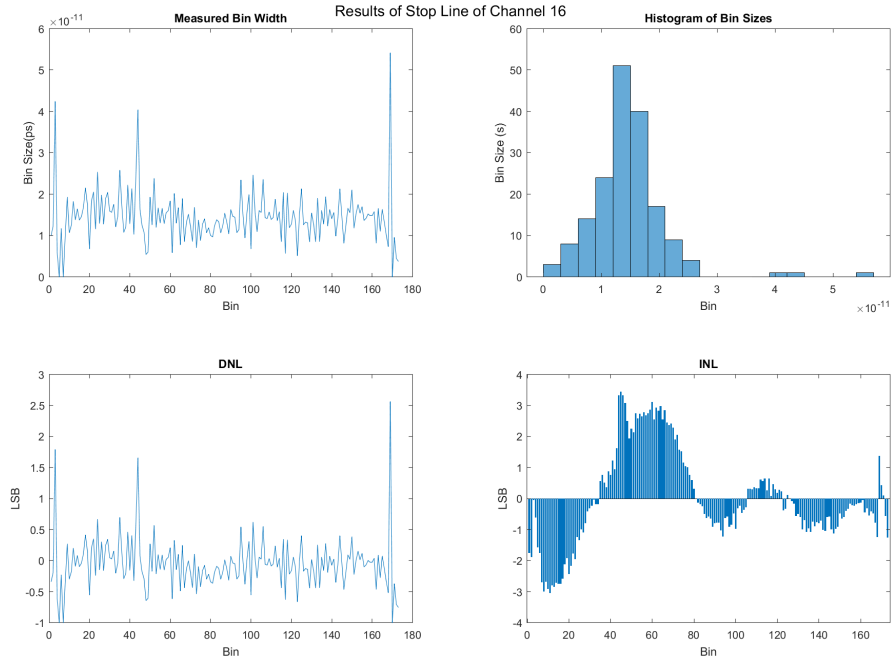

Measurement Precision Results of Channel 14
All measurements
Std Deviation -- 4.812760e-10
Count
Measured Time (s)
Measurements without Metastability error
Std Deviation -- 2.122271e-10
Count
Measured Time (s)

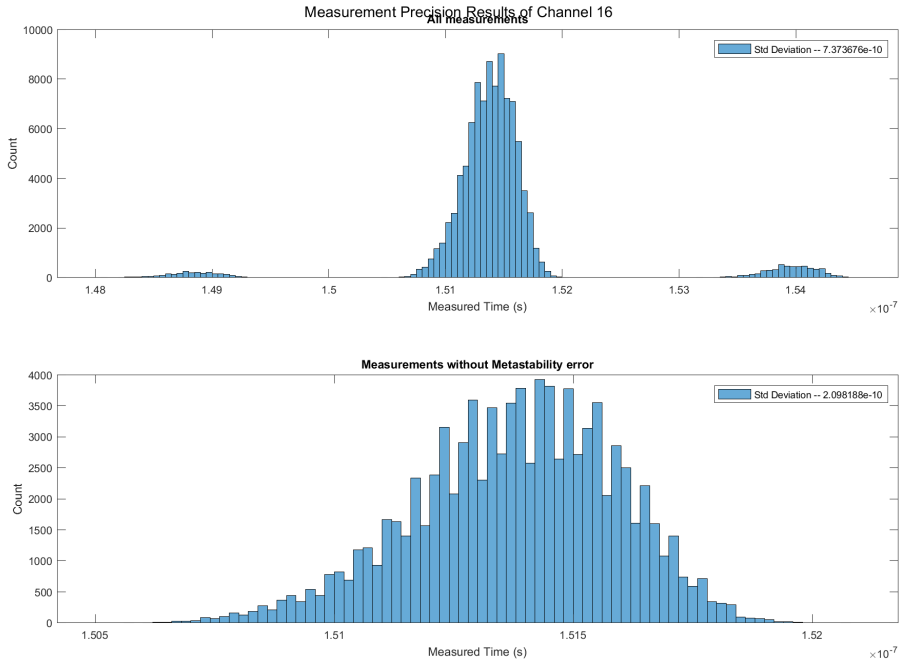

Results of Start Line of Channel 15
Measured Bin Width
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
DNL
LSB
Bin
INL
LSB
Bin


Results of Stop Line of Channel 15
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

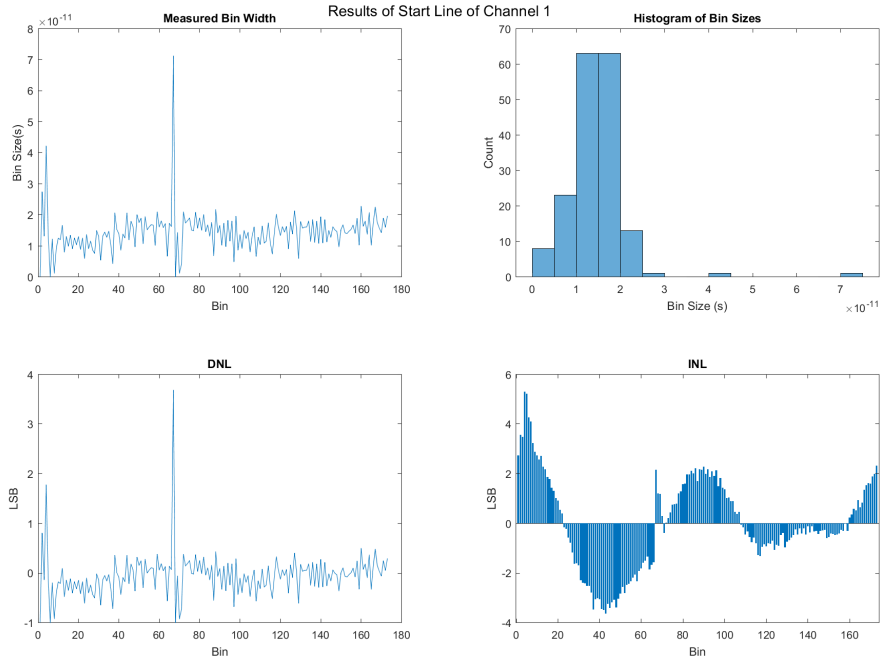

Measurement Precision Results of Channel 15
All measurements
Std Deviation -- 6.099461e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 1.887544e-10
Count
Measured Time (s)
×10^-7

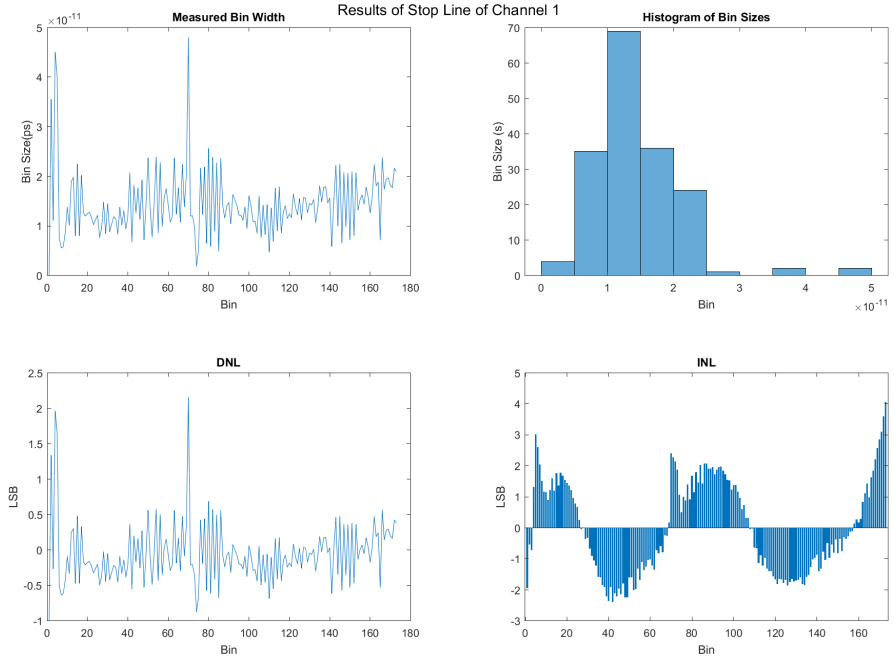

Results of Start Line of Channel 16
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

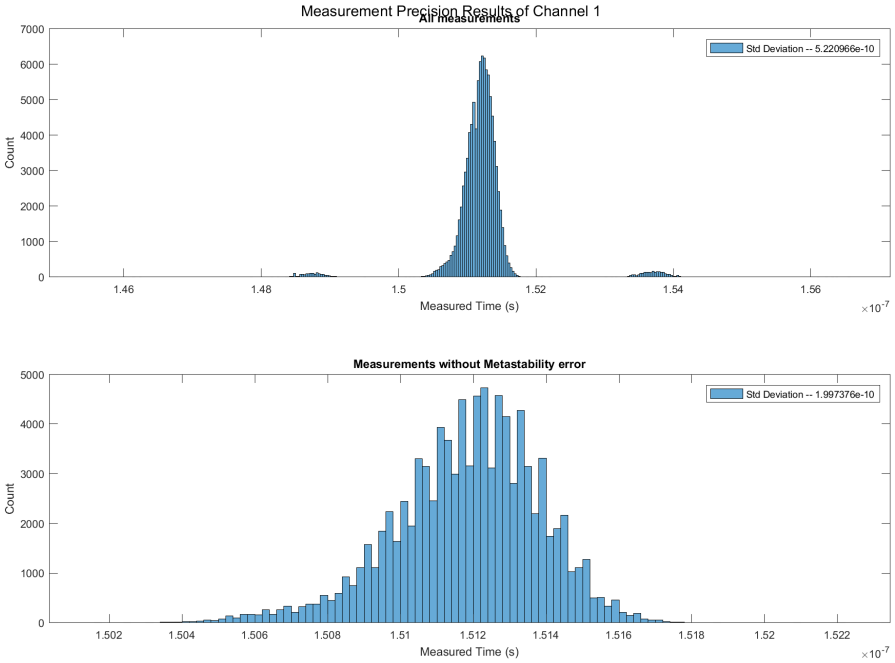


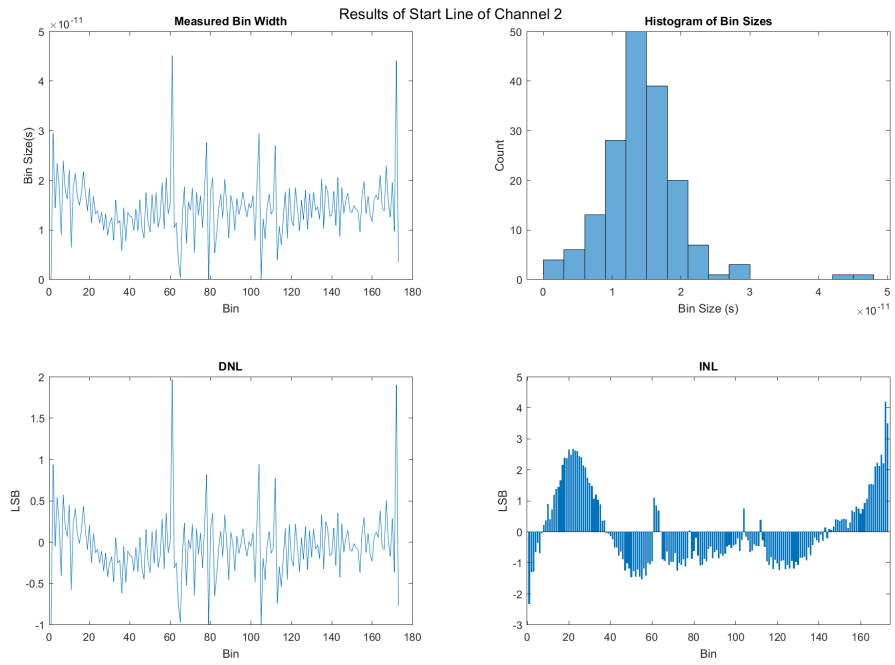


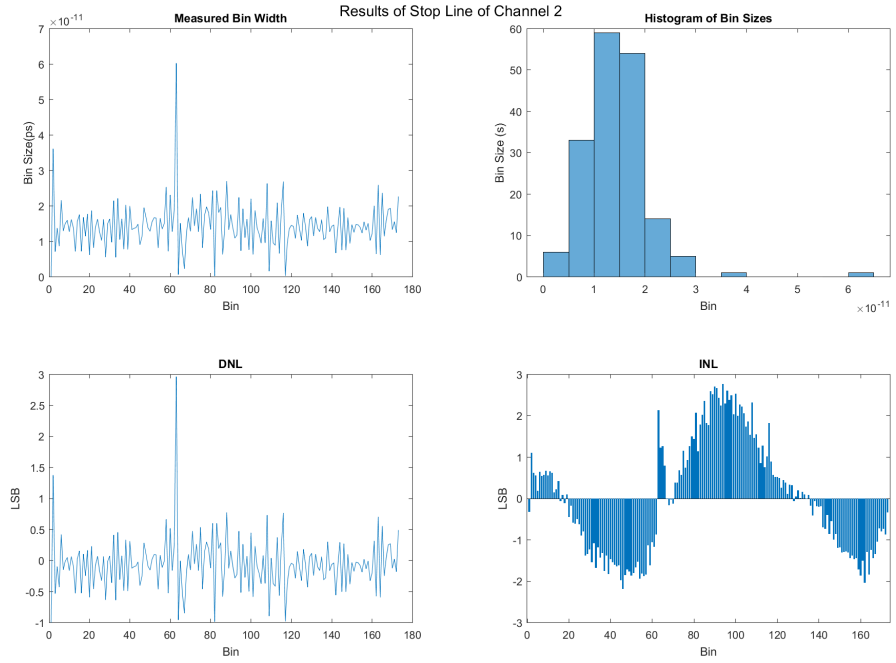
8-Channel Implementation

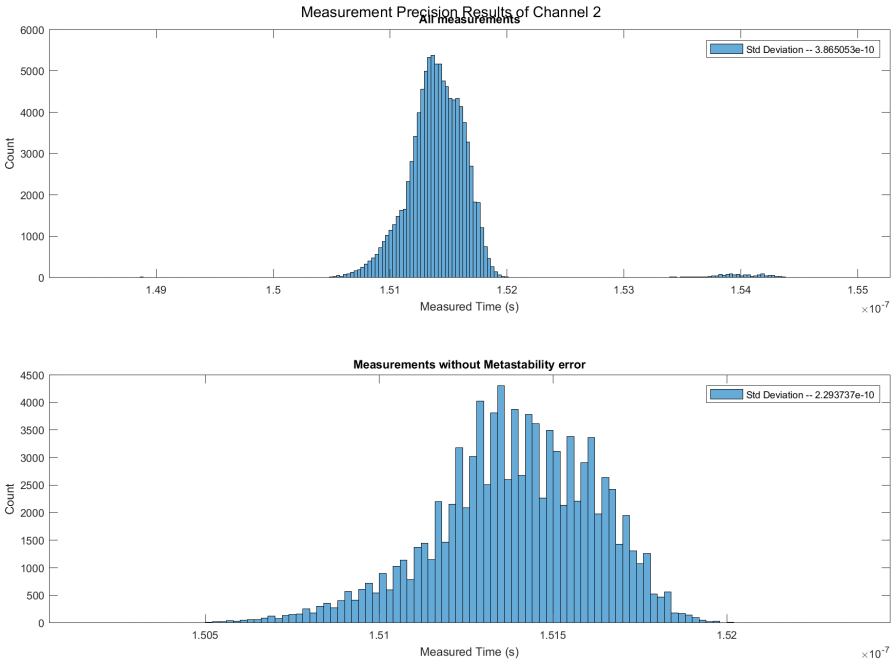

Results of Start Line of Channel 1
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

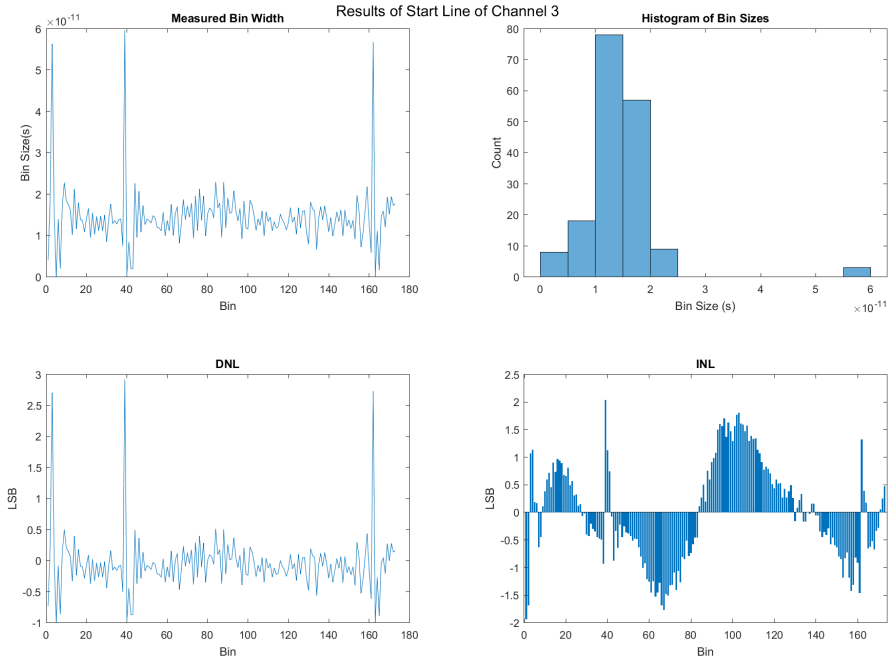

Results of Stop Line of Channel 1
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

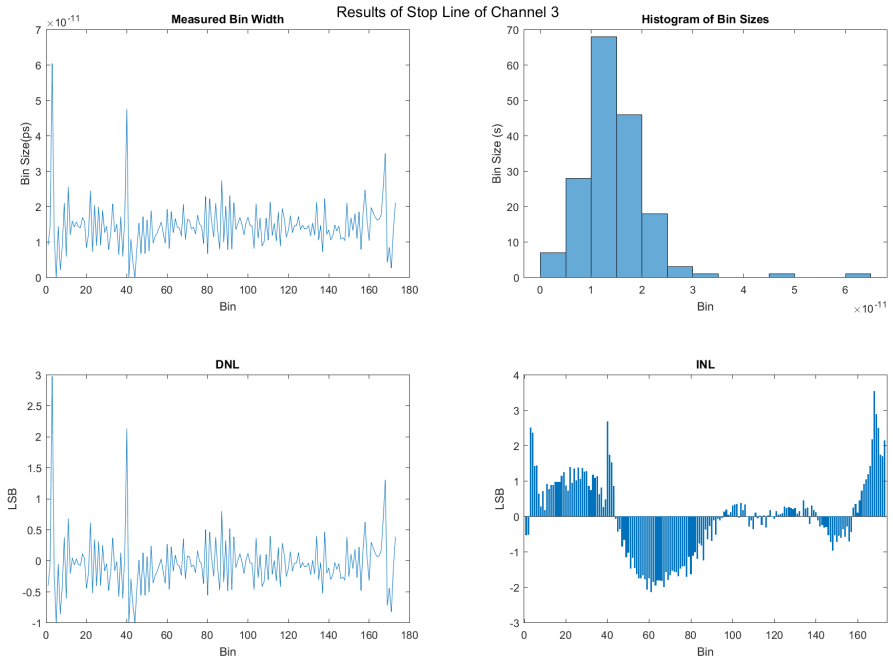

Measurement Precision Results of Channel 1
All measurements
Std Deviation -- 5.220966e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 1.997376e-10
Count
Measured Time (s)
×10^-7

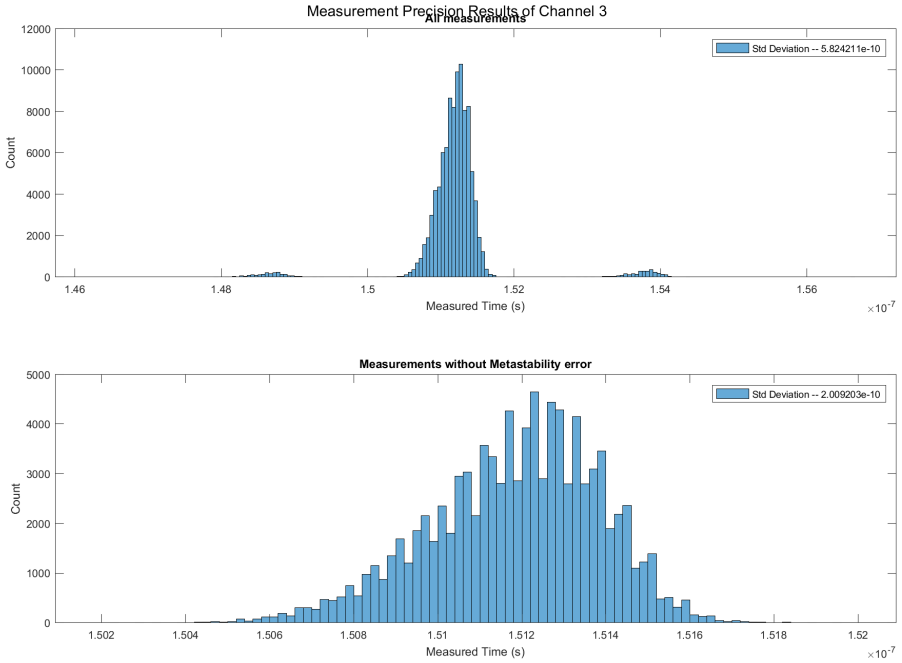

Results of Start Line of Channel 2
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

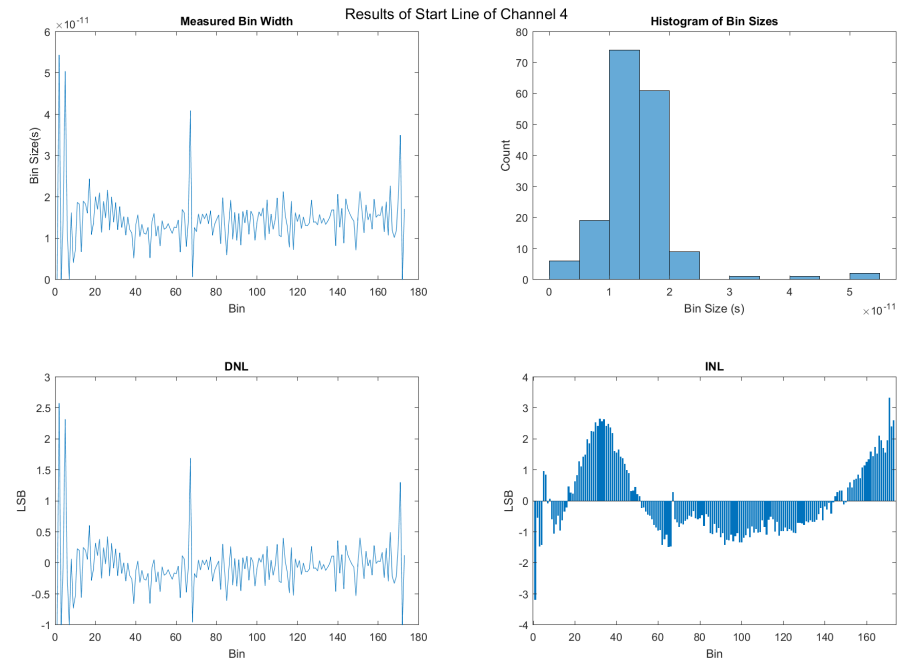

Results of Stop Line of Channel 2
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

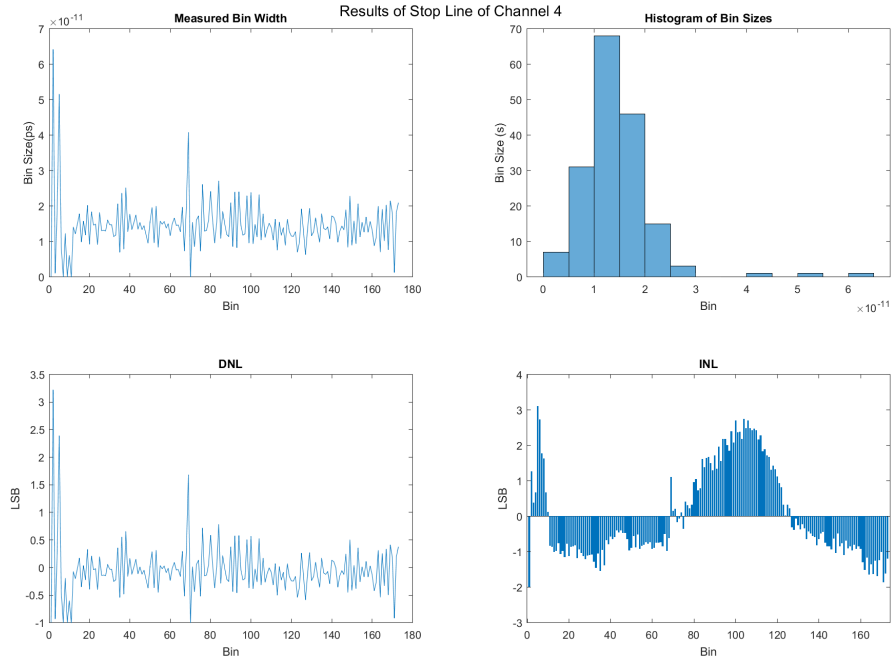

Measurement Precision Results of Channel 2
All measurements
Std Deviation -- 3.865053e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.293737e-10
Count
Measured Time (s)
×10^-7

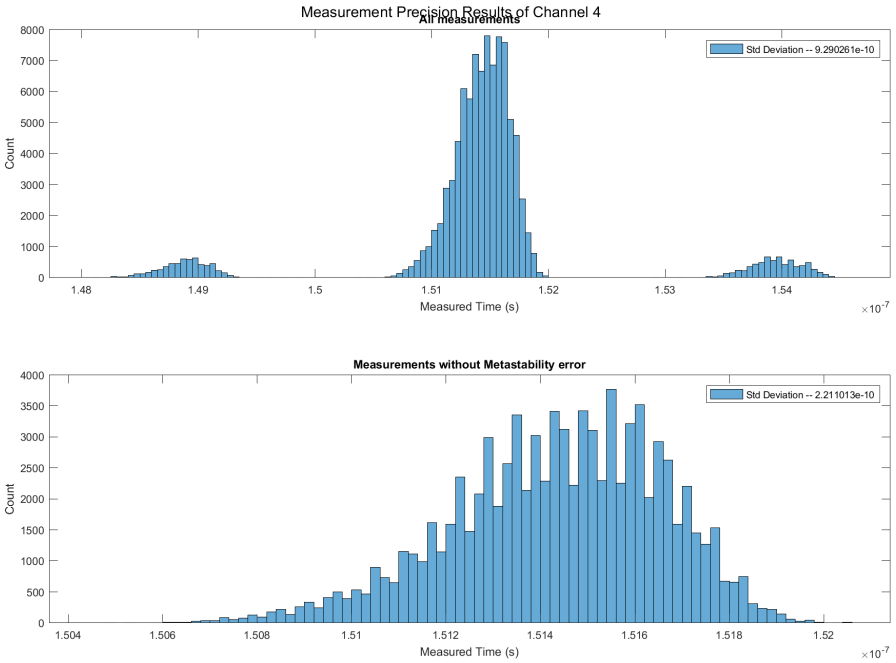

Results of Start Line of Channel 3
Measured Bin Width
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
DNL
LSB
Bin
INL
LSB
Bin


Results of Stop Line of Channel 3
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

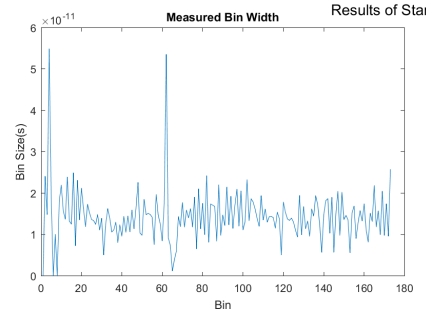

Measurement Precision Results of Channel 3
All measurements
Std Deviation -- 5.824211e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.009203e-10
Count
Measured Time (s)
×10^-7

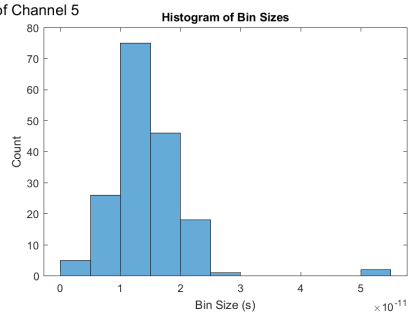

Results of Start Line of Channel 4
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

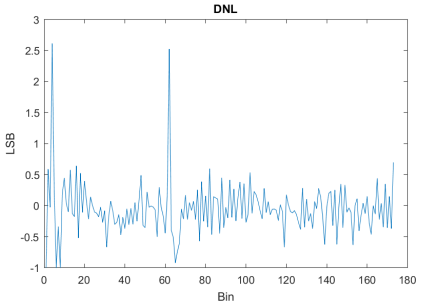

Results of Stop Line of Channel 4
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

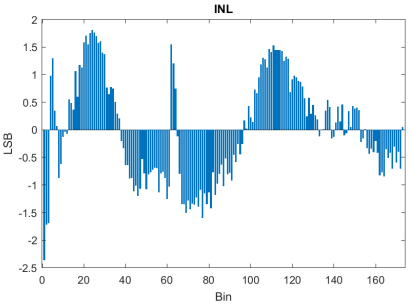

Measurement Precision Results of Channel 4
All measurements
Std Deviation -- 9.290261e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.211013e-10
Count
Measured Time (s)
×10^-7

Results of Start Line of Channel 5

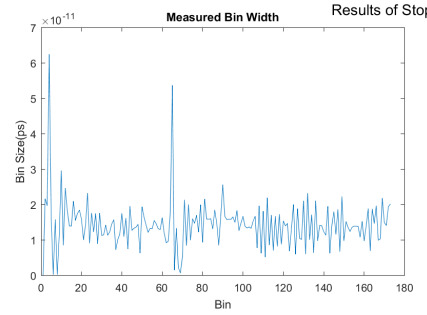


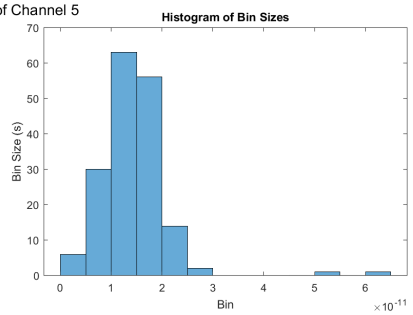


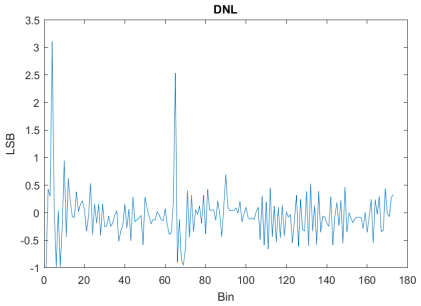


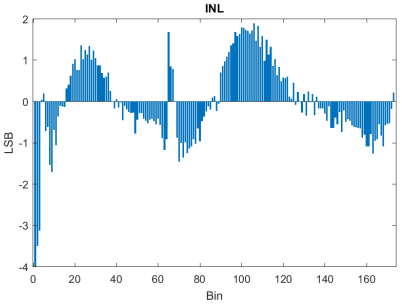


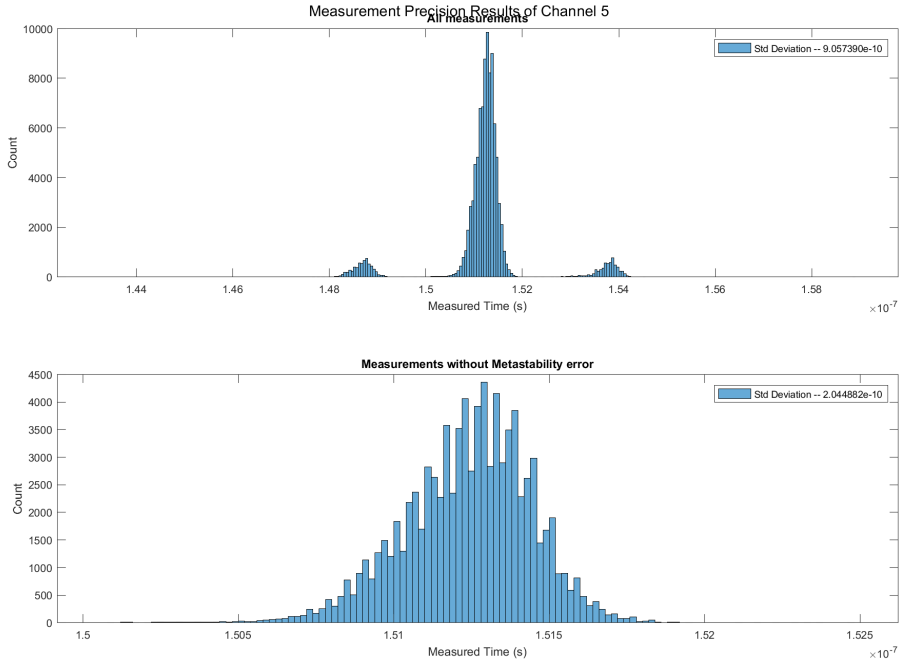
Results of Stop Line of Channel 5

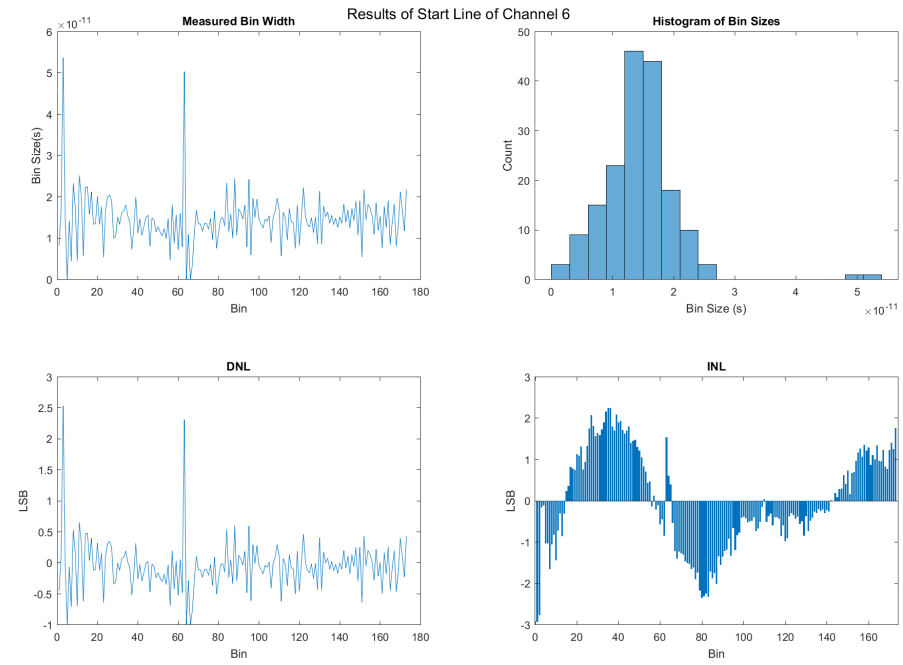


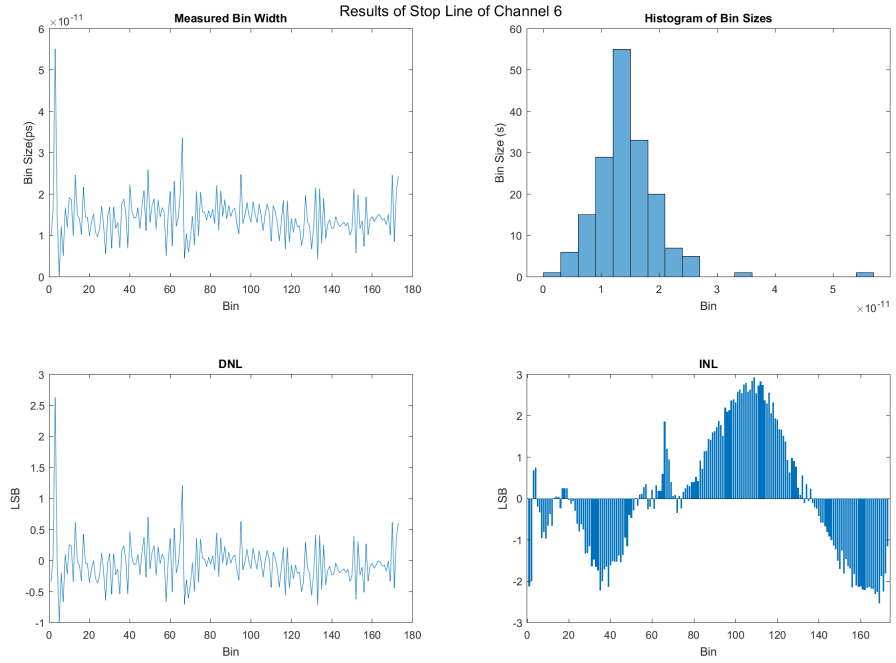


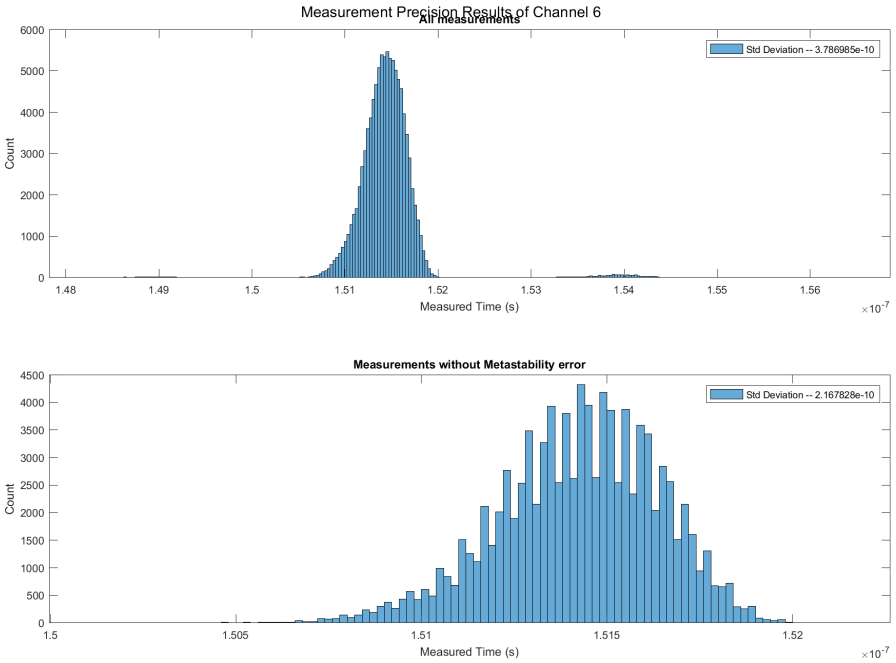




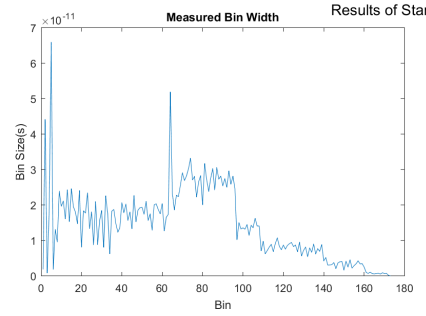

Measurement Precision Results of Channel 5
All measurements
Std Deviation -- 9.057390e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.044882e-10
Count
Measured Time (s)
×10^-7

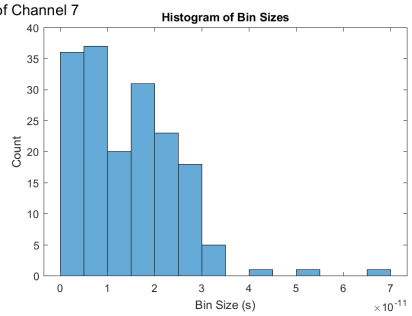

Results of Start Line of Channel 6
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

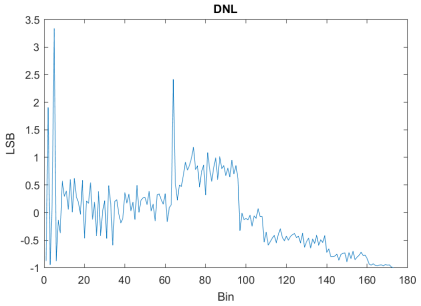

Results of Stop Line of Channel 6
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

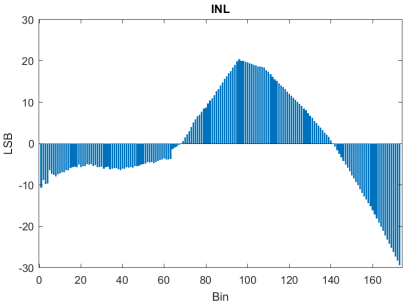

Measurement Precision Results of Channel 6
All measurements
Std Deviation -- 3.786985e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.167828e-10
Count
Measured Time (s)
×10^-7

Results of Start Line of Channel 7

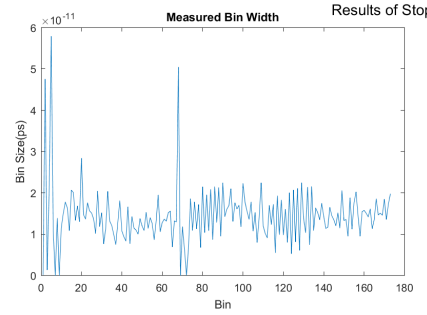


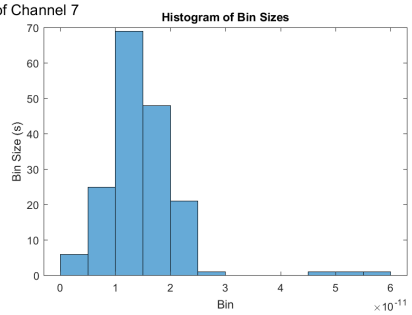


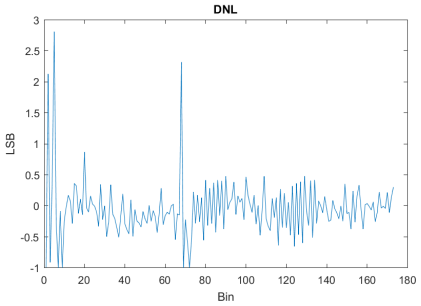


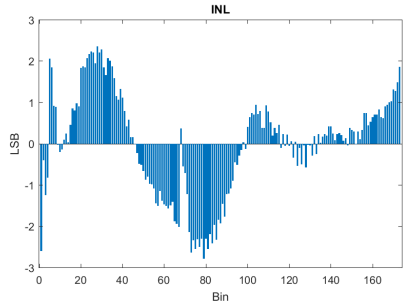


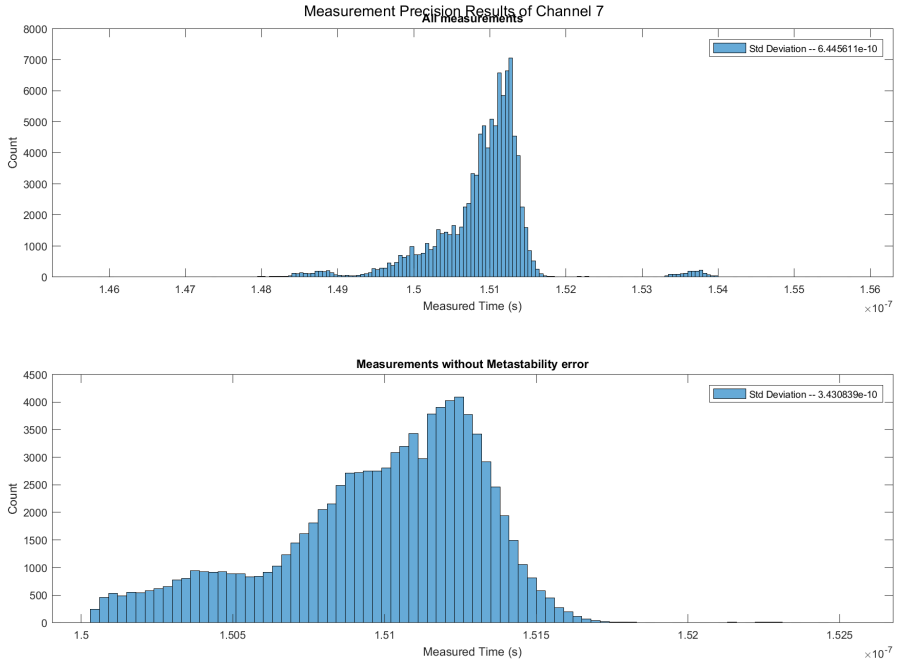
Results of Stop Line of Channel 7

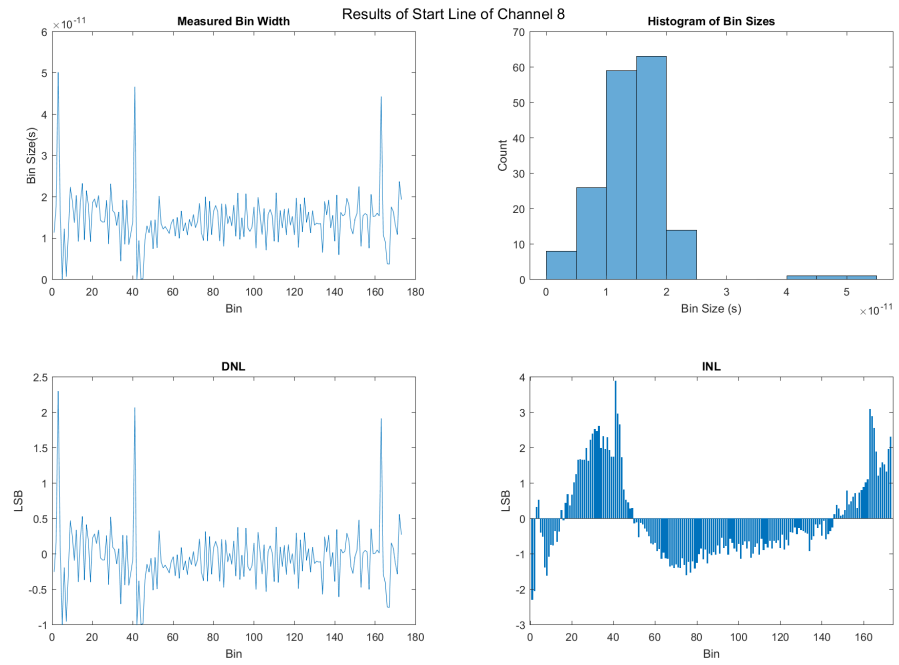


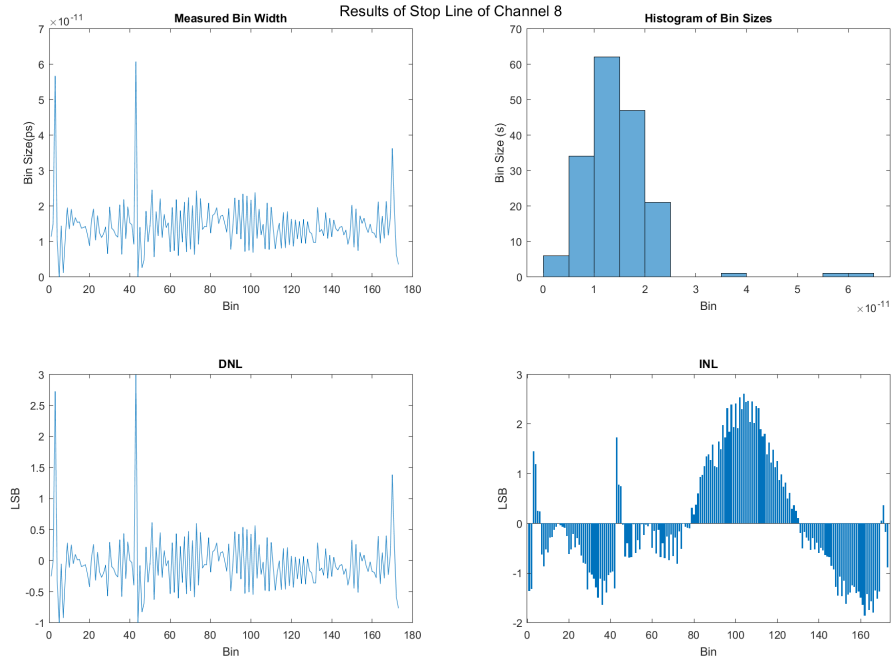


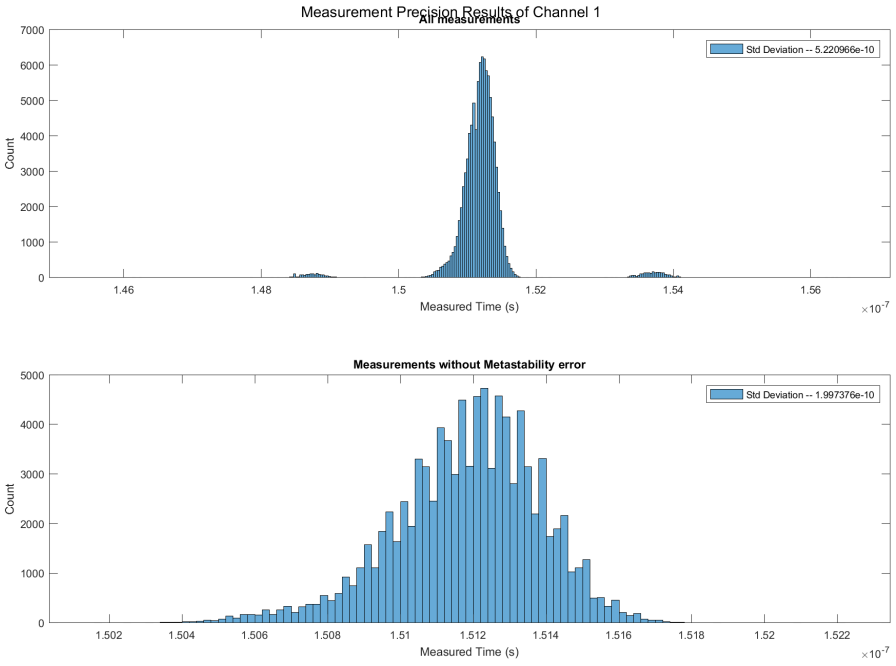


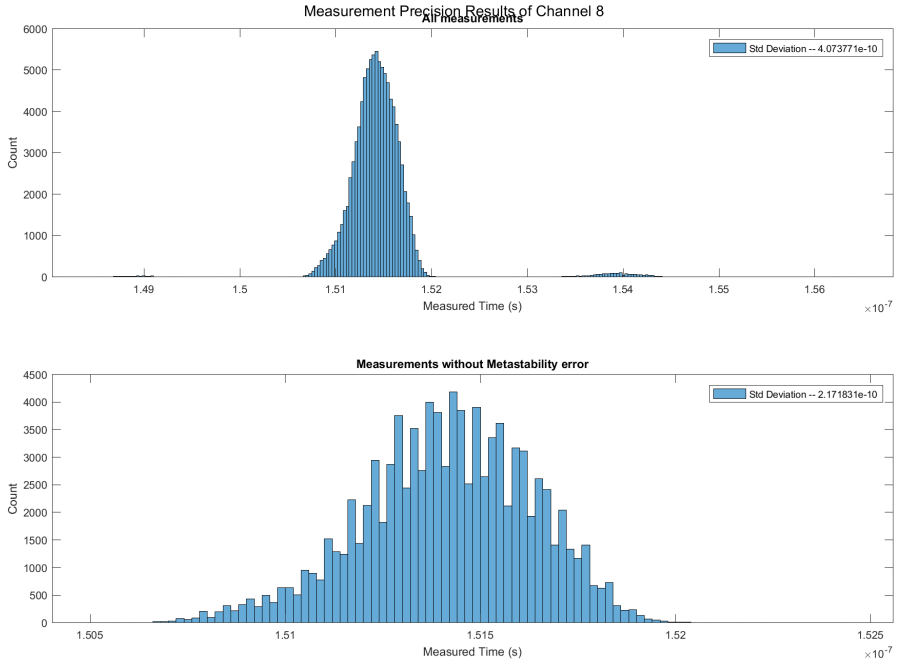



Measurement Precision Results of Channel 7
All measurements
Std Deviation -- 6.445611e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 3.430839e-10
Count
Measured Time (s)
×10^-7

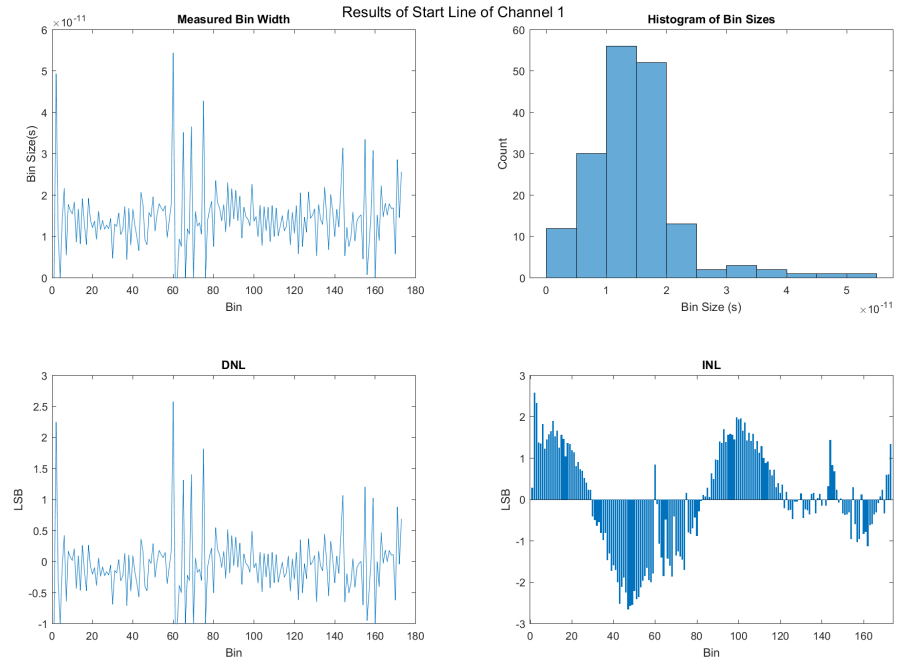

Results of Start Line of Channel 8
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

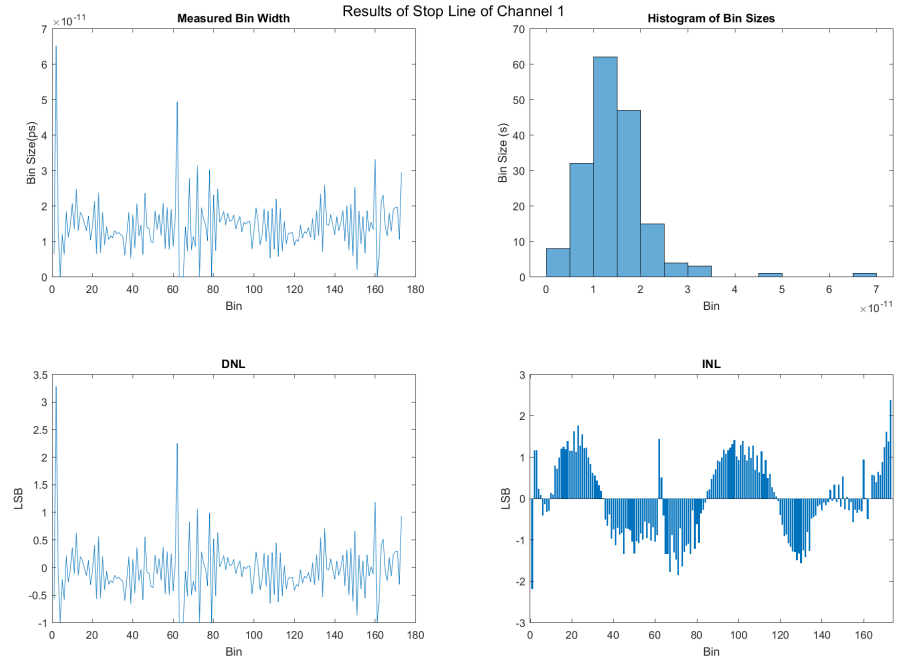

Results of Stop Line of Channel 8
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin

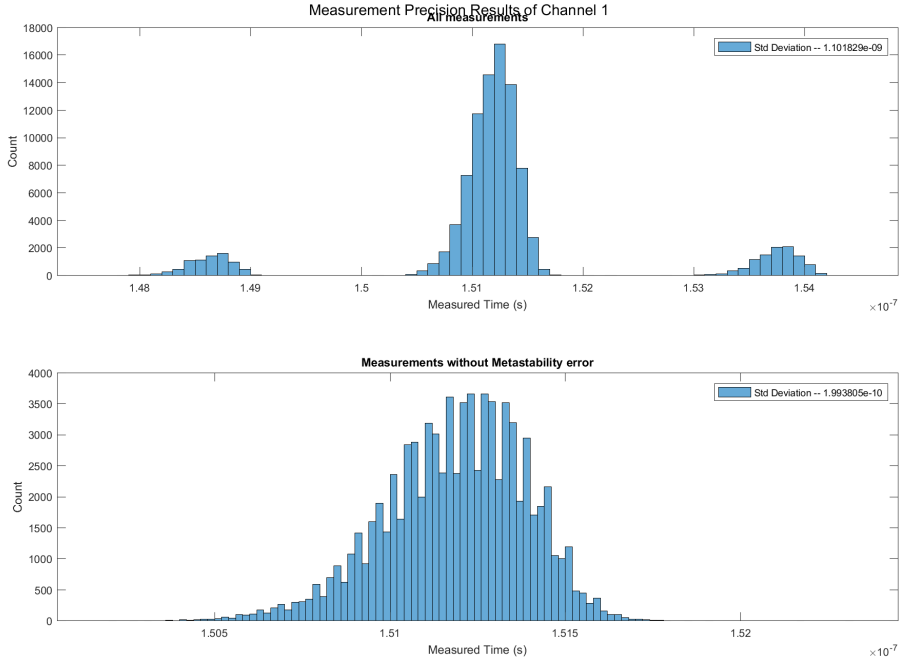

Measurement Precision Results of Channel 1
All measurements
Std Deviation -- 5.220966e-10
Count
Measured Time (s)
Measurements without Metastability error
Std Deviation -- 1.997376e-10
Count
Measured Time (s)

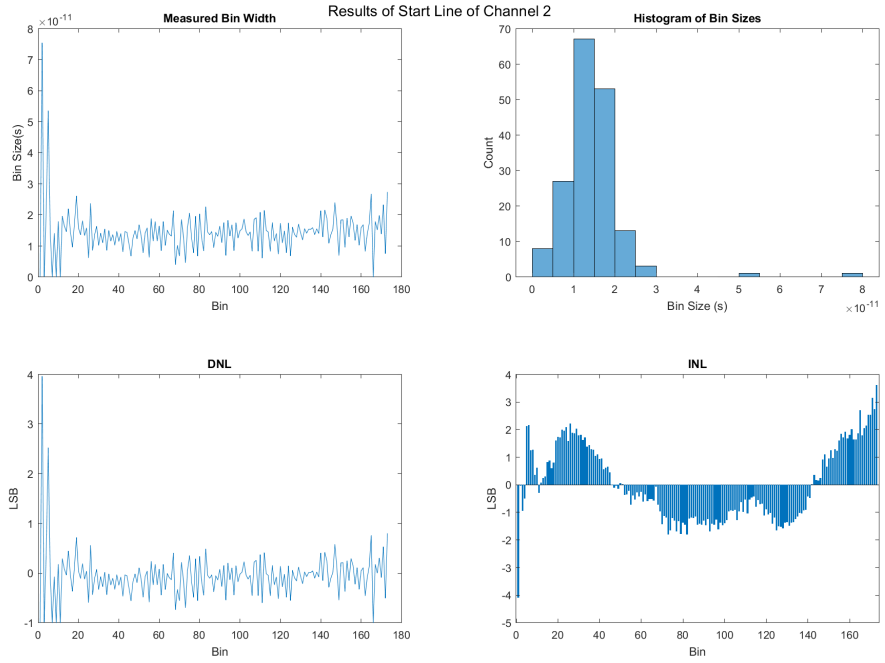


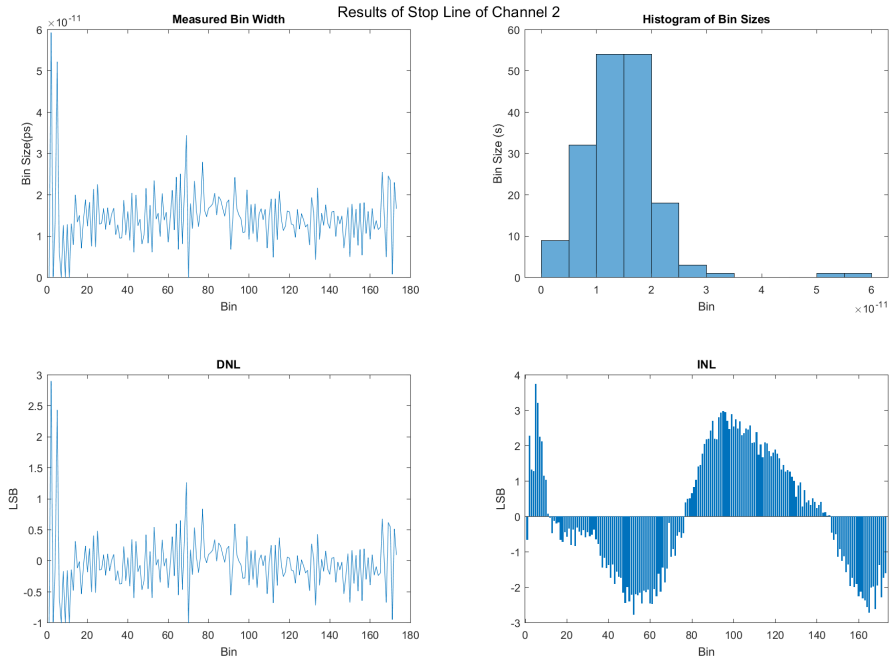
4-Channel Implementation

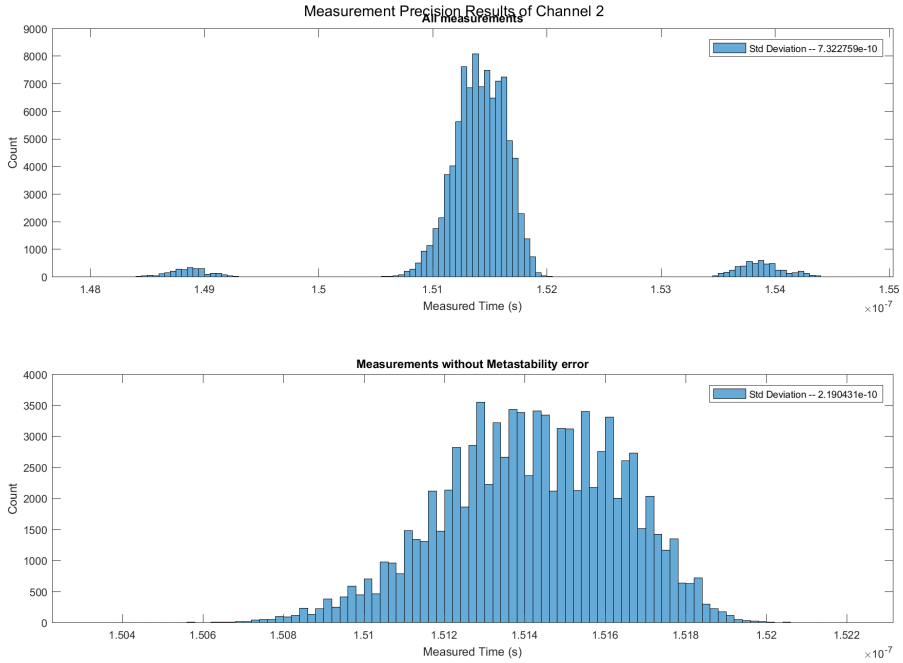


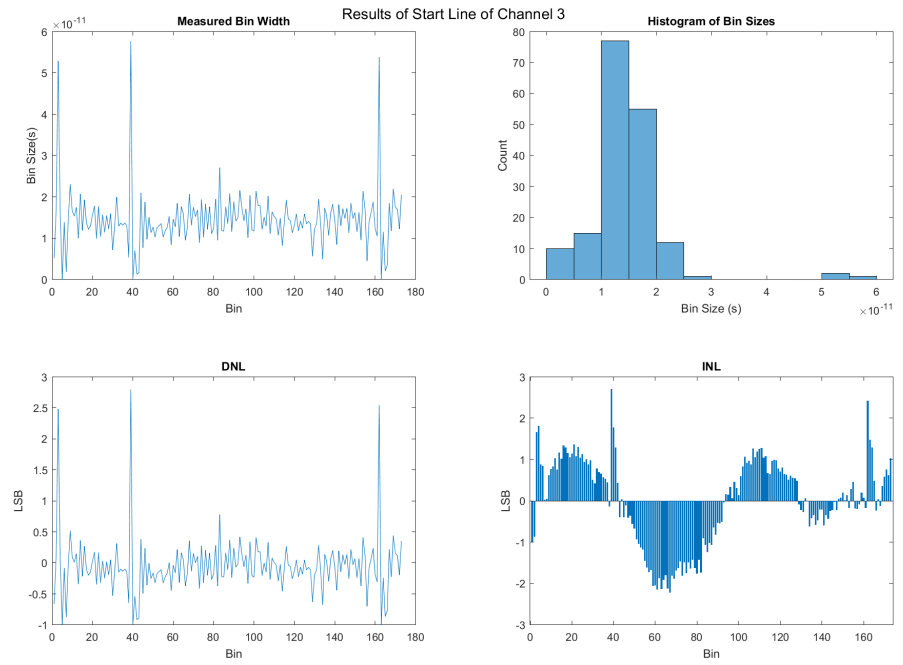

Results of Stop Line of Channel 1
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

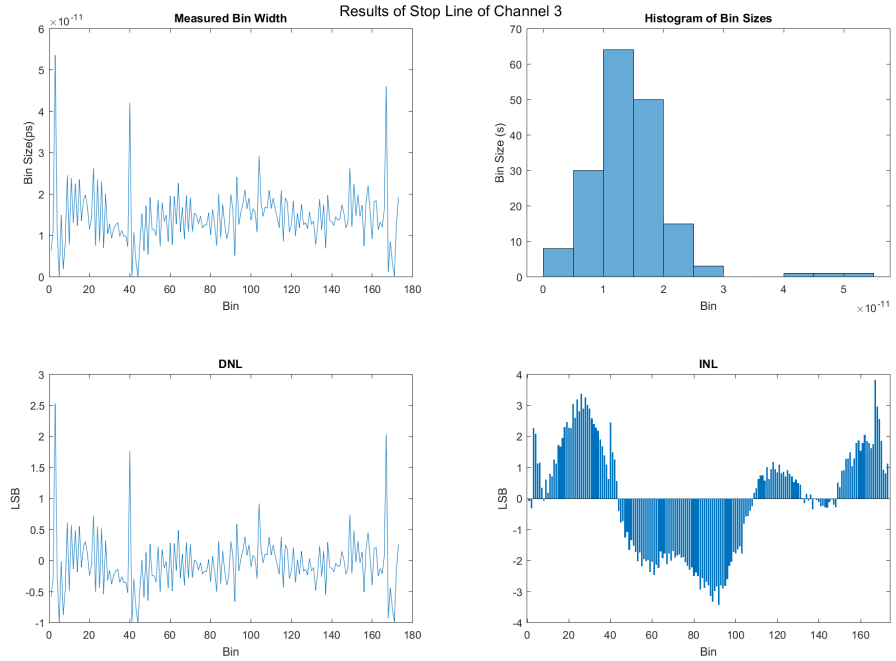

Measurement Precision Results of Channel 1
All measurements
Std Deviation -- 1.101829e-09
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 1.993805e-10
Count
Measured Time (s)
×10^-7

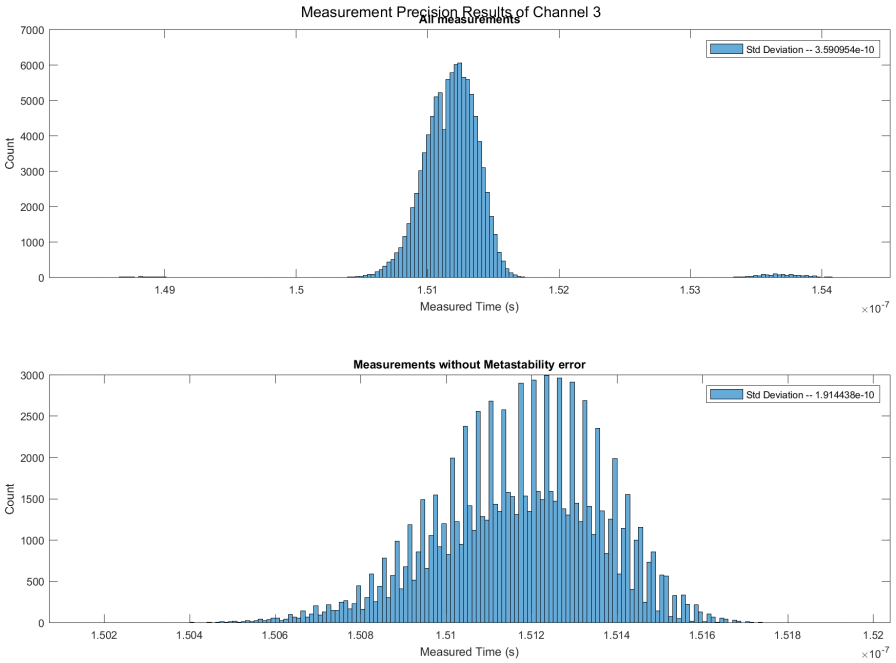

Results of Start Line of Channel 2
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

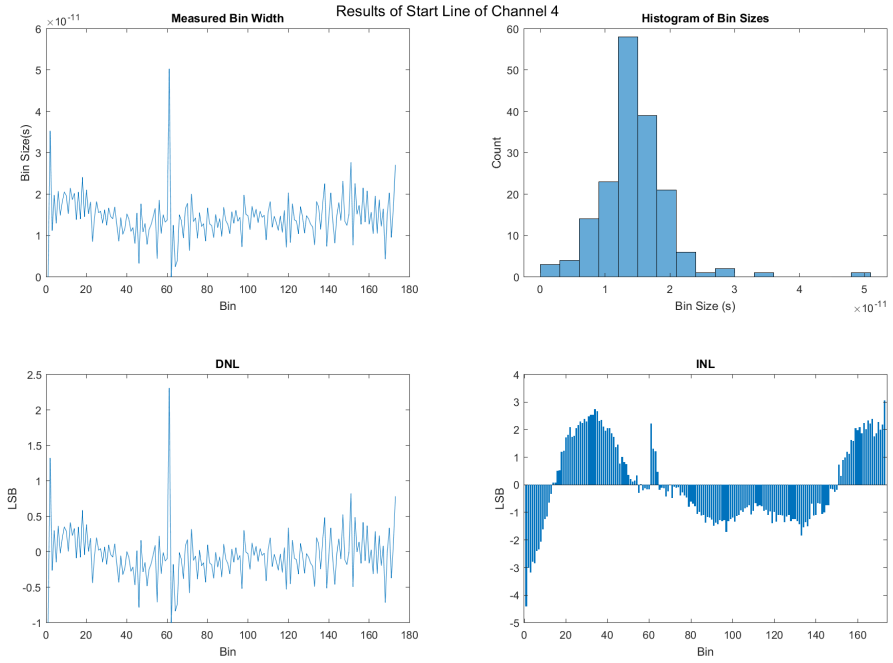

Results of Stop Line of Channel 2
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin

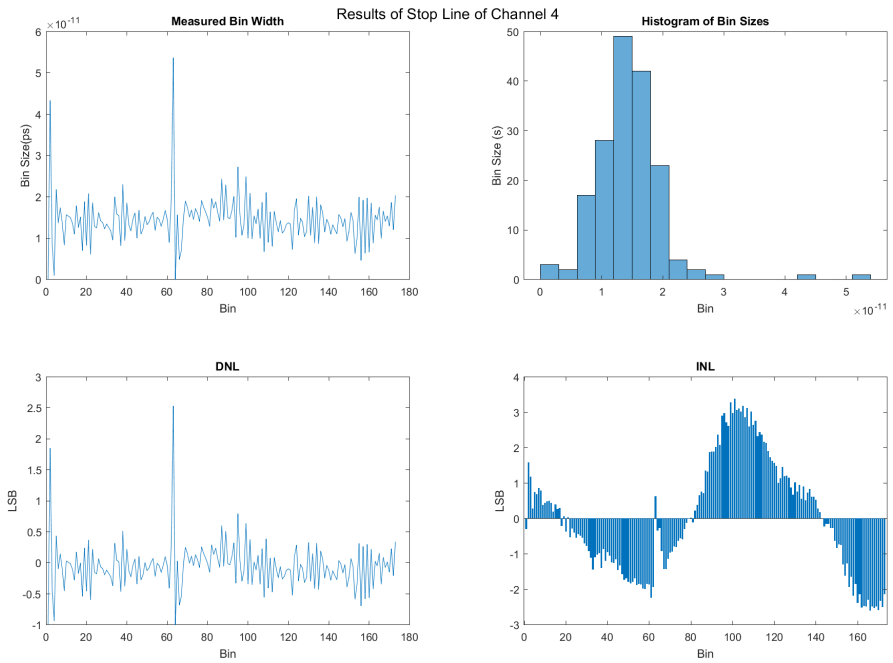

Measurement Precision Results of Channel 2
All measurements
Std Deviation -- 7.322759e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.190431e-10
Count
Measured Time (s)
×10^-7

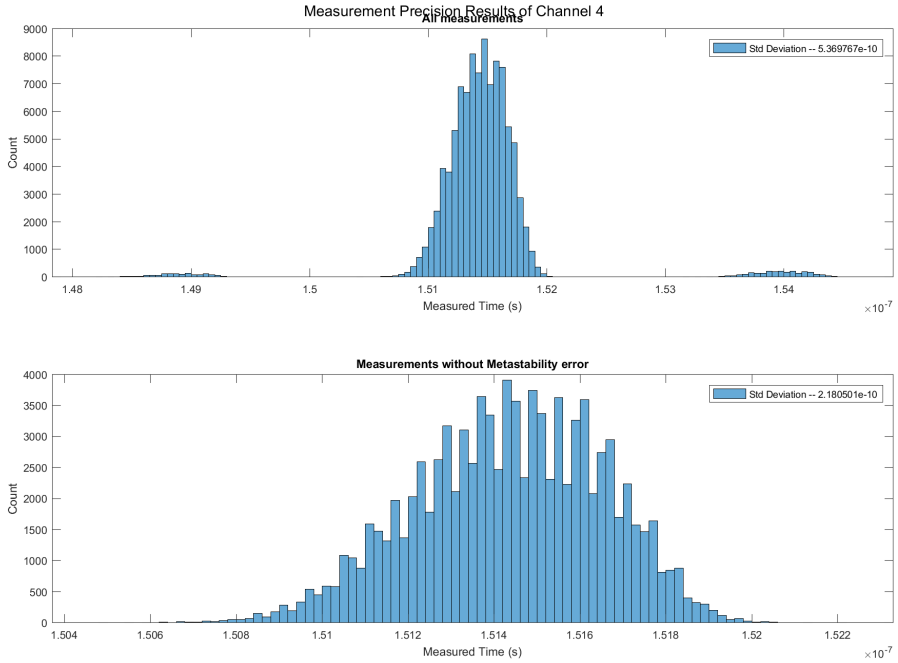

Results of Start Line of Channel 3
Measured Bin Width
×10^-11
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
×10^-11
DNL
LSB
Bin
INL
LSB
Bin


Results of Stop Line of Channel 3
Measured Bin Width
×10^-11
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
×10^-11
DNL
LSB
Bin
INL
LSB
Bin


Measurement Precision Results of Channel 3
All measurements
Std Deviation -- 3.590954e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 1.914438e-10
Count
Measured Time (s)
×10^-7


Results of Start Line of Channel 4
Measured Bin Width
Bin Size(s)
Bin
Histogram of Bin Sizes
Count
Bin Size (s)
DNL
LSB
Bin
INL
LSB
Bin


Results of Stop Line of Channel 4
Measured Bin Width
Bin Size(ps)
Bin
Histogram of Bin Sizes
Bin Size (s)
Bin
DNL
LSB
Bin
INL
LSB
Bin


Measurement Precision Results of Channel 4
All measurements
Std Deviation -- 5.369767e-10
Count
Measured Time (s)
×10^-7
Measurements without Metastability error
Std Deviation -- 2.180501e-10
Count
Measured Time (s)
×10^-7

References

[1] S. Choi, F. Thalmayr, D. Wee, and F. Weig, "Advanced driver-assistance systems: Challenges and opportunities ahead | McKinsey." [Online]. Available: https://www.mckinsey.com/industries/semiconductors/our-insights/advanced-driver-assistance-systems-challenges-and-opportunities-ahead

[2] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[3] H. Estl, "Sensor fusion: A critical step on the road to autonomous vehicles." [Online]. Available: https://www.eenewseurope.com/news/sensor-fusion-critical-step-road-autonomous-vehicles

[4] T. Herpe, C. Lauer, R. German, and J. Salzberger, "Multi-sensor data fusion in automotive applications," *Proceedings of the 3rd International Conference on Sensing Technology, ICST 2008*, pp. 206–211, 2008.

[5] M. Galvani, "History and Future of Driver Assistance," *IEEE Instrumentation & Measurement Magazine*, vol. 22, no. February, pp. 11–16, 2019.

[6] J. Walker, "The Self-Driving Car Timeline – Predictions from the Top 11 Global Automakers | Emerj - Artificial Intelligence Research and Insight," 2019. [Online]. Available: https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/

[7] National Highway Traffic Safety Administration, "Automated Vehicles for Safety | NHTSA." [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[8] T. Duvall, E. Hannon, J. Katseff, B. Safran, and T. Wallace, "A new look at autonomous-vehicle infrastructure | McKinsey." [Online]. Available: https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/a-new-look-at-autonomous-vehicle-infrastructure

[9] Michael Poulin, "LiDar Technology for Safer ADAS and Autonomous Automobiles," 2016. [Online]. Available: https://www.azosensors.com/article.aspx?ArticleID=701

[10] K. Heineke, P. Kampshoff, A. Mkrtchyan, and E. Shao, "Self-driving car technology: When will the robots hit the road? | McKinsey," 2017. [Online]. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/self-driving-car-technology-when-will-the-robots-hit-the-road

[11] "LIDAR technologies for the Automotive Industry: Technology benchmark, Challenges, Market forecasts," 2018. [Online]. Available: https://www.researchandmarkets.com/research/cvwt57/lidar?w=5

[12] Texas Instruments Inc., "LIDAR Pulsed Time of Flight Reference Design," no. January, 2018.

[13] A. Ziebinski, R. Cupek, H. Erdogan, and S. Waechter, "A Survey of ADAS Technologies for the Future Perspective of Sensor Fusion," in *Computational Collective Intelligence: 8th International Conference, ICCCI 2016, Halkidiki, Greece, September 28-30, 2016. Proceedings, Part II*. Springer, Cham, 2016, pp. 135–146. [Online]. Available: http://link.springer.com/10.1007/978-3-319-45246-3{_}13

[14] I. Starepravo, "How Autonomous Vehicles Sensors Fusion Helps Avoid Deaths | Intellias Blog," 2018. [Online]. Available: https://www.intellias.com/sensor-fusion-autonomous-cars-helps-avoid-deaths-road/

[15] Kai-Tai Song, Chih-Hao Chen, and C.-H. Huang, "Design and experimental study of an ultrasonic sensor system for lateral collision avoidance at low speeds," in *IEEE Intelligent Vehicles Symposium*. IEEE, 2004, pp. 647–652. [Online]. Available: http://ieeexplore.ieee.org/document/1336460/

[16] A. M. Singh, S. Bera, and R. Bera, "Review on Vehicular Radar for Road Safety," in *Advances in Communication, Cloud, and Big Data*. Springer, Singapore, 2016, pp. 41–47. [Online]. Available: http://link.springer.com/10.1007/978-981-10-8911-4{_}5https://link.springer.com/chapter/10.1007/978-981-10-8911-4{_}5{%}0Ahttp://link.springer.com/10.1007/978-981-10-8911-4{_}5

[17] "LIDAR vs RADAR for Applied Autonomy | Semcon." [Online]. Available: https://semcon.com/what-we-do/applied-autonomy/lidar-vs-radar-for-applied-autonomy/

[18] D. Santo, "Autonomous Cars' Pick: Camera, Radar, Lidar? | EE Times." [Online]. Available: https://www.eetimes.com/author.asp?section{_}id=36{&}doc{_}id=1330069

[19] G. P. Stein, E. Rushinek, G. Hayun, and A. Shashua, "A Computer Vision System on a Chip: a case study from the automotive domain," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, vol. 3. IEEE, 2005, pp. 130–130. [Online]. Available: http://ieeexplore.ieee.org/document/1565445/http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1565445

[20] L. I. Pierre Olivier, Vice President, Engineering and Manufacturing, "LEDDAR optical time-of-flight sensing technology: A new approach to detection and ranging," *White Paper*, vol. 39, no. 5, pp. 561–563, 2008.

[21] Albie Jarvis, "Velodyne Executive Addresses How Velodyne's Next Generation of Sensors Will Enable True Autonomy - Velodyne Lidar," 2018. [Online]. Available: https://velodynelidar.com/newsroom/velodyne-executive-addresses-how-velodynes-next-generation-of-sensors-will-enable-true-autonomy/

[22] B. Marshall, "Lidar, Radar & Digital Cameras: the Eyes of Autonomous Vehicles." [Online]. Available: https://www.rs-online.com/designspark/lidar-radar-digital-cameras-the-eyes-of-autonomous-vehicles

[23] A. Saxena, "How Automotive LIDAR works for Autonomous Vehicles," 2018. [Online]. Available: https://www.einfochips.com/blog/how-lidar-based-adas-work-for-autonomous-vehicles/

[24] X. Qin, L. Wang, D. Liu, Y. Zhao, X. Rong, and J. Du, "A 1.15-ps Bin Size and 3.5-ps Single-Shot Precision Time-to-Digital Converter With On-Board Offset Correction in an FPGA," vol. 9499, no. c, pp. 3–9, 2017.

[25] N. S. Voros and K. Masselos, *System level design of reconfigurable systems-on-chip*, N. S. Voros and K. Masselos, Eds. Berlin/Heidelberg: Springer-Verlag, 2005. [Online]. Available: http://link.springer.com/10.1007/b136832

[26] M. Zhang, H. Wang, and Y. Liu, "A 7.4 ps FPGA-based TDC with a 1024-unit measurement matrix," *Sensors (Switzerland)*, vol. 17, no. 4, 2017.

[27] S. M. Trimberger, "Three ages of FPGAs: A retrospective on the first thirty years of FPGA technology," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 318–331, 2015. [Online]. Available: https://en.wikipedia.org/wiki/DISAMATIC

[28] Xilinx, "Modeling Latches and Flip-flops," pp. 1–9, 2015.

[29] S. Henzler, S. E. K. Itoh, T. H. Lee, T. Sakurai, W. M. C. Sansen, and K. Chun, *Springer Series in Advanced Microelectronics: Time-to-Digital Converters*, Springer, Ed., 2010, vol. 0, no. 0.

[30] Z. Cheng, X. Zheng, M. J. Deen, and H. Peng, "Recent developments and design challenges of high-performance ring oscillator CMOS time-to-digital converters," *IEEE Transactions on Electron Devices*, vol. 63, no. 1, pp. 235–251, 2016.

[31] C.-C. Chen, C.-S. Hwang, Y. Lin, and G.-H. Chen, "Note: All-digital pulse-shrinking time-to-digital converter with improved dynamic range," *Review of Scientific Instruments*, vol. 87, p. 046104, 04 2016.

[32] Z. Soni, A. Patel, D. K. Panda, and A. B. Sarbadhikari, "Comparative Study of Delay Line Based Time to Digital Converter Using FPGA," *International Research Journal of Engineering and Technology(IRJET)*, vol. 4, no. 9, pp. 1169–1175, 2017. [Online]. Available: https://irjet.net/archives/V4/i9/IRJET-V4I9208.pdf

[33] Y. Wang and C. Liu, "A 3.9 ps Time-interval RMS Precision Time-to-Digital Converter using a Dual-sampling Method in an UltraScale FPGA," *IEEE Transactions on Nuclear Science*, vol. 63, no. 5, pp. 2632–2638, 2016.

[34] M. Tanveer, I. Nissinen, J. Nissinen, J. Kostamovaara, J. Borg, and J. Johansson, "Time-to-digital converter based on analog time expansion for 3d time of flight cameras," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 9022, 02 2014.

[35] E. Raisanen-Ruotsalainen, T. Rahkonen, and J. Kostamovaara, "An integrated time-to-digital converter with 30-ps single-shot precision," *Solid-State Circuits, IEEE Journal of*, vol. 35(10), pp. 1507 – 1510, 11 2000.

[36] P. Chen, C.-C. Chen, and Y.-S. Shen, "A low-cost low-power cmos time-to-digital converter based on pulse stretching," *Nuclear Science, IEEE Transactions on*, vol. 53, pp. 2215 – 2220, 09 2006.

[37] C. C. Chen, S. H. Lin, and C. S. Hwang, "An area-efficient CMOS time-to-digital converter based on a pulse-shrinking scheme," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 3, pp. 163–167, 2014.

[38] I. Diehl, K. Hansen, K. Kruger, C. Reckleben, F. Sefkow, L. Andricek, C. Jendrysik, J. Ninkovic, S. Petrovics, R. Richter, and F. Schopper, "Readout asic for fast digital imaging using sipm sensors: Concept study," 10 2015, pp. 1–3.

[39] L. Perktold and J. Christiansen, "A multichannel time-to-digital converter asic with better than 3 ps rms time resolution," *Journal of Instrumentation*, vol. 9, 12 2013.

[40] J. Mauricio, D. Gascon, D. Ciaglia, S. Gómez, G. Fernandez, and A. Sanuy, "Matrix: A novel two-dimensional resistive interpolation 15 ps time-to-digital converter asic," 10 2016, pp. 1–3.

[41] Q. Shen, S. Liu, B. Qi, Q. An, S. Liao, C. Peng, and W. Liu, "A multi-chain measurements averaging TDC implemented in a 40 nm FPGA," *2014 19th IEEE-NPSS Real Time Conference, RT 2014 - Conference Records*, pp. 6–8, 2015.

[42] Y.-H. Chen, "A counting-weighted calibration method for a field-programmable-gate-array-based time-to-digital converter," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 854, no. C, pp. 61–63, 2017.

[43] Y. Wang, J. Kuang, C. Liu, Q. Cao, and D. Li, "A flexible 32-channel time-to-digital converter implemented in a Xilinx Zynq-7000 field programmable gate array," *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 847, no. November, pp. 61–66, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.nima.2016.11.031

[44] B. Qi, S. Liu, Q. Shen, S. Liao, W. Cai, Z. Lin, W. Liu, C. Peng, and Q. An, "A compact readout electronics for the ground station of a quantum communication satellite," *IEEE Transactions on Nuclear Science*, vol. 62, no. 3, pp. 883–888, 2015.

[45] A. Tontini, L. Gasparini, L. Pancheri, and R. Passerone, "Design and Characterization of a Low-Cost FPGA-Based TDC," *IEEE Transactions on Nuclear Science*, vol. 65, no. 2, pp. 680–690, 2018.

[46] Y. Wang and C. Liu, "A nonlinearity minimization-oriented resource-saving time-to-digital converter implemented in a 28 nm Xilinx FPGA," *IEEE Transactions on Nuclear Science*, vol. 62, no. 5, pp. 2003–2009, 2015.

[47] D. Chaberski, R. Frankowski, M. Zieliński, and Ł. Zaworski, "Multiple-tapped-delay-line hardware-linearisation technique based on wire load regulation," *Measurement: Journal of the International Measurement Confederation*, vol. 92, pp. 103–113, 2016.

[48] Y. Wang, J. Kuang, C. Liu, and Q. Cao, "A 3.9-ps RMS Precision Time-to-Digital Converter Using Ones-Counter Encoding Scheme in a Kintex-7 FPGA," *IEEE Transactions on Nuclear Science*, vol. 64, no. 10, pp. 2713–2718, 2017.

[49] J. Wu and Z. Shi, "The 10-ps wave union TDC: Improving FPGA TDC resolution beyond its cell delay," *IEEE Nuclear Science Symposium Conference Record*, pp. 3440–3446, 2008.

[50] Y. Wang and C. Liu, "A 4.2 ps Time-Interval RMS Resolution Time-to-Digital Converter Using a Bin Decimation Method in an UltraScale FPGA," *IEEE Transactions on Nuclear Science*, vol. 63, no. 5, pp. 2632–2638, 2016.

[51] R. Machado, L. A. Rocha, and J. Cabral, "A novel synchronizer for a 17.9ps Nutt Time-to-Digital Converter implemented on FPGA," *2018 110th AEIT International Annual Conference, AEIT 2018*, pp. 1–6, 2018.

[52] S. Grzelak, M. Kowalski, J. Czoków, and M. Zieliński, "High resolution time-interval measurement systems applied to flow measurement," *Metrology and Measurement Systems*, vol. 21, no. 1, pp. 77–84, 2014.

[53] P. Napolitano, A. Moschitta, and P. Carbone, "A survey on time interval measurement techniques and testing methods," *2010 IEEE International Instrumentation and Measurement Technology Conference, I2MTC 2010 - Proceedings*, pp. 181–186, 2010.

[54] J. Yu, "Verilog Generate Configurable RTL Designs - Verilog Pro," 2018. [Online]. Available: https://www.verilogpro.com/verilog-generate-configurable-rtl/

[55] Vivado, "Vivado Design Suite User Guide: Synthesis," pp. 1–120, 2018.

[56] Xilinx, "Vivado Design Suite Properties Reference Guide," 2019.