UMinho | 2020

Rui Pedro Oliveira Machado   Readout Circuit for Time-Based Automotive Sensors

Universidade do Minho
Escola de Engenharia

Rui Pedro Oliveira Machado

Readout Circuit for Time-Based Automotive Sensors

março de 2020

REPÚBLICA PORTUGUESA | CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR
FCT Fundação para a Ciência e a Tecnologia
NORTE2020
CENTRO 2020
ALENTEJO 2020
POCH
PORTUGAL 2020
UNIÃO EUROPEIA Fundo Social Europeu

Universidade do Minho
Escola de Engenharia

Rui Pedro Oliveira Machado

Readout Circuit for Time-Based Automotive Sensors

Tese de Doutoramento
Programa Doutoral em Sistemas Avançados de Engenharia para a Indústria

Trabalho efetuado sobre a orientação de
Professor Doutor Jorge Miguel Nunes dos Santos Cabral
Professor Doutor Luís Alexandre Machado Rocha (à memória de)

março de 2020

ACKNOWLEDGMENTS

Accepting the PhD challenge was one of the hardest decisions that I have ever made. Not giving up was even harder. This journey has brought up the best and the worst of me, allowing me to know myself better and to become a better person. Throughout this emotional rollercoaster, I was only able to maintain my perseverance due to some people, to whom I want to thank for the time, support and valuable teachings.

To Dr. Jorge Cabral, my supervisor which I consider as a friend, thanks for guiding and caring about me throughout my entire academic path. To Dr. Luís Alexandre Rocha, may he rest in peace, thanks for everything. I will always remember you as an example of what a true scholar should be. Special thanks to Filipe Serra Alves, for being my supervisor without any obligation. I have learnt so much from you, both professionally and personally. I hope one day I can repay you for everything you did for me. Thanks to INL for receiving me. Thanks to ESRG for being an amazing group. Thanks to Fundação para a Ciência e Tecnologia (FCT) and Bosch Car Multimedia for financing my PhD (grant PD/BDE/114562/2016).

Thanks to all my friends, Pedro (Pimba), Luís Novais, Tiago Vasconcelos, Vasco Lima, Eurico Moreira, Miguel Azevedo, Tânia Moreira and everyone from LAR group. Special thanks to Juca, for the Saturday chill out coffees and gaming sessions. A Special thanks to my best friend, Antero, for the snooker games on Saturday afternoons remembering the times when we were kids, and for the trust and strength.

To my mother and father, a heartful thanks for the love, education, sacrifice and all the rest. I am who I am thanks to you. For all it is worth, you are the best parents one could have wished for. To my beloved sister, thank you for making me laugh so many times, you may not see it, but you are truly important. I am proud of who you have become. To my grandfather and role model, Manuel Machado, my source of strength and guardian angel, to whom I dedicate this thesis. Wherever you are, I hope you are proud.

Finally, to the love of my life, Juliana Martins. Thank you for everything you did for me the last 10 years. Whatever the future holds for us, please know that you are definitely the best thing that ever happened to me. You changed me in so many ways, and always for the better. Thank you for never doubting me, for seeing in me what nobody else sees, for truly knowing how and who I am. Thanks for making me believe that everything is possible and for encouraging me to always challenge myself. A friend, a girlfriend, a safe haven. You are my everything and only with your caress was I able to surpass this challenge.

Rui Machado, Guimarães, March 9th, 2020.

STATEMENT OF INTEGRITY

RESUMO

A pesquisa pelo veículo autónomo (AV) iniciou-se há já algumas décadas, com a introdução de vários sistemas inteligentes nos veículos do nosso quotidiano. O melhor exemplo deste tipo de sistemas são os *Advanced Driver-Assistance Systems* (ADAS). As grandes marcas da indústria automóvel e principais *Original Equipment Manufacturers* (OEMs) estão focados no desenvolvimento do primeiro AV. O *Light Detection And Ranging* (LiDAR) é considerado como uma tecnologia chave para implementação do AV.

O sistema de leitura e medição do tempo de voo (*Time-of-Flight* - ToF) é um dos subsistemas constituintes do sensor LiDAR, e assume extrema importância. Os sistemas de medição de ToF de alto desempenho são normalmente implementados recorrendo ao desenho de células lógicas específicas e customizadas, o que leva a um aumento do tempo de desenvolvimento do sistema e, consequentemente, do custo. Estes tipos de sistemas apresentam desempenhos superiores aos necessários e o seu nível de integração é reduzido. O desenvolvimento de sistemas de medição de ToF capazes de serem completamente desenhados por linguagens de descrição de hardware (HDL) e implementados através de um fluxo de desenvolvimento totalmente automatizado permitirá alcançar maior portabilidade e níveis de integração.

O propósito desta tese é o desenvolvimento e implementação de uma arquitetura para um sistema de medição de ToF, capaz de facilitar o processo de migração destes sistemas entre tecnologias e plataformas. As arquiteturas existentes foram analisadas e foram implementadas e avaliadas múltiplas arquiteturas recorrendo a plataformas de prototipagem. Para assegurar um processo de migração fluído, as ferramentas de desenho de *Application Specific Integrated Circuit* (ASIC) foram estudadas. Como resultado, foi desenvolvido um sistema de medição de ToF para aplicações automóveis LiDAR e estabelecido um fluxo de desenvolvimento que suporta a migração automatizada de arquiteturas ToF.

O contributo da presente tese baseia-se no estudo sobre como devem ser desenhados e implementados os sistemas de medição de ToF para permitirem um fluxo de desenvolvimento automatizado e aumentar a sua portabilidade e integração, mantendo o desempenho necessário em aplicações automóveis LiDAR. A investigação iniciou-se com uma revisão do estado da arte em sistemas de medição de ToF, que culminou no desenvolvimento de duas arquiteturas em *Field-Programmable Gate Array* (FPGA) e na fabricação de um ASIC utilizando fluxo de desenvolvimento proposto baseado em *Structured Data Path* (SDP) para migrar arquiteturas de medição de ToF baseados em FPGA para tecnologia ASIC.

Palavras-Chave: ASIC, FPGA, LiDAR, Tempo de voo, Time-to-Digital Converter (TDC).

ABSTRACT

The pursue for autonomous car started long ago with multiple smaller and smarter systems being researched and introduced gradually in our daily vehicles. The so-called Advanced Driver Assistance Systems (ADAS) are the best example of these stepwise process towards a full-autonomous vehicle (AV). Nowadays, all the major automotive groups and Original Equipment Manufacturers (OEMs) are pursuing the goal of launching an AV on the market. Light Detection And Ranging (LiDAR) is considered the key enabling technology to implement this AV.

LiDAR sensor are composed by a multitude of systems and components, being the time-of-flight (ToF) readout system of extreme importance. State-of-the-art high-performance ToF readouts are implemented using highly customized cells, which increases both development time and costs. The performance achieved with these solutions usually highly exceeds the required for LiDAR and their level of integration is also reduced. The development of ToF measurement architectures capable of being completely described using hardware description languages (HDL) and implemented using a full automatized design flow, will help to attain reduced development time, increased portability and a high level of integration.

This Thesis aims to develop and implement a ToF readout architecture to simplify the migration effort between platforms and technologies. Existing architectures are analyzed and, based on the acquired knowledge, multiple architectures developed and assessed using a fast prototype platform. To ensure a seamless migration process, the tools used on Application Specific Integrated Circuits (ASIC) development are studied, and the design flow steps that can be automated or supported by scripting are identified. The accomplishment of these activities enabled the development of a ToF measurement system for automotive LiDAR sensors and a design flow guideline and respective supporting scripts.

The present Thesis contribute by reasoning about how should a ToF measurement system be designed and implemented to enable a full automated design flow process, increase portability and integration, while maintaining the required performance for automotive LiDAR based systems. The research started with an exhaustive literature review on FPGA ToF measurement systems, which lead to the implementation of two FPGA-based architectures, and to the fabrication of an ASIC TDC using the proposed Structured Data Path (SDP) design flow for migrating FPGA-based ToF systems to ASIC technology.

Keywords: ASIC, FPGA, LiDAR, Time-of-Flight (ToF), Time-to-Digital Converters (TDC).

INDEX

INDEX OF FIGURES

INDEX OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ADAS	Advanced Driver-Assistance Systems
ADC	Analog-to-Digital Converter
ADPLL	All-Digital Phased Locked Loop
ASIC	Application Specific Integrated Circuit
AV	Autonomous Vehicle
AXI	Advanced eXtensible Interface
BMW	Bayerische Motoren Werke
BSP	Board Support Package
CCOpt	Clock Concurrent Optimization
CCS	Composite Current Source
CLB	Configurable Logic Block
CTS	Clock Tree Synthesis
DAC	Digital-to-Analog Converter
DLL	Delay Locked Loop
DNL	Differential Non-Linearity
DR	Dynamic Range
DRC	Design Rules Check
DSP	Digital Signal Processor
ECU	Engine Control Unit
EOF	End of File

FIFO	First-In First-Out
FPGA	Field-Programmable Gate Array
GM	General Motors
HDL	Hardware Description Language
IC	Integrated Circuit
INL	Integral Non-Linearity
IP	Intellectual Property
LEF	Library Exchange Format
LiDAR	Light Detection And Ranging
LSB	Least Significant Bit
LUT	Look-Up Table
LVS	Layout Versus Schematic
MCMM	Multi Corner Multi Mode
MEMS	Microelectromechanical Systems
MSB	Most Significant Bit
NDR	Non-Design Rules
NLDM	Non-Linear Delay Model
OCV	On-Chip variation
OEM	Original Equipment Manufacturer
PET	Positron Emission Tomography
PL	Programmable Logic

PLL	Phased Locked Loop
PS	Processing System
PVT	Process Voltage Temperature
RADAR	Radio Azimuth Direction And Ranging
RAM	Random Access Memory
RTL	Register Transfer Level
SAE	Society of Automotive Engineers
SDC	Synopsys Design Constraints
SDF	Standard Delay Format
SDP	Structured Data Path
SoC	System on Chip
SPI	Serial Peripheral Interface
TA	Time Amplifier
TCL	Tool Command Language
TDC	Time-to-Digital Converter
TDL	Tapped Delay Line
ToF	Time-of-Flight
TSMC	Taiwan Semiconductor Manufacturing Company
VCO	Voltage Controlled Oscillator
WU	Wave-Union
XDC	Xilinx Design Constraints

1. Introduction

The world, and the understanding humanity has of it, is constantly changing. New technologies are developed daily to address people's needs and ever-increasing demands, changing the reality in which we live. Automotive industry always played an important role on this technology advancement. Over the last few years, a collection of systems has been introduced in vehicles, aiming to improve the overall driving experience, increase drivers' safety and reduce driving hazards due to human error. The current trend fueling automotive research is the autonomous vehicle (AV) concept, a technology that, when achieved, will completely shift the driving paradigm. The realization of such concept relies on the ability to endow a vehicle with capabilities to percept its surrounding environment. Thus, vison systems such as Radar Cameras and LiDAR are currently in the spotlight of research, being LiDAR pointed out as one of the core technologies, propelling numerous research works worldwide.

In this chapter, an introductory concept of this Thesis is presented. First, this work's motivation is explained, contextualizing its pertinence, formulating the problem statement and defining the Thesis scope. Then, the research questions are devised, and the objectives and methodologies defined. Finally, the main contributions to the scientific state-of-the-art achieved during this Thesis are listed and the structure of the remainder of this Thesis is presented.

The Chapter is organized as follows: Section 1.1 formalizes the problem statement and presents the project's dependent requirements and constraints; Section 1.2 motivates this Thesis, describes the Thesis' scope and presents its research questions, targeted objectives and proposed methodologies to attain them, while answering the formulated questions. Section 1.3 presents this Thesis contributions and Section 1.4 describes the structure and organization of this document.

1.1. Contextualization and Problem statement

Some decades ago, if someone talked about a car, it would be describing an almost full mechanical system. It was in 1882 that Karl Benz first patented the Benz Patent-Motorwagen. Then, in 1908, the first automobiles became accessible to the masses with the famous Model T, commercialized by Ford. Over the years, futuristic insights about driverless cars, pushed the automotive industry further, resulting in impressive technological improvements. This pursue lead to the incremental appearance of various functionalities, supported by various sensors and electronics, which reduced the driver's workload and increased safety. Nowadays, a variety of sensors can be found in cars, among them, angular sensors used in the pedals to measure the throttle position, pressure sensors used to measure the fuel and boost pressure, temperature sensors, imaging sensors, etc. Recently, sensors like LiDAR, RADAR and Cameras, are gaining relevance due to its role on sensing the car surrounding environment, which is mandatory when implementing an Automated Vehicle (AV). For instance, in 2018, there were already 53 companies working on LiDAR technology for automotive in California [1.1]. Also, in 2016, the demand for all kind of sensor solutions was increasing [1.2].

Electronics have always played an important role on automotive industry, improving safety, automatizing some driving tasks in a controlled environment, and generally improving the driving experience with infotainment systems. Although the acceptance and introduction of Advanced Driving Assistance Systems (ADAS) was slow [1.3], nowadays they are part of everyday driving experience and are already considered as a must have system depending on the tier of the car. According to [1.3], an increase of 50% on the number of ADAS systems included in cars was verified in just two years, from 2014 to 2016, with the inclusion of surrounding view systems having an increase of more than 150%, in the same time span. ADAS are playing an important role on teaching people about autonomous driving and its revenues may be used to finance AV research [1.4], [1.5]. Nevertheless, this will only be possible if the adoption rate of ADAS on mass production vehicles increases [1.4], [1.5]. This highlights the need for cheaper, but still high reliable systems [1.3], [1.6]. The ever-increasing adoption of ADAS, together with the recent search for a full automated driving experience is continuously pushing electronic sensors and systems forwards. With sensors technologies holding the key for the future of automobile industry [1.7], research on low cost and more reliable sensors and readout mechanisms used in ADAS systems and AV is required to tackle the new challenges ahead in the competitive automotive industry.

According to the Society of Automotive Engineers (SAE), a full autonomous vehicle, level 5 in Figure 1.1, must be capable of controlling every aspect of the driving task in the entire imaginable scenarios. This means that electronic sensors and systems must be capable of performing even in the most severe conditions [1.8]. When no user interaction is allowed, as in the case of a level 5 vehicle, the electronics requirements drastically change compared to a typical ADAS. According to Goel [1.9], solving the challenges introduced by AV will require better hardware, to collect more data and with higher precision, and better software to analyze and make decisions based on the gathered data. Moreover, this new hardware will also have to be able to cope with the typical automotive requirements, i.e. low power, low area, resistance to harsh environment, etc. Fulfilling all these requirements is not a trivial task, and multiple OEMs have already invested millions of dollars and made partnerships trying to be the leaders on this new and emerging AV market. Companies like Honda, Ford, GM, Toyota, Volvo, Hyundai, BMW and Tesla have already announced their intention of having fully autonomous vehicles on the road between 2020 and 2030 [1.1], [1.8].

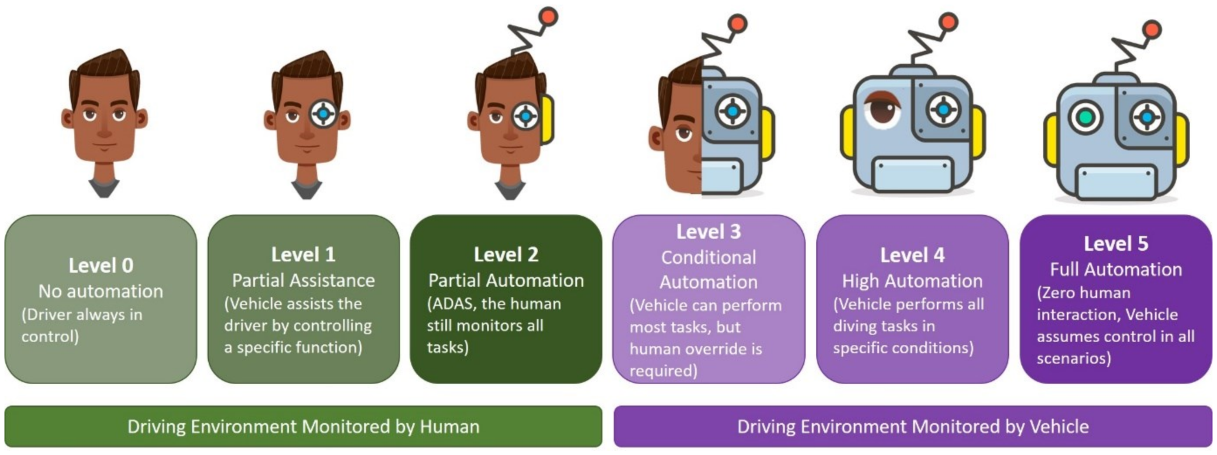

Level 0
No automation
(Driver always in control)
Level 1
Partial Assistance
(Vehicle assists the driver by controlling a specific function)
Level 2
Partial Automation
(ADAS, the human still monitors all tasks)
Level 3
Conditional Automation
(Vehicle can perform most tasks, but human override is required)
Level 4
High Automation
(Vehicle performs all diving tasks in specific conditions)
Level 5
Full Automation
(Zero human interaction, Vehicle assumes control in all scenarios)
Driving Environment Monitored by Human
Driving Environment Monitored by Vehicle

Figure 1.1- Levels of Automation (according to SAE)

Although being relatively new in automotive markets, the vehicle surroundings mapping advantages offered by LiDAR, when compared to the more established technologies (i.e. cameras, ultrasonic sensors and radar), have propelled massive innovations. Thus, LiDAR is already considered as a key enabling technology for achieving AVs [1.7], [1.10]. This popularity increase has also enabled a dizzying drop on LiDAR's cost, from around US$50000 to US$10000, with forecasts predicting a lower than US$200 cost per LiDAR module in 2022 [1.10].

The current scenario on ambient mapping solutions make use of Radar for long- and short-range measurements, video cameras for medium range and ultrasound for short-range measurements, mainly used on parking assistance ADAS [1.11]. The introduction of LiDAR may replace and/or complement some of these technologies, resulting in a schema like the one depicted on Figure 1.2.

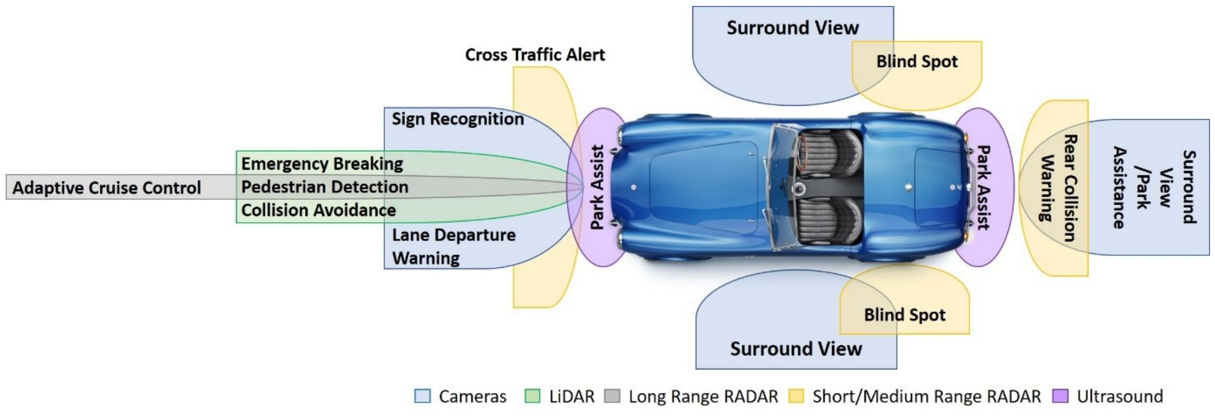


Figure 1.2- Car Vision Technologies (based on [1.3])

Ultrasonic sensors are only viable for short range measurements since the effects of attenuation are strong beyond a few meters distance. Furthermore, although ultrasonic sensors resolution may be suitable for object detection, it is not for objected identification. Although LiDAR can also perform in short range measurements and give detailed data on the object shape, facilitating the object identification, its cost is still way above of an ultrasonic sensor. Therefore, LiDAR sensors research mainly targets medium to long range measurement applications.

Other technologies targeting medium to long range measurements are Cameras and Radar [1.10]. Although Cameras offer a cost-effective solution due to its availability, the data processing power required, in order to extract useful information from the captured data, is a drawback of this technology. Furthermore, cameras are highly sensible to ambient light conditions. The only parameter in which LiDAR cannot outperform cameras is road signs and color detection. Thus, time critical detection tasks are better performed by LiDAR and cameras can be used to complement the information acquired by it, using a sensor fusion approach.

Most of LiDAR current applications could also be addressed by Radar. Since Radar solutions are available at lower cost and are easy to integrate, due to smaller size, this technology is the one used on modern-day vehicles. However, LiDAR research enabled LiDAR solutions to be shrunk over the years and the recent industry shift to solid-state LiDAR will further enhance integration and reduce costs [1.10]. Therefore, LiDAR is now capable of compete with Radar since it offers a set of performance improvements like larger distance, better angular resolution and larger field-of-view. This enables an improved object classification, with higher resolution in a broader scene/frame, without significant backend processing. The possibility to cover short, medium, and long range with a single sensor is attractive. Radar relies on the technology used to cover different ranges, meaning that a combination of technologies per each range must be made. Although LiDAR performance degrades with adverse weather conditions, when compared with Radar which offers a robust performance even under heavy rain, snow and fog, the use of 1550nm wavelength enables LiDAR to reach acceptable performance values [1.10]. An overview on the performance comparison between LiDAR, Radar and Cameras, based on the data reported by Mizuho Securities USA, and presented at the AutoSens 2017 conference held in Brussels [1.12], is given in Table 1.1.

Table 1.1- Vision Technologies Comparison

	LiDAR	*Radar*	*Camera*
Range	Best	Best	Worst
Field of View	Best	Better	Worst
Width and Height	Best	Worst	Worst
3D Shape	Best	Worst	Worst
Object recognition at long range	Best	Worst	Worst
Rain, Snow, Dust	Best	Best	Worst
Night	Best	Best	Worst
Signs and Color	Worst	Worst	Best

Source: AutoSens 2017, "LiDAR systems for automotive: Benefits and the challenges for OEMs" [1.12].

LiDAR working principle is based on transmitting a pulsed or continuous light signal (generated by a laser beam) which will be reflected by the different objects at the scene being scanned (Figure 1.3). By measuring the characteristics of the reflected signal, a high-fidelity picture of the scene being illuminated can be reconstructed. The usual parameters used in LiDAR measurements are the pulse's power, time-of-flight (ToF), and phase shift of the received signal [1.7], [1.10], [1.13]. The maximum achievable range for LiDAR measurement is presented in [1.13], and can be calculated according to the following equation:

$$Range = \sqrt{\frac{P*A*T_a*T_o}{D_s*P_i*B}}, \qquad (1.1)$$

where *P* is the laser's pulse power, *A* is the area of the optics used, *Ta* is the transmittance of the atmosphere, which is dependent on the ambient conditions, *To* is the transmittance of the optics, *Ds* the detector's sensitivity, and *B* is the laser beam divergence.

Moreover, the reflectance of the targets on the scene is also an important factor on the maximum achievable measurement range, since for the same emitted power pulse and the same detectors sensitivity, a less reflective target will be harder to identify, because most of the laser's pulse power will be absorbed.

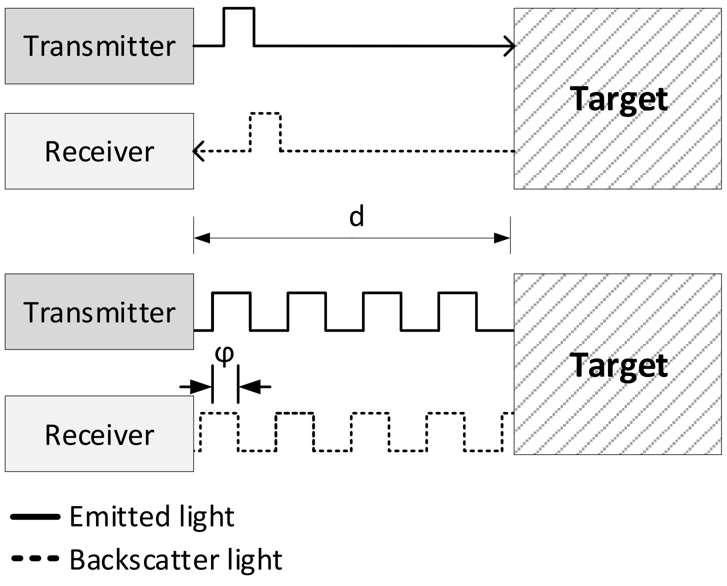


Figure 1.3- LiDAR Working Principle

When using a continuous wave light source (bottom of Figure 1.3), the distance to a target can be indirectly calculated by measuring the phase difference between the emitted pulse and the measured received signal, using:

$$d = \frac{c*\varphi}{4*\pi*f}, \qquad (1.2)$$

where *f* is the modulated frequency of the emitted signal, φ is the phase difference, and *c* is the speed of light (3*108 m/s).

Pulsed laser-based LiDAR systems are attractive due to its low power consumption, safety, low-cost, small-size and light weight [1.13], being currently the employed solution. Nevertheless, these solutions' performances are more affected by ambient light and weather conditions. The distance measurement is directly performed in pulsed LiDAR (top of Figure 1.3) by calculating the time interval between the instant when the light pulse was emitted and the instant that it is detected by the photoreceiver. Equation (1.3) is used to calculate the distance:

$$d = \frac{C*t}{2}, \qquad (1.3)$$

where d is the distance to a target, and t the measured time interval.

Independently of the distance to be measured, since the laser beam does not diverge significantly, a precise point distance measurement to the object can be made. Moreover, the narrow wavelength bandwidth of the LiDAR's laser allows for a higher ambient noise immunity by implementing a narrow bandwidth receiver.

Despite the achieved improvements on LiDAR technology, there are still progress to be made in order to reduce production costs and increase system's integration. Additionally, the amount of data capable of being collected by LiDAR sensors and the possible applications unlocked by the sensor, comprise a challenge by themselves.

The ideal LiDAR sensor solution is yet to be developed. The ideal solution would use a single laser, illuminating the entire scene and a 2D single photon detector. However, this ideal solution has high power peak consumption and the design of a detector immune to ambient noise, while being sensitive to the multiple backscattered light pulses, is yet to be accomplished. An even major drawback for this architecture is the abundance of retroreflectors in a typical driving situation, which reflect most of the light and has almost no backscattering, blinding the sensor and rendering it useless. The alternative is to use a scanning device. However, moving parts add mechanical noise to the measurements, deteriorating the overall system's performance.

LiDAR sensors can be classified according to their beam steering implementation method into mechanical LiDAR and solid-state LiDAR [1.10]. The mechanical group includes 2.5D macro scanners and 3D macro scanners. The solid-state LiDAR implementations can be divided into fixed beam, fixed multi beam, flash and MEMS (Micro-Electro-Mechanical System) LiDAR. Some solutions are already available commercially. The most significant solutions are presented on Table 1.2.

Despite the available options for LiDAR sensors, there is yet to be a solution that can combine high performance, low-cost and easy integration. Thus, LiDAR offers a good field of research. The Sensible Car project is a partnership between Bosch Car Multimedia and University of Minho in which research on

Table 1.2 -LiDAR Commercial Devices

Product name	*SRL1 Infrared short range lidar sensor*	*Leddar M16*	*Scala Wide angle scanning laser sensor*	*HDL-64E*	*HDL-32E*	*VLP 16*	*Peregrine 3D Flash LIDAR Vision system*	*SpectroScan 3D MEMS LIDAR MLS 201*
Lidar class	Fixed Beam	Fixed multi beam	2.5D macro scanner	3d Macro Scanner			Flash LiDAR	Microscanner
Manufacturer	Continental	Leddartech	Valeo IBEO	Velodyne Google, quanergy			ASCar	SpectroScan Boing
Basic principle	Direct TOF	Direct TOF	Direct TOF	Direct TOF			Direct TOF	Direct TOF
Range	1-10m (standard) 10-13.5m (expanded)	0-100 m	0.3 – 375 m	50-120m depending on reflectivity (0.1-0.8)	1-70m	100m	20.3m	20m
#pixels	3	16	2500	>100,000			1000-10,000	1000-100,000
#Laser-detector pair	1	16	4	64	32	16	-	-
Accuracy	+-100cm	+-5cm	+-40cm	+-2cm	+-2cm	+-3cm	-	-
Data rate	-	6.25 Hz to 100 Hz	-	1.3M px/s	700k px/s	300k px/s	-	-
Vertical fov	11°	-	3.2°	26.8°	40°	30°	Optional FoV lens: 15°	30°

Horizontal resolution	-	-	0.25°	5Hz: 0.08° 10Hz: 0.17° 20Hz: 0.35°	5Hz: 0.08° 10Hz: 0.17° 20Hz: 0.35°	5Hz: 0.1° 10Hz: 0.2° 20Hz: 0.4°	30° 45° 60°	0.2°
Vertical resolution	-	-	0.8°	0.4°	1.3°	2.0		0.2°
Horizontal fov	27°	9, 18, 24, 34, 45, 95°	145°	360°	360°	360°		60°
Pulse time	33 ns	-	-	10 ns	-	-	5 ±3 ns	-
Wavelength (nm)	905	940	905	905	903	903	1570	1550
Power(W)	<1.8	4	40	60	12	8	24	30
size (mm)	150*73*36	104*66*48	108*60*100	203*284	86*145	104*72	50*76*149.5	133.35*88.9*177.8
Weight	<100 g	180 g	510 g	15 kg	1.3 kg	0.83 kg	<680 g	2.27 kg
Estimated price	-	-	$250	$75000	$29900	$7999	-	-

LiDAR sensors is being addressed. The objective is to research a new integrated system based on laser detectors, which allows for vehicle's surroundings mapping, while complying with small size, low-cost and mass scale production requirements.

A typical LiDAR system is depicted in Figure 1.4. The sensor is composed by a laser, controlled by a laser driver block, responsible for generating the pulse pattern and for starting a Time-to-Digital Converter (TDC) module. On solid-state LiDAR, the laser beam can be redirected using a MEMS micromirror (MEMS LiDAR), in which case a micromirror driver module is required to control the micromirror movement, together with a synchronization block to guarantee the correct interaction between the pair laser-micromirror.

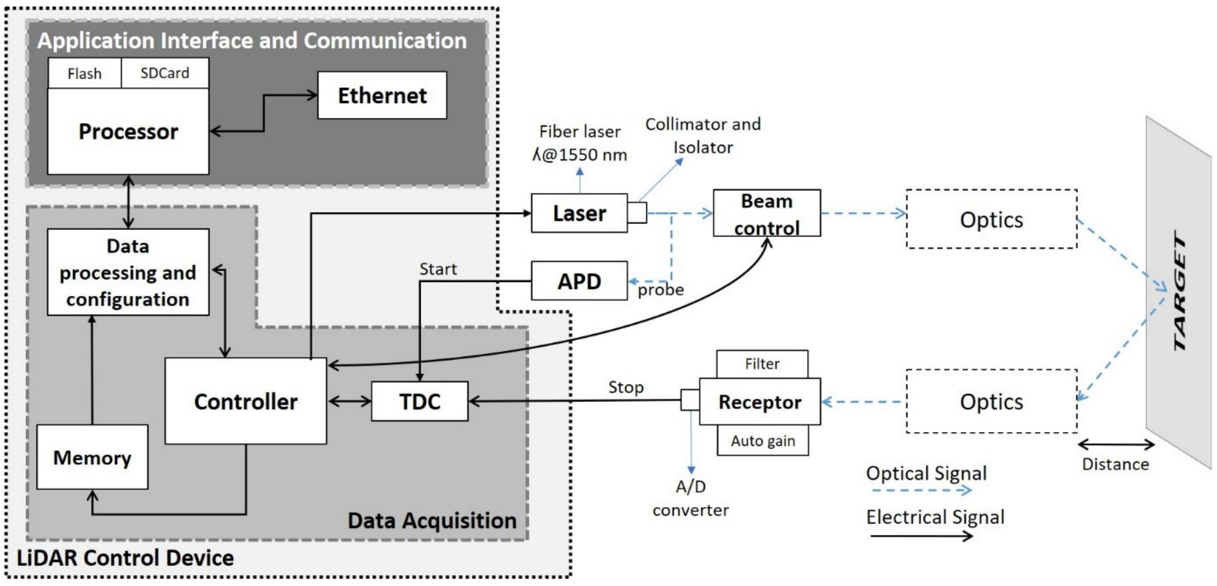


Figure 1.4- LiDAR Block Diagram

Another solution to obtain a larger field of view with just one laser is the use of a Flash LiDAR, in which a single pulse is fired [1.10], being the back-scattered light captured by a focal plane array of photodetectors located near the laser. This solution removes the need for the MEMS driver and synchronization block. However, it requires a TDC module per photodetector, which greatly increases production cost and power consumption. The receiver module is typically an analog block comprising an optical lens and optical filters in order to control the Field-of-View (FoV) of the detector, increasing its immunity to ambient light and another noise sources. The module also has an amplifier stage that converts the output of the detector into a usable signal. A comparator stage helps reducing the noise by generating a pulse only when the power of the received signal is above a defined threshold. The higher the threshold value, the higher will be the immunity to noise. However, for the same laser pulse power, the maximum detectable range will be lower. If the threshold value is to be reconfigurable, a DAC (Digital-to-Analog Converter) module will have to be implemented to adjust it. The last block on the detector module is a pulse generator, responsible for generating a valid stop pulse for the TDC module. The TDC module defines the minimum distinguishable distance of the LiDAR sensor, since this is the module responsible for measuring the time-of-flight of the laser pulse. The better the TDC resolution, the higher the LiDAR's resolution can be. For instance, a 250 ps (picoseconds) resolution TDC enables a precision of 3.75 cm (according to equation 1.3). The outputted value from the TDC is sent to a controller unit with a time-of-flight module, responsible for calculating, based on the outputted TDC value, the effective time interval between shooting the laser and detecting the backscattered light pulse. A microcontroller unit can be responsible to manage the data flow throughout the many modules comprising the LiDAR sensor and the memory module, where the data is stored. If a more software-oriented approach is being used, the microcontroller unit can be responsible for the entire of the data processing operation and calculations. However, due to a system and algorithms complexity increase, a divide and conquer approach is usually the strategy adopted, with multiple functions and algorithms being migrated to hardware accelerated modules. Timing constraints are also getting tighter (since the amount of data to acquired and processed in a single frame is increasing exponentially, which leaves less time for deciding and actuating in conformity), further endorsing the modular hardware acceleration approach. A data compressing block may also be included to reduce the cloud point information that needs to be treated and/or transmitted, simplifying the implementation of other modules responsible for data processing. The data processing modules can implement a myriad of services from object detection to human behavior prediction, object tracking, vehicle and path modelling, among others. Finally, the interconnection and communication module is responsible for shifting out the data to the system's network or a dedicated electronic controller unit (ECU).

1.2. Motivation, scope and Research Questions

One of the Sensible Car project objectives is to develop a LiDAR sensor capable of scanning a 50° horizontal per 9° vertical field of view with a maximum range of 180 meters. An angular resolution of 0.15° for both horizontal and vertical axis is required to enable the identification of objects as small as 400 mm at 180 m. In addition, a range accuracy greater than 0.1 meters and a frame rate between 10-20 Hz must also be achieved. Furthermore, in order to achieve a good depth contrast, which facilitates object detection, a depth resolution higher than 7cm is desirable. This Thesis targets one of the Sensible Car LiDAR's project core subsystems, the time-of-flight measurement unit. Since the project is to be concluded in a three-year time window, and there must be enough time for system integration and testing, it was defined that the ToF measurement unit research and implementation had to be done in less than three years. Moreover, apart from the time constraint, the ToF measurement unit must be easily integrated within the LiDAR sensor and easily ported between prototyping platforms (used in the initial stages of the project) and the final product platform (the final system is intended to be implemented in ASIC).

Nowadays, solutions for high performance time-to-digital conversion usually imply a custom-made process which increases both project's costs and development time. Thus, the aim of this Thesis is to develop a time interval measurement readout system to perform the Time-to-Digital Conversion (TDC) in a LiDAR sensor, capable of achieving high performance, with reduced customization and highly automated design flow processes, to accelerate development and reduce the system's cost. The main principle is based on being capable of directly migrating a fully digital, synthesizable TDC architecture, implemented in a low-cost prototype platform, i.e. FPGA, to an ASIC with minimum intervention.

The main motivation for this Thesis consists on the possibility to work with a large set of tools for hardware and software development which will allow the author to improve its knowledge. Furthermore, prior to the start of this work, there was no knowledge regarding TDCs development in the author's research group or Bosch Car Multimedia, which rises the challenge even further. Finally, the tools used to design digital systems are developed to be efficient in optimizing and analyzing synchronous designs, and therefore, the design of a system in which the relevant information is the one hidden between clock cycles, and that requires a precise characterization of the real circuit timings and not only the worst and best case scenarios is, by itself, a large and interesting challenge.

1.2.1. Research Questions and Objectives

The following research questions were formulated in order to guide this research, making possible to reach the aforementioned objective:

- RQ1: What are the current research trends on ToF measurement systems for LiDAR sensors?
- RQ2: Which architectures are simultaneously suitable for FPGA deployment (fast prototyping) and ASIC implementation, while maintaining the required performance for LiDAR sensors?

- RQ3: During the porting process (of the selected ToF measurement architecture) from FPGA-based to ASIC technology, how to minimize the development effort and time?
- RQ4: How does the developed ToF measurement solutions (FPGA and ASIC) compare with the current state-of-the-art for LiDAR sensors?

The following sub-objectives were defined in order to gradually pursue the main goal of this Thesis, while answering the Thesis' research questions:

- O1: Review the state-of-the-art in time interval measurement systems;
- O2: Develop different architecture working prototypes to gain insight on the main challenges and technical limitations when developing high performance time interval measurement systems;
- O3: Evaluate the developed architectures performance to understand the scenarios in which they are viable, and how should the ToF measurement systems characterization be performed;
- O4: Study the digital design flow for ASIC technology;
- O5: Create scripts to configure and automatize the ASIC digital design flow;
- O6: Implement a time interval measurement peripheral, addressing the issues identified during the state-of-the-art review and prototype development;
- O7: Evaluate the proposed architecture, development flow and implemented peripheral and position it within the existing state-of-the-art.

1.2.2. Research methodology

To focus and guide the activities involved in the research process that would enable the attainment of the proposed objectives (O1 to O7) and answer the formulated research questions (RQ1 to RQ4), several research methodologies, from RM1 to RM8, were adopted during this PhD Thesis.

RM1 – State-of-the-art of TDC: A review on the state-of-the-art of time interval measurement systems was performed on both academia and commercial fields to understand which type of architectures are being implemented, what are its typical performances, and which applications are being targeted. The review focused on architectures implemented in FPGA since these are the ones that can easily be ported to ASIC due to its intrinsic digital nature. The results of this study were published in article J1 [1.14], which proposes a taxonomy for FPGA-based TDCs classification, and identifies research gaps and new approaches that are not yet explored and may be an important contribution to build TDC systems. This study was performed to address RQ1 and RQ2, targeting O1 and providing a solid field knowledge, crucial to target the remaining research questions. Afterwards, a study on the available ASIC architectures was made to understand the typical design flow used on ASIC TDC development and the performance values achieved.

RM2 – Study the FPGA development framework: Developing time interval measurement systems, with resolution under the clock system frequency, requires a deep understanding on how the development frameworks are configured, as well as advanced knowledge of the FPGA platform being used. Therefore, a preliminary study of the Xilinx Vivado framework was made in order to: test multiple optimization configurations; assess how to avoid automatic optimizations on parts of a design; force the framework to generate specific hardware directly mapped to the FPGA configurable logic blocks (CLB); and understand how manual layout (placement and routing) could be applied to parts of the design, to improve the overall time interval measurement system's performance. The FPGA platforms used were studied to learn which type of resources were available and how should them be configured. This enabled to establish a solid expertise required to target O2 and O3.

RM3 – Development of FPGA-based TDC prototypes: To understand the challenges and issues involved on TDCs development, two different architectures were implemented in a FPGA device, namely the Xilinx Z7010. The first architecture was implemented with the objective of achieving the highest possible resolution, while the second one was designed targeting low resource utilization. Both architectures were designed to ensure portability and scalability. Details of the first architecture were presented in a conference proceeding C1 [1.15], with special focus on the synchronization block. Later, the same synchronization block was updated and a design methodology for synchronizer blocks was developed and presented in another conference proceeding C2 [1.16]. The details of the second architecture are partially presented in publication P1. The architecture consists on a modified version of the TDC presented by Wu and Xu in [1.17], which sacrifices the maximum achievable resolution in order to obtain improved linearity and homogeneity when multiple time interval measuring channels are implemented, and low resource and power consumption is required. Using this methodology, RQ2 was addressed and objective O2 achieved.

RM4 – Evaluate FPGA-based TDCs: The evaluation and characterization of the developed architectures was done to address RQ2 and target O3. The characterization process enabled a better understanding of the main metrics that need to be considered for proper TDC assessment, namely, which tests need to be performed and how to perform them. Two tests were performed on both architectures. The first test was a code density test with 100 thousand samples, made to retrieve the TDC's mean resolution and differential and integral non-linearity. After, a performance measurement with 100 thousand samples was executed to obtain the TDC's precision based on its standard deviation. A third test could be done to analyze the TDCs performance variation with temperature. However, since the objective of the FPGA implementations is to study the viable solutions for ASIC porting, and because the performance temperature variation is highly dependent on the technology used, there is no significant advantage in performing such test. Furthermore, according to [1.18], the Xilinx Zynq 7000 FPGA family does not have a significant performance variation with temperature. Therefore, the temperature tests were only made with the final ASIC TDC implementation. The results of the tests performed for the first and second architecture are presented in the conference proceeding C1 [1.15] and in publication P1, respectively.

RM5 – Study the ASIC design flow tools: The development tools for ASIC design are not the same as the ones for FPGA. Moreover, while using FPGA, a single framework with all the included tools was available. Although there are frameworks comprising every tool needed for an ASIC digital design flow, it is a de facto standard in industry that for synthesis, Synopsys' DesignCompiler offer the better results while Cadence's Innovus performs better during system layout. These tools are from different vendors, therefore the data transferred from one stage to the other must be handled by the ASIC designer. Moreover, as in the FPGA case, the tools are optimized to handle synchronous designs and to perform hardware intensive optimizations to reduce area, power consumption and design complexity. Those kinds of functionalities can only be applied to parts of the design (the synchronous part), while the module responsible for measuring sub-clock timings must be left out by the tool. Consequently, a thoughtful study on the tools used during digital ASIC design flow was performed, to understand how to manage the data exchange between tools, how to correctly configure the tools, how to manually manipulate the layout, and how to automate the different processes involved. With this methodology O4 was completed while addressing RQ3 and building the required knowledge to target O5 and O6.

RM6 – Migrate and Evaluate the FPGA-based TDCs to ASIC: Although being designed to be easily ported between platforms, the results from a TDC architecture porting could lead to unbearable performance drops. Therefore, the implemented FPGA TDC architectures were ported to ASIC and a pre-evaluation on its performance was made, to understand which of the architectures would be able to produce better results. The migration process implies the creation of multiple design flow scripts, to configure Synopsys and Cadence IC design Tools and to generate the final architecture layout, used during fabrication and final test simulations. From the obtained results, the first architecture was chosen to be fabricated. The porting results of the two architectures are presented in this Thesis on Chapter 4. These results address RQ3 and RQ4, while enabling to achieve O5 and start progressing into O6 and O7. The details on the ASIC design flow methodology adopted are reported in article J2 [1.19], while in article J3 [1.], the migrated architecture is presented in detail, together with the description of the scripts used to configure the tools during the synthesis and layout process.

RM7 – Tape-out the ASIC TDC peripheral: the final process before ASIC fabrication is known as tape-out. This process consists on the final chip layout configurations, i.e. creation of the ASIC's pad-ring, DRC (Design Rules Check) and LVS (Layout Vs Schematic) checks, and final timing simulation tests. During this process, a Printed Circuit Board (PCB), used to integrate and assess the ASIC TDC was designed and developed. Details on the entire ASIC design flow and PCB development are presented in article J3. These activities address RQ3 and enabled to complete O6, producing all the required resources to address RQ4 and achieve O7.

RM8 – Characterize and integrate the fabricated TDC: In order to be able to compare the implemented TDC with available implementations, a set of tests was performed to characterize the device. The main parameters are the TDC resolution, non-linearity, precision and temperature performance drifts. The first two parameters can be obtained by a code density test, similar to the ones performed on the FPGA-based TDCs prototypes. The precision and temperature drift tests were done in a thermal chamber, in a range from 0 to 50 Celsius degrees. A set of Matlab scripts were developed to analyze the data from the performed tests. The obtained results allow the comparison of the developed TDC with the existing solutions. The architectures comparison, described in Chapter 6, were done using the main TDc performance metrics, described in Chapter 2. This comparison addresses RQ4 while enabling to achieve O7.

1.2.3. Research Development Timeline

In order to better understand which research methodologies were applied to address the formulated research questions and how were the proposed objectives targeted, the research timeline is depicted in Figure 1.5. The RQs are represented as circles and used has the starting point to achieve a single or group of objectives, also represented as a circle. The connection is done by a rectangle specifying which methodology or set of methodologies were used to answer the research questions, accomplishing the objectives.

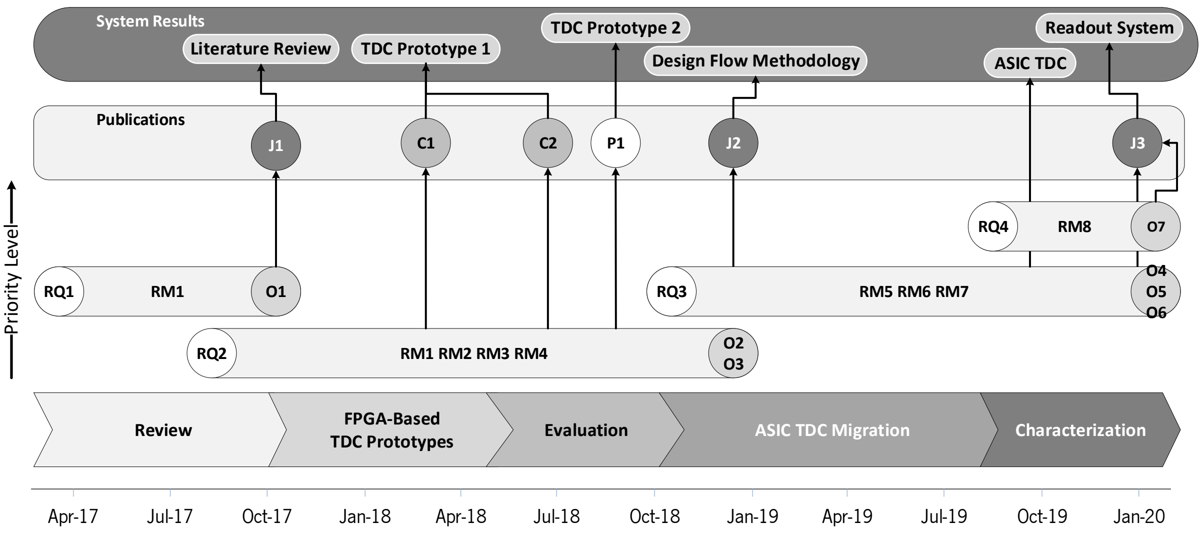


Figure 1.5- Thesis Timeline

1.3. Contributions

To support the development of this Thesis and to validate its scientific contribute, in the field of TDCs and ASIC design flow, the following publications were submitted to peer-reviewed international indexed conferences and journals:

Journal Papers:

- R. Machado, J. Cabral and F. S. Alves, "All-Digital Time-to-Digital Converter Design Methodology Based on Structured Data Paths," in IEEE Access, vol. 7, pp. 108447-108457, 2019. doi: 10.1109/ACCESS.2019.2933496
- R. Machado, J. Cabral and F. S. Alves, "Recent Developments and Challenges in FPGA-Based Time-to-Digital Converters," in IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 11, pp. 4205-4221, Nov. 2019. doi: 10.1109/TIM.2019.2938436

Conference Papers:

- R. Machado, L. A. Rocha and J. Cabral, "A novel synchronizer for a 17.9ps Nutt Time-to-Digital Converter implemented on FPGA," 2018 AEIT International Annual Conference, Bari, 2018, pp. 1-6. doi: 10.23919/AEIT.2018.8577365
- R. Machado, J. Cabral and F. Alves, "Designing Synchronizers for Nutt-TDCs," 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), Vienna, Austria, 2019, pp. 1-6. doi: 10.1109/EBCCSP.2019.8836914

Under Revision:

- R. Machado, F. Alves, A. Geraldes, J. Cabral, "Technology Independent ASIC based Time to Digital Converter", submitted IEEE Transactions on Circuits and Systems I: Regular Papers
- R. Machado, F. Alves, J. Cabral, "Gray-Code TDC Architecture with Improved Linearity and Scalability", submitted 2020 6th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP)

1.4. Thesis Organization

The remainder of this Thesis is structured as follows (Figure 1.6):

Chapter 2 introduces the basic concept regarding TDCs, providing a theoretical background to understand the design decisions made throughout this Thesis work. First, the main performance metrics are explained. After, the most popular architectures, implemented in FPGA and ASIC platforms, are described and discussed in detail. For each architecture, the most relevant research works present on the literature are mentioned. Since this Thesis targets a system with improved integration and portability, more emphasis is given to architectures that can be implemented in a fully digital system.

A description of the FPGA prototype platform and respective development environment is presented in Chapter 3, followed by the description of the selection process regarding which TDC architectures should be explored. Two architectures were selected considering the requirements for automotive LiDAR applications. These architectures modifications and support modules are described in detail, highlighting the benefits and identifying its limitations. A discussion on the performance assessment results, for each architecture, is presented at the end of the chapter. The main conclusions of this chapter support the decisions made throughout the development process of the final ASIC TDC.

Chapter 4 introduces the development tools for digital ASIC design. Afterwards, the complete system implementation process is presented, describing the required architectural changes, the details of the scripted development design flow that enables an almost seamless migration from FPGA to ASIC platform, and the expected TDC performance, inferred from the simulation results and extracted timing information.

The main experimental results of the developed TDC are presented in Chapter 5. Chapter 5 starts by describing the performed tests and experimental setup used. Then, the performance of the TDC is assessed, identifying its limitations and discussing the needed improvements.

Chapter 6 concludes this Thesis and summarizes the acquired knowledge and future research work for the time interval measurement device is proposed either to enhance its performance or its integration level.

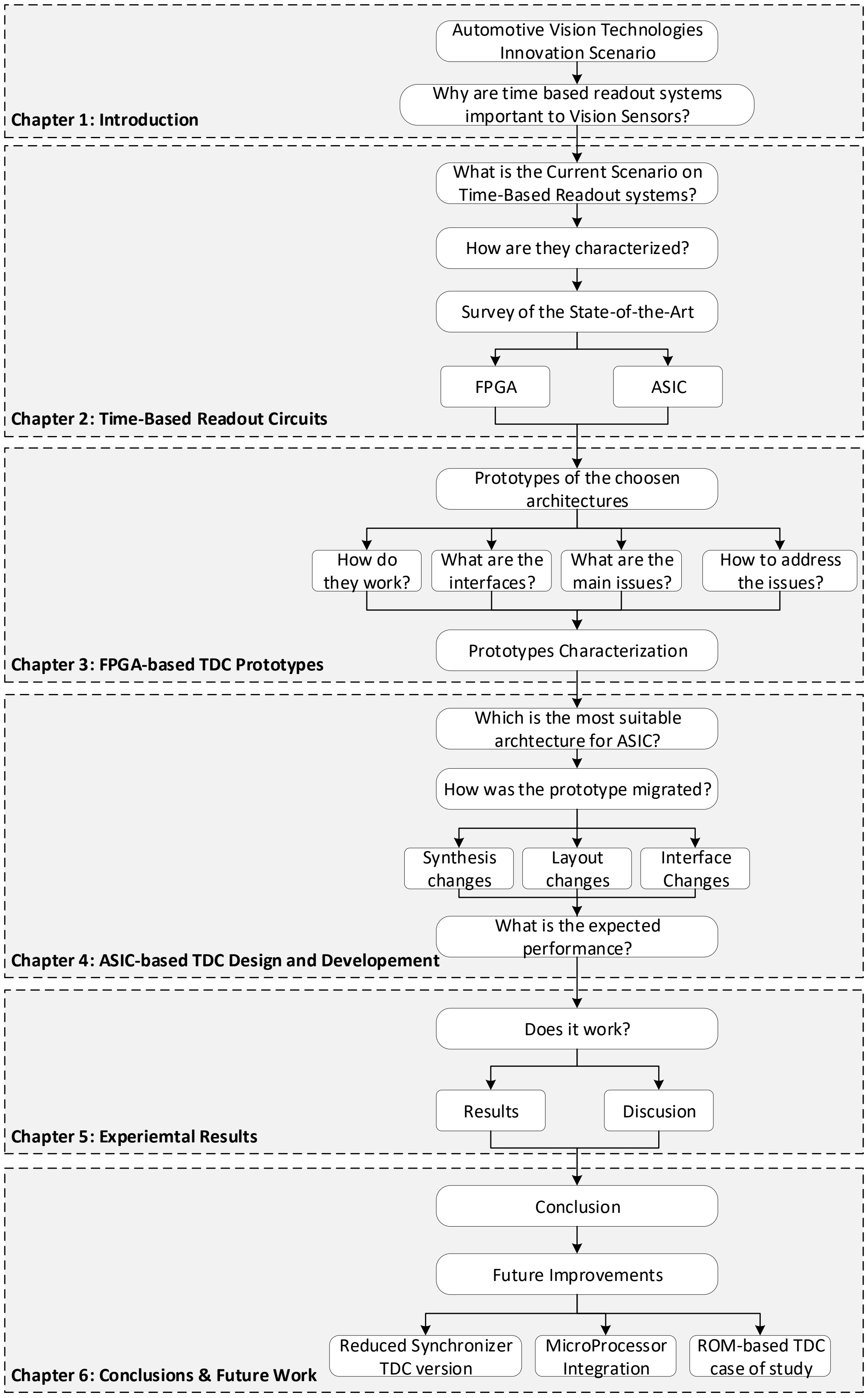

Automotive Vision Technologies Innovation Scenario
Why are time based readout systems important to Vision Sensors?
Chapter 1: Introduction
What is the Current Scenario on Time-Based Readout systems?
How are they characterized?
Survey of the State-of-the-Art
FPGA
ASIC
Chapter 2: Time-Based Readout Circuits
Prototypes of the choosen architectures
How do they work?
What are the interfaces?
What are the main issues?
How to address the issues?
Prototypes Characterization
Chapter 3: FPGA-based TDC Prototypes
Which is the most suitable archtecture for ASIC?
How was the prototype migrated?
Synthesis changes
Layout changes
Interface Changes
What is the expected performance?
Chapter 4: ASIC-based TDC Design and Developement
Does it work?
Results
Discusion
Chapter 5: Experiemtal Results
Conclusion
Future Improvements
Reduced Synchronizer TDC version
MicroProcessor Integration
ROM-based TDC case of study
Chapter 6: Conclusions & Future Work

Figure 1.6- Thesis Structure

References

[1.1] M. Avary, “3 autonomous vehicle trends to follow in 2019,” 2019. [Online]. Available: https://www.weforum.org/agenda/2019/01/3-autonomous-vehicle-trends-to-follow-in-2019/. [Accessed: 22-Jul-2019].

[1.2] N. Tyler, “Demand for automotive sensors is booming,” 2016. [Online]. Available: http://www.newelectronics.co.uk/electronics-technology/automotive-sensors-market-is-booming/149323/. [Accessed: 16-Oct-2019].

[1.3] K. Heineke, P. Kampshoff, A. Mkrtchyan, and E. Shao, “Self-driving car technology: When will the robots hit the road?,” 2017. [Online]. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/self-driving-car-technology-when-will-the-robots-hit-the-road. [Accessed: 22-Jul-2019].

[1.4] A. Padhi and P. Kampshoff, “Autonomous-driving disruption: Technology, use cases, and opportunities,” 2017. [Online]. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/autonomous-driving-disruption-technology-use-cases-and-opportunities. [Accessed: 22-Jul-2019].

[1.5] P. Gao, H.-W. Kaas, D. Mohr, and D. Wee, “Disruptive trends that will transform the auto industry,” 2016. [Online]. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/disruptive-trends-that-will-transform-the-auto-industry. [Accessed: 22-Jul-2019].

[1.6] S. Choi, F. Hansson, H.-W. Kaas, and J. Newman, “Capturing the advanced driver-assistance systems opportunity,” 2016. [Online]. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/capturing-the-advanced-driver-assistance-systems-opportunity. [Accessed: 22-Jul-2019].

[1.7] V. Hiligsmann, “How sensor technology will shape the cars of the future,” 2017. [Online]. Available: https://www.melexis.com/en/insights/knowhow/how-sensor-technology-shape-cars-future. [Accessed: 22-Jul-2019].

[1.8] J. Walker, “The Self-Driving Car Timeline – Predictions from the Top 11 Global Automakers,” 2019. [Online]. Available: https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/. [Accessed: 22-Jul-2019].

[1.9] A. Goel, “What Tech Will it Take to Put Self-Driving Cars on the Road?,” 2016. [Online]. Available: https://www.engineering.com/DesignerEdge/DesignerEdgeArticles/ArticleID/13270/What-Tech-Will-it-Take-to-Put-Self-Driving-Cars-on-the-Road.aspx. [Accessed: 22-Jul-2019].

[1.10] M. Khader and S. Cherian, “An Introduction to Automotive LIDAR.” Texas Instruments, 2018. [Online]. Available: http://www.ti.com/lit/wp/slyy150/slyy150.pdf.

[1.11] D. Bronzi, Y. Zou, F. Villa, S. Tisa, A. Tosi, and F. Zappa, “Automotive Three-Dimensional Vision Through a Single-Photon Counting SPAD Camera,” IEEE Trans. Intell. Transp. Syst., vol. 17, no. 3, pp. 782–795, Mar. 2016.

[1.12] AutoSens. LIDAR Systems for Automotive: Benefits and the Challenges for OEMs. (02-Mar-2018). Accessed: 16-Oct-2019. [Online Video]. Available: https://www.youtube.com/watch?v=InyXQ3IiTBA.

[1.13] P. McCormack, “LIDAR System design for Automotive/Industrial/Military Applications.” Texas Instruments, p. 10, 2011.

[1.14] R. Machado, J. Cabral, and F. S. Alves, “Recent Developments and Challenges in FPGA-Based Time-to-Digital Converters,” IEEE Trans. Instrum. Meas., vol. 68, no. 11, pp. 4205–4221, Nov. 2019.

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

[1.16] R. Machado, J. Cabral, and F. Alves, “Designing Synchronizers for Nutt-TDCs,” in 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), 2019, pp. 1–6.

[1.17] J. Wu and J. Xu, “A Novel TDC Scheme: Combinatorial Gray Code Oscillator Based TDC for Low Power and Low Resource Usage Applications,” in 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), 2019, pp. 1–7.

[1.18] 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 Trans. Nucl. Sci., vol. 63, no. 5, pp. 2632–2638, 2016.

[1.19] R. Machado, J. Cabral, and F. S. Alves, “All-Digital Time-to-Digital Converter Design Methodology Based on Structured Data Paths,” IEEE Access, vol. 7, pp. 108447–108457, 2019.

2. Time-based Readout Circuits

This chapter presents an exhaustive state-of-the-art, describing the literature and commercial solutions for time-of-flight measurement, which is, in LiDAR sensors, a crucial sub-system. Based on the analysis of the state-of-the-art, a discussion regarding the literature gaps is presented and the pillars for this Thesis' research are established.

Time-to-Digital Converters are circuits that measure the time interval between two events and convert it to a digital representation. This type of circuits has been extensively used in Positron-Emission Tomography (PET) research. The increased research interest on LiDAR sensors, also contributed for an increasing interest regarding TDCs, since time measurement resolution has a direct impact on the LiDAR's maximum precision, and thus on improving object detection [2.1]. Other applications for such systems are Oscilloscopes and high precision measurement systems, phase-locked loops (PLL) and time-of-flight rangefinders. The timing and performance characterization of electronic circuits, like jitter, skew and clock ageing, can also be made using a TDC. Other less explored applications are time-based accelerometers, in which the pull-in time is used to calculate the acceleration [2.2]. In these accelerometers, improved time resolution directly enables higher precision on the detection of the pull-in phenomenon instant, and thus the accelerometer's resolution.

Lately, FPGA technology has seen great performance improvements and costs reduction, due to developments in the available fabrication technologies (currently, the modern FPGAs are already implemented in 16 nm technology and below). Enhancements on FPGA's architectures (mainly on the configurable logic blocks and routing resources) have also contributed to this performance increase. As a result, FPGA platforms became more attractive to possible TDCs implementations, not only increasing the TDCs range of applications but also enabling performance values that can compete with the ones achieved by ASIC-based TDCs. With technology scaling down, the maximum achievable resolutions of TDCs in the digital domain increases, making this circuits more reliable, and ultimately replacing some analogue components, which in lower technology nodes, with lower voltages, have worst performances [2.3]. Due to these motives, all-digital TDC implementations are becoming increasingly attractive. The chart presented in Figure 2.1 depicts the increasing interest in TDC research over the last 28 years.

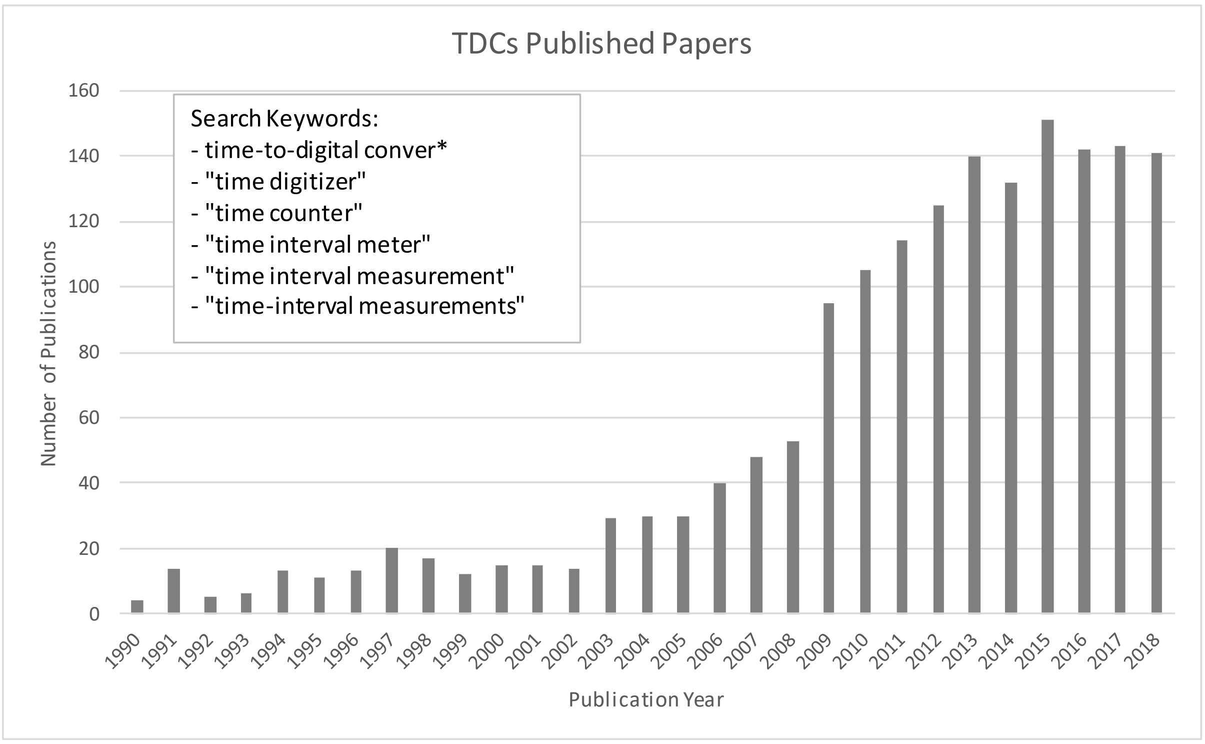


Figure 2.1- TDCs Research Interest Evolution

The Chapter is organized as follows: Section 2.1 describes the main performance metrics used to characterize TDCs, independently of the platform used. The most popular TDC architectures are presented in Section 2.2, describing its principles of operation, advantages and drawbacks, and highlighting the most relevant literature research works' results. No clear distinction is made regarding the implementation platform used, and, since one of this Thesis objectives is the seamless migration between platforms, all-digital TDC architectures are emphasized. On Section 2.3, the available commercial devices are listed to contextualize the current industry scenario regarding time-of-flight measurement modules. The Chapter ends with Section 2.4, where a discussion on the state-of-the-art gaps, relevant research paths identification, and description of this Thesis proposed research is presented.

2.1. Performance Metrics

Before analyzing the state-of-the-art, it is important to understand what metrics are used to characterize TDC's performance (see Figure 2.2 and Figure 2.3). These metrics are required to fully understand which are the benefits that each architecture has to offer and how do they compare to other architectures. Latter in the Section, the most relevant TDC architectures are presented.

Figure 2.2 presents real measurement data utilizing the ASIC TDC fabricated during this Thesis. It summarizes the main metrics for TDC characterization which will be further discussed in this document. Accuracy is defined as the deviation of the performed measurement to the real value. Precision is defined as the measurement standard deviation. Throughout this document, every time the term precision is used, it will be referring to the single-shot precision (described in section 2.1.2). In interpolative TDCs, the LSB value defines the minimum distinguishable time interval, and is tightly coupled to the propagation delay of one logic cell, or group of cells. A step, also commonly referred to as bin, is the cell or group of cells responsible for implementing a time delay equal to one LSB. Throughout this Thesis, both terms are used. Neither in FPGA nor ASIC, is possible to have cells with exactly the same propagation delay. Thus, the TDC's steps will also have discrepancies (even when different cells are merged to form a step). These discrepancies are known as the TDC's non-linearities and will be discussed in Section 2.1.3.

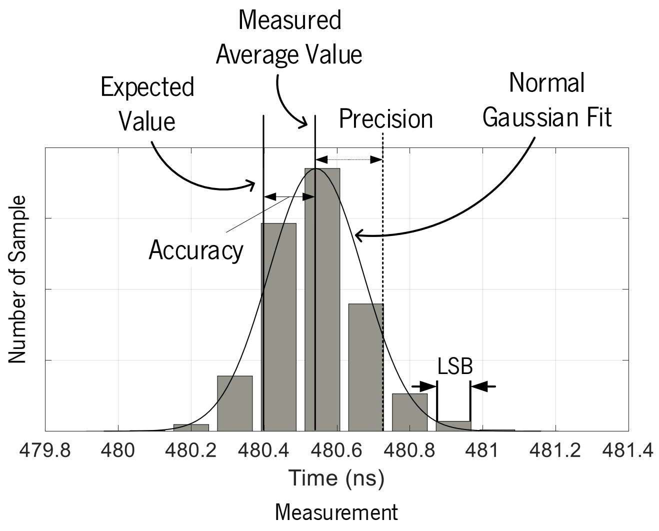


Figure 2.2- Characterization metrics

2.1.1. Dynamic range

The dynamic range is the maximum time interval that can be measured by a TDC before it overflows. Nowadays, technology enables for high performance TDCs even in prototype platforms, so TDCs are becoming more attractive for new applications that require large measurement ranges and benefit from fine resolutions (sub-nanosecond resolutions). An example of such applications are time-based accelerometers [2.2], where measurements in the range of a few milliseconds are required, and the fine resolution is desirable to improve the pull-in time measurement. LiDAR applications used in avionics for geo-mapping (airborne LiDAR) are another example that require large measurement range while benefiting from fine measurement resolution. Usually, coarse counters are included in the TDC architecture to complement the range of the TDC and reduce the number resources required to improve the dynamic range.

2.1.2. Resolution and Precision

It is common to refer to the TDC's resolution as the Least Significant Bit (LSB). This is the minimum incremental step that can be detected. For example, if a counter is clocked by a 50MHz crystal, its resolution (LSB) would be equal to 20 ns (the period of the reference clock). The precision of a TDC is usually presented as a standard deviation and represents the error from the expected measurement. Although there are some exceptions, this typically follows a gaussian distribution. According to [2.4], the rms value of the TDC precision can be calculated as follows:

$$\sigma_{TDCrms} = \sqrt{\sigma_q^2 + \sigma_{INL}^2 + \sigma_{CLK}^2 + \sigma_{extra}^2}, \quad (2.1)$$

where σ_q is the quantization error, given by LSB/$\sqrt{6}$, σ_{INL} is the TDC Integral non-linearity standard deviation, σ_{CLK} is the uncertainty (jitter) of the system reference clock, and σ_{extra} represents the contribution from external sources of jitter [2.5], [2.6].

Another commonly used method to evaluate the TDC precision is the single-shot precision [2.5], [2.7]-[2.10]. The single shot precision is usually obtained by performing a set of measurements (typically above 1000 samples) and calculating its standard deviation (see equation (2.2)). When the same channel is used to calculate the time of arrival of the start and stop events, according to [2.11], a single-shot resolution metric can be obtained by dividing the calculated measurement standard deviation by the square root of two.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \qquad (2.2)$$

2.1.3. Non-linearity

The TDC linearity is evaluated based on its Differential non-linearity (DNL) and Integral non-linearity (INL) (see Figure 2.3). Linearity errors are induced by mismatch on the TDC's LSB size due to Process, Voltage and Temperature (PVT) variations. The DNL is the deviation of a quantization step (bin) regarding its ideal value [2.12]. DNL is usually obtained through a code density test [2.13], which consists on feeding the TDC with a fixed and periodic input time interval. The input signal must have a frequency unrelated to the system reference clock, and multiple measurements must be collected to reduce statistical error influence. Since the input frequency is unrelated to the reference clock, the probability for each quantization step to be sampled is the same. By recording the number of times each quantization step was sampled, it is possible to obtain a realistic approximation of each step delay using equation (2.3):

$$\tau_i = n_i * \frac{T_{CLK}}{N}, (2.3)$$

where τ_i is the ith cell's delay, n_i is the number of times the ith delay cell was recorded, T_{CLK} is the system reference clock period and N is the number of measures performed. If we consider $\bar{\tau}$ as the theoretical delay, calculated as:

$$\bar{\tau} = \frac{T_{CLK}}{N_{cells}}, \qquad (2.4)$$

being N_{cells} the number of cells needed to fulfil a system reference clock period, then the DNL of each cell is defined as:

$$DNL_i = \tau_i - \bar{\tau}, \qquad (2.5)$$

The INL represents how large an error can be during a single measurement, estimating the non-linearity along the entire chain. Its value can be obtained by adding the DNL values of each cell of the TDL (see equation 2.6). It is also common to present the system's INL as the module of the maximum INL value.

$$\begin{aligned} INL_i &= \sum_{i=0}^{N-1}(\tau_i - \bar{\tau}) \\ INL &= max(|INL_i|) \end{aligned} \quad , \quad (2.6)$$

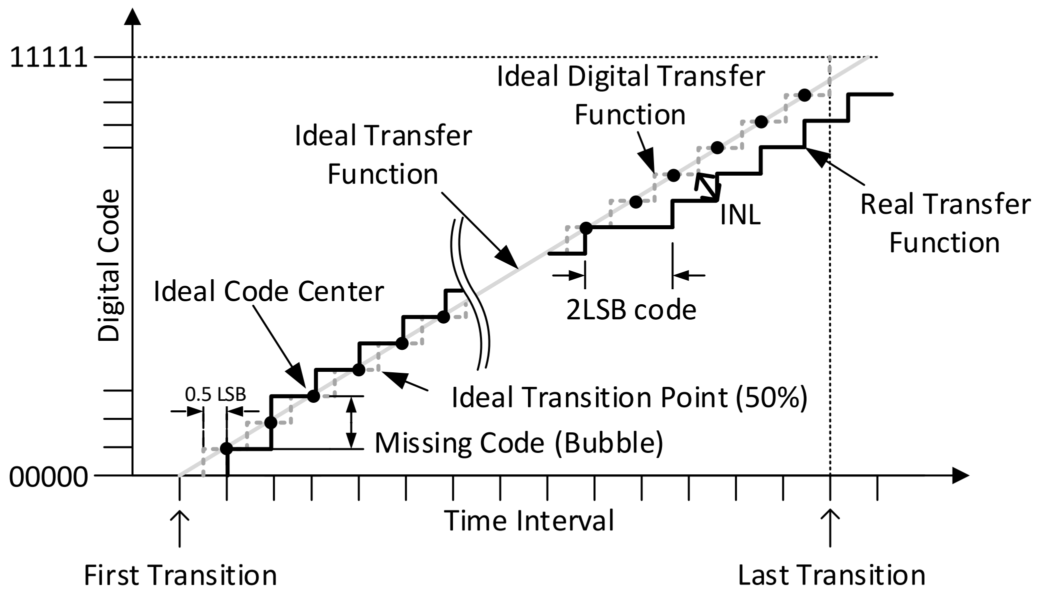


Figure 2.3- Linearity metrics

2.1.4. Dead Time

The dead time of a TDC is the time interval required, from the arrival of the stop signal, until the TDC is ready to perform a new measure. This metric is highly dependent on the TDC architecture. For instance, TDCs based on tapped delay lines usually report dead times equal to one reference clock period [2.14], while pulse shrinking [2.15] or ring oscillators [2.16] architectures can have a dead time dependent on the time interval to be measured, which can reach several hundreds of nanoseconds. Since high sample rate is required for modern applications, architectures capable of achieving low dead times are becoming popular. A common practice to reduce dead time and increase sample rate is to use multiple TDC channels multiplexed, measuring the same input signal in an interleaved schema [2.7].

2.1.5. Power Consumption, Area and Resource Usage

Power and area play an important role on modern applications since the current mobile trend requirements focus on low power and high levels of integration. In digital systems, the power is usually characterized as dynamic (switching) or static (leakage). The first is directly related to the operation frequency of the system, while the second one is technology dependent. Another technology dependent characteristic is the system's area or used resources, depending on whether the system is being implemented in ASIC or FPGA, respectively. Smaller ASIC technologies does not necessarily mean that the same TDC architecture can be implemented in a smaller area, since the delay elements need to be redesigned and it is not always possible to shrink the cells. However, smaller technologies can definitely enable higher performances to be achieved, at the expense of higher fabrication costs. Modern FPGA technologies also enable higher performance values at higher platform costs. FPGA architectures size are measured in resources utilization count rather than on area dimensions as in the ASIC scenario. In the case of FPGAs, the selected platform can also constrain the type of TDC architectures that can be implemented. Regarding the FPGA platforms used during this Thesis, when the term resources is used, it will be referring to: the FPGA's Configurable Logic Blocks (CLB) elements, namely, registers, Carry4 and Look-Up tables (LUTs); the FPGA's BRAM blocks; the FPGA's PLL blocks; and the FPGA DSP blocks. The type of resource being used will be enumerated whenever pertinent.

2.2. TDC Architectures

TDCs are highly dependent on the available resources and/or technology in use. While in ASIC platforms there is theoretically no constraint to the implementation of any TDC architecture, FPGA platforms limit the range of implementable architectures. Therefore, contrarily to FPGA where architectures share a lot of similarities, in ASIC platforms it is often possible to find completely new approaches. In the following sections, the TDC architectures are presented and analyzed, divided accordingly to their principle of operation. Nevertheless, the differences and nuances between the FPGA-based and ASIC-based TDC architectures are presented, whenever the implementation is possible on both platforms. The most relevant and distinct FPGA and ASIC architectures are presented from section 2.2.1 to section 2.2.7.

2.2.1. Coarse Counter Architectural Group

Course counters are TDC implemented using binary-, gray-code or ripple counters, which are incremented by a reference clock. The works in [2.17]–[2.19] are examples of coarse counters' implementation. The main advantage of such architectures is the simplicity of the design, its portability and the low resources utilization in FPGAs or small area in ASIC. Nevertheless, the achievable resolution is bounded to the frequency of the reference clock used. For instance, in order to achieve a resolution equal to 1 nanosecond, a 1 GHz clock is required. The course counters can be implemented using a free running schema, in which the clock is always enabled, and the time event to be measured, usually denoted as hit, is responsible for sampling the value in the counter registers. Coarse counters can also be implemented using an enable schema, in which the hit signal is responsible for enabling and disabling the counter. The range and resolution of this architecture is given by:

$$Range = 2^n, \quad (2.7)$$

$$\tau = \frac{1}{f_{CLK}}, \quad (2.8)$$

where n is the number of bits of the counter register and f_{CLK} is the system clock frequency.

When multiple bit coarse counters are implemented, the routing of the hit signal for the counter's registers must be carefully planned, regardless of its use as enable or sampling signal. Otherwise errors greater than 1 LSB may occur, especially when a binary code schema is used.

Although the operating frequencies of nowadays FPGAs and ASICs technologies are higher, for example, the works in [2.20] and [2.21] reported the use of 500 MHz and in [2.22] a 710 MHz reference clock, for resolutions under the nanosecond scale, these architectures are still not suitable. Furthermore, even if it was possible to use a high frequency reference clock to achieve picosecond resolutions (a clock higher than 10 GHz would be required), it would be harder to secure low skew values between the signals for the counter's registers. The largely enhanced skew effects increase the risk for metastability and counting errors. Therefore, the Coarse Counter architecture should only be employed when resolutions of a few nanoseconds and high measurement ranges are required. Recently, Wu and Xu proposed a Gray code counter without sampling stage [2.23]. This enables the operating frequency of the counter to be approximately equal to the cells' plus routings' propagation delays. Since the counting schema used is the Gray code, only one bit is changing at a time, which eliminates the possibility for incorrect counting sequence due to delays mismatch between the counting cells.

2.2.2. Analog TDC

Analog TDCs are usually built using a Time-to-Analog converter (TAC), followed by an Analog-to-Digital Converter (ADC). In these types of TDCs, a capacitor is charged by a fixed current source. The basic architecture is depicted in Figure 2.4. The amount of charge stored in the capacitor is proportional to the time the capacitor was charged. The final digital value is obtained using an ADC to convert the charge value into a digital value. The dynamic range (DR) of this architecture is given by equation (2.9):

$$DR = 2^n * T_{LSB}, \qquad (2.9)$$

where n is the number of bits of the ADC and T_{LSB} is the resolution of the ADC.

Although resolutions under 50 ps are possible, the use of a capacitor and a controlled current source, greatly increase the area and power consumption [2.4]. Moreover, this type of TDCs are highly susceptible to temperature drifts. More details on this architecture can be found in literature [2.4], [2.24]. Recently, a 50 ps resolution and precision analogue TDC has been implemented in a 110 nm process technology by Cossio [2.25].

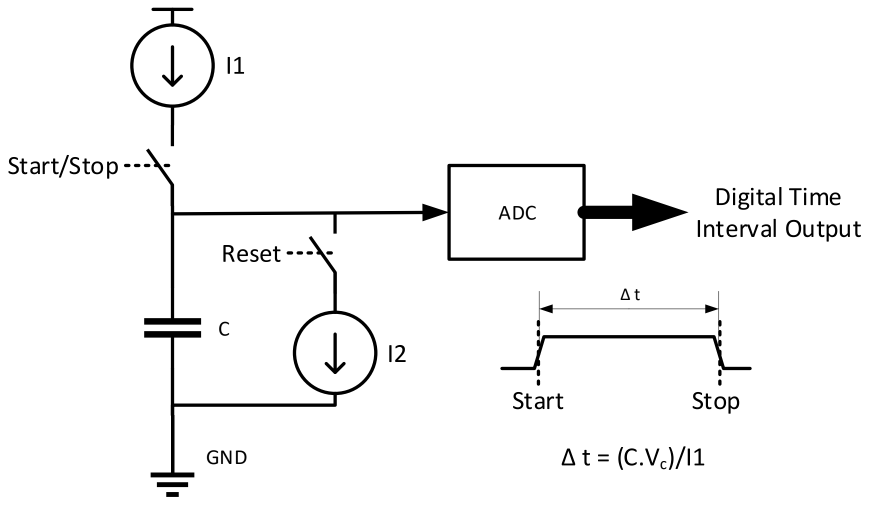


Figure 2.4- Analog TDC overview

2.2.3. Phased Clocks

An alternative to achieve resolutions in the range of a few hundreds of picoseconds is the use of TDCs based on phased clocks. These architectures are simple to implement and require very low resources when implemented in FPGA, since FPGAs platforms have PLL blocks integrated [2.26]–[2.30]. In ASIC, if a PLL is needed to generate the different clock phases, then the complexity of the architecture increases. Therefore, this type of architecture is especially advantageous in FPGA platforms.

Besides being relatively easy to implement, phased clocks architectures have good linearity and can achieve resolutions better than 300 picoseconds [2.29]. However, when compared to other high performance TDCs, the resolution of phased clocks continues to be one of its main drawbacks. Phased clocks architectures are based on two main techniques: oversampling and phase detection.

1) Oversampling: this architecture is based on using phased clocks as reference clocks to independent counters. The time event to measure is used as the counters' enable, just like in the case of coarse counters architecture. In fact, this approach is identical to the previous one, replicated m times, where m is the number of phased clocks used, and consequently the number of independent counters. It is important to mention that these phases have to be generated from the same reference clock and ideally, equally spaced. To determine the measurement value, equation (2.10) can be used:

$$t_{Oversampling} = (n_0 + n_1 + \cdots + n_n) * \frac{T_{CLK}}{m}, \qquad (2.10)$$

where n_0 to n_m represent the number of counts in each counter. These numbers are added and then multiplied by the TDC's resolution (given by the reference clock period T_{CLK}, divided by the number of phases used).

Resolutions equal to 1 nanosecond have been reported when using this architecture in [2.26]–[2.28], [2.31]. The main implementation challenge is related to the routing of the signal to be measured. The phase difference between the generated clocks is responsible for defining the clocks resolution. Therefore, the signal to be measured must be routed with the minimum skew possible between counters to avoid degrading the measurement. Also, as the number of used phases increases, the effect of jitter accumulates, degrading the TDC's performance. Therefore, to avoid phase overlap, special attention should be given to the design of the phase generation mechanism.

2) Phase detection: phase detection architectures sample the event to be timed with multiple phases (see Figure 2.5). The output of the sample process is a unique code dependent on which was the phase that first detected the event. The resolution (LSB) of these architectures is given by the phase difference between the multiple phases. As in the previous case, with the increase of the number of generated phases, the performance of the TDC tends to degrade due to the errors associated with the phase generation. With the increase of the frequency used as reference clock, and the routing skews, jitter and uncertainty of the phase generation, scenarios where the phase m arrives before phase $m+1$ can occur, resulting in bubble errors, which lead to missing codes, jeopardizing the TDC's performance. Again, the phase generator mechanism assumes high relevance and its design must be carefully done when targeting sub-nanosecond resolution. A synchronization stage, like the one depicted in Figure 2.5, is also needed to assure the creation of a common clock domain allowing the correct sampling of the code pattern by the reference clock, before it can be used to determine the instant of arrival of the time event.

The synchronization module number of stages and resource utilization increases with the number of phases implemented.

Opposed to what happens in the coarse counters and oversampling architectures, phase clocks based on phase detection offer great resolution and linearity but are not suitable for large dynamic ranges. To address this issue, it is common to extend the dynamic range of this architecture, complementing it with a coarse counter. In this way, the phase detection module just needs to cover the time equivalent to a full reference clock period while the coarse counter covers time intervals greater than the reference clock.

Using the setup depicted in Figure 2.5, this TDC architecture can be described by the following equations:

$$\tau = \frac{1}{N_{phases}}, \quad (2.11)$$

$$T_{fine} = T_{CLK} - (phase * \tau), \quad (2.12)$$

$$TDC_{measure} = n * T_{CLK} + T_{fine}, \quad (2.13)$$

where *phase* is the clock phase's number that sampled the input signal (from 0, for the 0° phase clock, to *m*, to the *m*° phase clock), and τ is the resolution given by the phase difference between clocks.

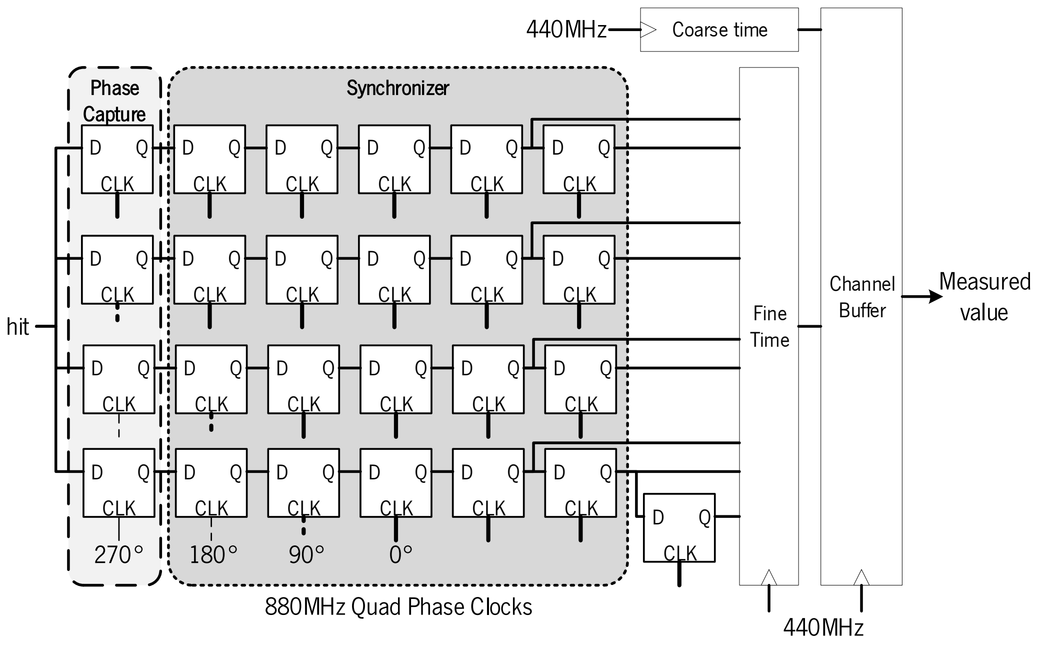


Figure 2.5- Phased Clocks based architecture (adapted from [2.29])

Research works, that use this architecture in FPGA platforms, report great linearity values without implementing calibration mechanisms [2.29], [2.32], [2.33]. A resolution of 625 picoseconds was reported in [2.32], the recent research work by Sano et al. [2.29] reports a 280 picoseconds (LSB) resolution with high linearity, being the DNL errors below 0.5 LSB. In [2.33], a precision of 56 picoseconds and resolution below 156 picoseconds was achieved, proving the potential of phase detection architectures. These results corroborate that high performance TDCs with lower design complexity and resource utilization can be achieved using phased detection architectures. Nevertheless, it is important to understand that the high linearity is also related to the relatively large LSB size. As the number of phases generated increase, the size of the LSB gets smaller and the errors associated with the clock's phase generation and routing paths get more pronounced, thus deteriorating the TDC's linearity.

Phase detection interpolation is not a common architecture in ASIC technology. Although it has the advantage of being a pure digital architecture which simplifies the design and implementation process. The performance achieve by such architectures cannot compete with more sophisticated ones, like DLLs and pulse shrinking. Nevertheless, the linearity values reported by phased clocks are usually in the range of less than 0.2 LSB, making this architecture very attractive for applications where resolution in the range of a few hundreds of picoseconds is required. The resource consumption or area utilization per TDC channel is also reduced when compared to other TDC architectures. Since the architecture can be fully implemented in a digital flow, this architecture is a good candidate for ASIC migration. For the same reason, FPGA and ASIC platforms usually share the same phase detection architectures and issues. The main difference between FPGA and ASIC phased clocks architectures is the PLL block, which is already included in modern FPGA but, in ASIC platforms, must be designed and implemented, increasing the overall TDC architecture complexity. Recently, the work by Wang et al. [2.34] proposed a phased detection architecture where, instead of sampling the time event with the phased clocks, it was the phased clocks which were sampled by the time event (Figure 2.6). This enables for power savings since the sampling process will only occur once per time event. Furthermore, it removes the need of the synchronization stage, as presented in Figure 2.5, since the sampling is done in the same clock domain, enabling area savings. The TDC was implemented in a 130 nm process technology, reporting a 780 ps resolution, with a maximum bin variation of +/- 40 ps, corresponding to a DNL of +/-0.05 LSB.

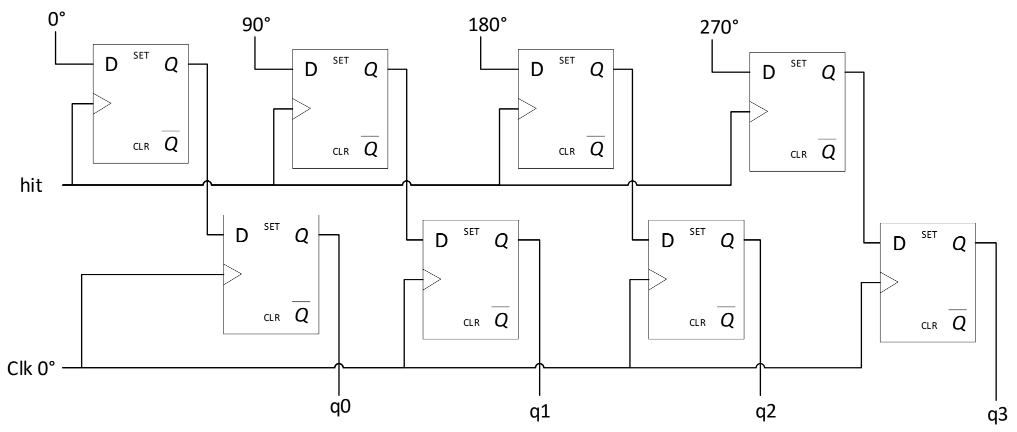


Figure 2.6- ASIC Phased Clocks (Adapted from [2.34])

2.2.4. Tapped-Delay Lines (TDL) and Delay-Locked-Loops (DLL)

Even though in ASIC a multitude of architectures exist, DLL architectures and variants are one of the most popular since they enable high resolution, which can be further improved using n-stage interpolation schemas, for FPGA platforms, the most researched and adopted architecture to achieve high performance TDCs is based on tapped-delay-lines (TDL) [2.7], [2.8], [2.21], [2.31], [2.35]-[2.104]. The design process of these architectures may seem simple however, when high linearity and high performance is required, special precautions must be taken during their implementation phase.

Tapped-Delay Lines

A typical TDL architecture is composed by an input stage, a delay line paired with a sample stage, a decoding block, and a calibration block, as depicted in Figure 2.7. The first and last blocks, the input stage and the calibration stage, are not mandatory. However, with technology scaling down and therefore, a higher impact of PVT variations in the propagation delays of the cells, if high precision is required, these blocks must be implemented. Otherwise, the non-linearity errors will deteriorate the TDC performance. The basic principle of operation consists in delaying an event signal throughout a chain of buffers (delay cell or interpolation step). The state of the delay chain is sampled by a reference clock every cycle. This results in a thermometer code that has the information of the number of delay cells that the event signal was capable of traverse in between reference clock cycles. This code is then passed to a decoder that will convert this thermometer code to a binary value corresponding to number of delay cells traversed. This value can be directly used to recover the event timing information, or it can be passed to a calibration table as an index to obtain the calibrated time information. The input stage can be used to manipulate the time event in order to generate multiple pulses, allowing multiple transitions to be sampled in the delay line, which, using of statistical methods, contributes to an increase of the TDCs precision. The critical blocks are the pair composed by the delay line and the sample line (Figure 2.8), since these define the base resolution of the TDC. The components presented in Figure 2.7 are usually implemented per TDL channel. In multiple chains TDCs, the base blocks are replicated for each channel. For a single TDL channel, the time interval measurement value can be obtained according to:

$$t_{fine} = n * \tau, \quad (2.14)$$

where τ is the propagation delay of each cell element and *n* is the number of cells traversed by the delayed signal.

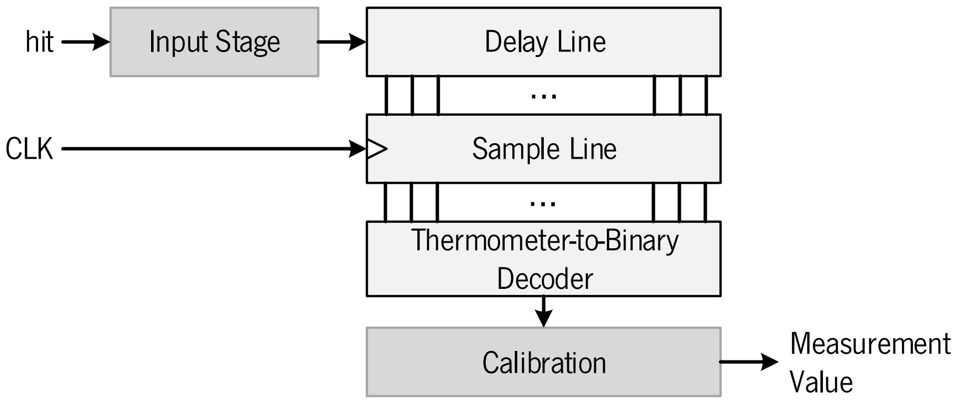


Figure 2.7- TDL TDC Block Diagram

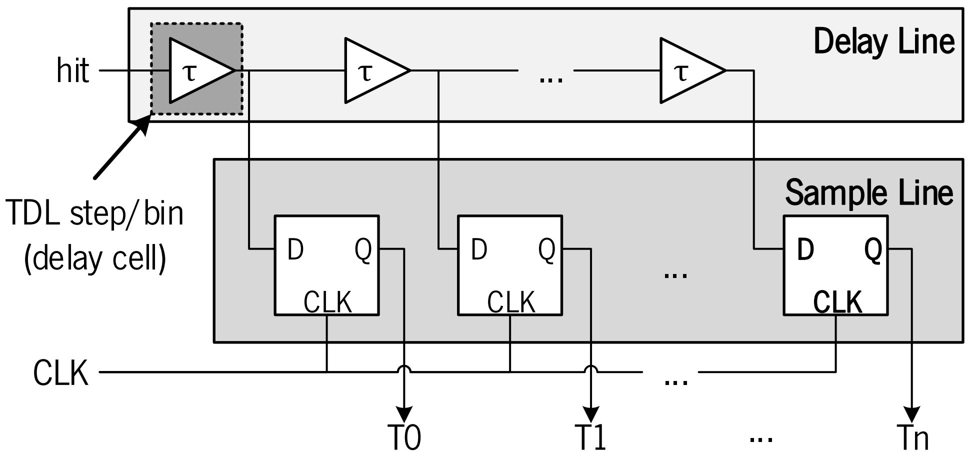


Figure 2.8- TDL architecture

In TDL architectures, the maximum achievable resolution is always dependent on the propagation delay of the cell used as the basic delay element (TDL step) and the clock skew to the sampling registers [2.59].

In ASIC-base implementations, these basic delay cells are usually custom designed so that a specific propagation delay time is achieved. In FPGA-based implementations, one of the available cell resources must be selected. In modern FPGA there are two cells that can potentially be used to implement delay lines, the look-up tables (LUTs) or the Carry4 cells. Carry4 cells offer lower propagation delay and have dedicated routing paths, which are desired characteristics when designing a TDL TDC and therefore, a vast number of reported implementations used these resources as the TDL base step. Nevertheless, there are still some implementations of TDLs using LUTs [2.44], [2.52], which were able to achieve interesting resolutions. In [2.105], flip-flops were used to implement the TDL steps, using the output of a previous flip-flop to clock the next one. Although good linearity has been achieved, the resolution was not as good as the one achieved when using Carry resources to implement the TDL step.

Regardless of the cell used to build the delay chain steps (Carries, LUTs, flip-flops, or custom designed cells), when implementing a TDL architecture, the following issues, divided by variation, must be addressed.

1) Single TDL: TDLs are usually designed to cover a time interval equal to the period of a reference clock. When larger dynamic range are required, the TDL architecture is paired with coarse counters, otherwise a very large number of steps would be required. Since the arrival of the time event to be measured by the TDC is asynchronous to the coarse counting mechanism, the TDL and coarse counter must be synchronized for proper operation and to avoid metastability. Otherwise, the performed measurement may have an error of several coarse counter clock periods. For this reason, a second coarse counter, with a clock signal delayed by 180° is usually implemented and the value outputted by the TDL (sampled value) is used to identify which counter has the correct value, i.e. the value that is not metastable. A metastability error occurs when the hit signal arrives close to the rising edge of the reference clock (used to increment the coarse counter). Therefore, if the TDL value sampled is close to zero or to the maximum TDL value, there is a chance for metastability on the coarse counter. In these scenarios, the value on the second coarse counter, incremented by the 180° delayed clock, is used. When multiple transitions per time event are to be measured by the same TDL, the described method does not work [2.106]. In those scenarios, a methodology like the one proposed in [2.107] should be adopted.

Since the resolution of TDLs is attached to the delay element used, statistical methods are usually employed to overcome this limitation. In the case of single TDLs, this is done using a Wave-Union (WU) launcher, which was first proposed by Wu [2.108]. Research works reporting a 10 ps resolution and 38 ps precision, using a Lattice FPGA to implement a WU TDL can be found in [2.80] and [2.109]. The same architecture, implemented on Xilinx Virtex-5 [2.110] and Spartan-6 [2.90] FPGAs achieved a precision of 32 ps and 14 ps, respectively.

Process fabrication mismatch in carry cells is responsible for high variations on the propagation delay from cell to cell, since basic delay cells with exactly the same propagation delay and characteristics are impossible to achieve (either in FPGA or ASIC). These variations jeopardize the average propagation delay achievable and degrade the overall TDC linearity. To address the issue, the placement of the delay line carry cells must be carefully made. Nevertheless, even with optimal placement, the difference on propagation delays is enough to deteriorate the TDC's linearity. These non-linearity issues must be addressed if a high performance TDC is desired. Therefore, calibration mechanisms capable of reducing the propagation delay variation throughout the TDL must be implemented.

Won and Lee [2.14] propose a method to tune the delay line in order to minimize the carry cells propagation delay mismatch problem. The method consists in sampling the even and odd steps of the TDL in two separated thermometer codes. These thermometer codes are individually converted to binary and finally added to give the TDC measured value. The proposed method was evaluated using three different FPGA platforms, fabricated in different technologies. The best reported result reached a 10.1 ps LSB and a precision under 10 ps.

With the introduction of Xilinx UltraScale+ architectures, the FPGA base structure changed and with it, new possibilities were unlocked to improve the TDL resolution. The UltraScale+ architecture allows the capture of both carry and sum results of the same slice. This method enables to theoretically double the achievable resolution when compared to the traditional sampling method, since for the same interpolation time, two times more steps exist. Liu et al. [2.20] used this feature to implement a TDL with 2.3 ps resolution and 3.9 ps precision.

All the aforementioned methods were combined with calibration. Calibration in FPGA-based TDL architectures can be achieved using decimation [2.43], [2.61], [2.65], [2.83] and bin-by-bin calibration [2.9], [2.40], [2.45], [2.46], [2.48], [2.53], [2.55], [2.59], [2.67], [2.69]–[2.71], [2.73], [2.82], [2.84], [2.85], [2.111], [2.112]. In ASIC-based TDL architectures, the flexibility of creating a custom-designed cell, allows for the implementation of delay lines using a Voltage Controlled Oscillator (VCO) schema, as depicted in Figure 2.11.

Decimation consists in grouping multiple delay cells in a single step. The number of cells per step varies. The propagation delay of each step is equal to the sum of the propagation delay of the individual cells included in the step. Thus, steps with similar propagation delay can be achieved by grouping different numbers of delay cells, resulting on more uniform steps across the interpolation time. Decimation increases the delay chain linearity but has a negative impact in the TDC resolution, but neither adds processing overhead nor increases the number of resources required [2.21].

The bin-by-bin calibration enables the TDC linearity to be improved without degrading its resolution, which leads to better measurement precision. It consists in building a calibration table based on the real propagation delay of each delay cell used to build the TDL [2.21], [2.35], [2.112]. The real delay of each of the delay cells is obtained by performing a code density test and a calibration table is created with the accumulated real delay for each delay cell of the TDL. When a value (cell number) is obtained from the TDL, it is used as an index to address the calibration table position which contains the calibrated time interval value. This technique requires more resources (BRAM block to implement the calibration table) and extra processing time (typically one extra clock cycle to access the calibration table), however it has no impact on the TDC resolution while improving its linearity. Some research works built these tables prior to the implementation or at system start-up, by pre-building a memory with the calibration values that will stay the same throughout the TDC operation [2.46], [2.69]. In other research works the values presented in the calibration table are dynamically updated, achieving higher precisions since the calibration values are regularly updated based on the PVT conditions [2.35], [2.48], [2.111]. Recently, a technique to calibrate the TDC cells' delays and increase its linearity, while consuming less resources than the bin-by-bin calibration, was proposed by Chaberski et al. [2.86]. The proposed technique is based on the use of dummy cells to control the capacitance load on the different stages of the TDL.

As the number of stages of a TDL increases two main issues must be target, the linearity and the size of the outputted thermometer code. The solution for linearity issues is the implementation of a calibration mechanism, which was already discussed. The main problem with large thermometer codes is the complexity and the latency introduced by the thermometer-to-binary decode module. Larger TDLs require decoders with multiple combinatorial stages, which, depending on the reference clock used, may need multiple cycles to finish the conversion. This limits the maximum achievable sampling rate of the TDC, i.e. increases its deadtime.

Finally, when implementing TDLs, a very common issue is the existence of bubble codes, which are the same as zero delay cells. The problem is typically originated by the clock skew between the sampling flip-flops [2.21], [2.48]. These bubbles have a negative impact on the decoder stage, since it must be capable of correcting the thermometer code before performing the decoding. Therefore, bin realignment based on a histogram obtained using the code density test is usually performed. Other possible solution is to add a bubble removal stage before the decoding block. Recently, Wang et al. [2.103] proposed the use of the number of cells at logic level '1', instead of using a decoding schema which detects the position of the last sampled cell in the delay chain. The premise states that even when a bubble exists, sooner or later, that supposedly zero delay cell will be sampled. Therefore, the relevant information regarding the time event to measure is store in the number of logic '1's in the delay chain rather than the position of the last logic '1'. Therefore, if a decoding schema that counts the number of '1's is implemented, the bubble occurrences can be ignored.

2) Multi-chain TDL: Several research works have focused on implementing multi-chain TDLs to improve the overall performance, resolution and precision of the TDC channel [2.35], [2.74], [2.76]–[2.78], [2.87], [2.92], [2.113], [2.114]. The principle is based on the small discrepancies between multiple TDLs due to process variations (Figure 2.9). If two TDLs are measuring the same time event (hit) the delays of each bin of the TDLs will not be perfectly match. Therefore, the resolution of the TDC channel can be effectively divided by the number of TDLs measuring the same signal. This also reduces the effect that larger bins (usually called ultra-wide bins) have on the measurement, increasing the overall TDC linearity. The results can be further improved by increasing the number of chains used to average a single signal. However, at a higher resource cost, since each TDL usually requires a considerable amount of resources. Moreover, when implementing multiple TDLs, the routing of the time event signal must be done in a way that the offset between channels is minimized.

3) Hybrid TDL: As aforementioned, the linearity of a TDL deteriorates with the increase of the number of delay stages that need to be implemented. To reduce the TDL length, a recent trend combines phased clock architectures with TDL architectures [2.8], [2.9], [2.63], [2.97], [2.115], [2.116]. In such architectures (Figure 2.10), the first stage measurement is done by identifying which was the first phase to detect the time event. A fine measurement is done by a TDL that only covers a time interval between two clock phases rather than the entire reference clock period. In [2.9], Szplet et al. reported a resolution of 1.9 ps using this architecture. In this topology, the routing should be done to minimize the skew between clocks since multiple clocks will be sampling the same TDL. Otherwise, the TDL behavior will be dependent on the sampling phase.

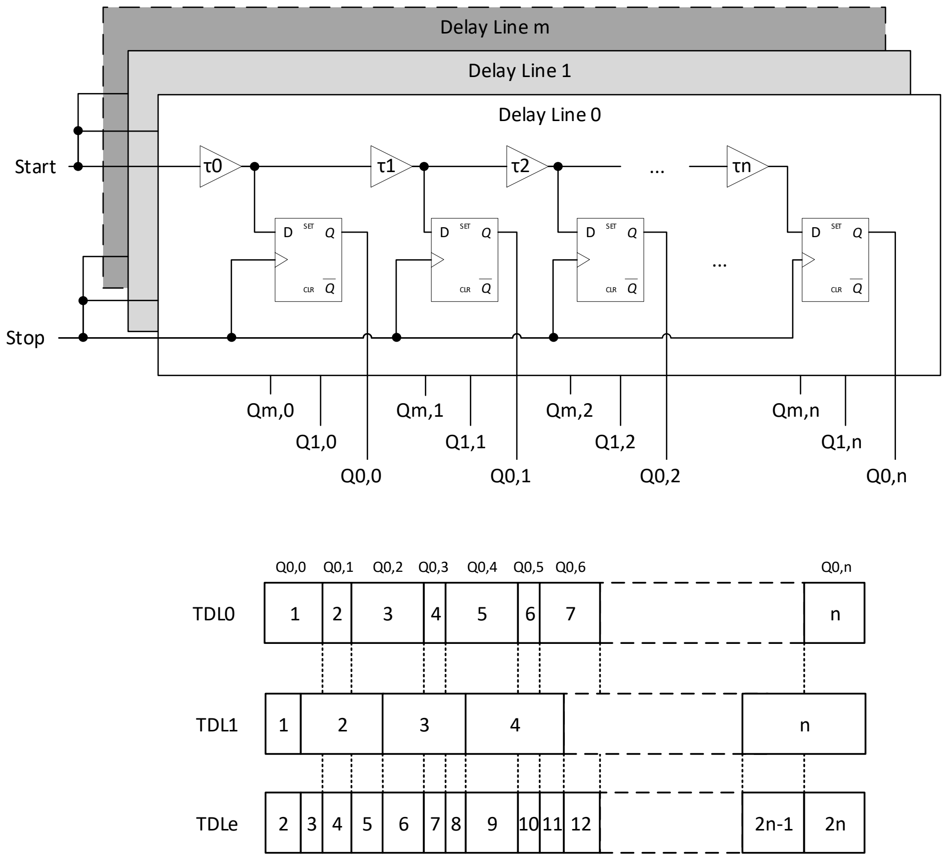


Figure 2.9- Multi-Chain TDL architecture

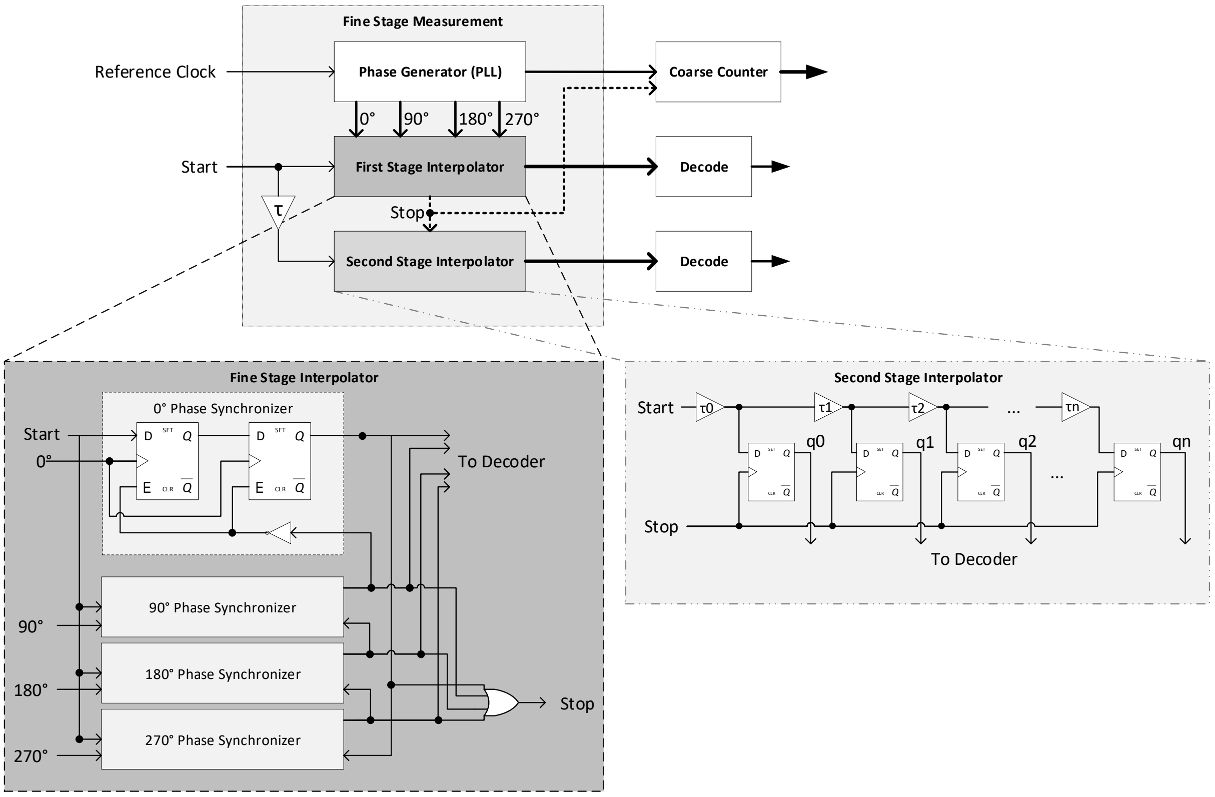


Figure 2.10- Hybrid TDL architecture

Delay-Locked-Loops (DLL)

As mentioned in the previous sections, DLL architectures share similarities with TDLs regarding its based structure. The main differences are in the type of cells used and the auxiliary circuits required to sustain a stable oscillation, which automatically calibrates and shields the delay line against PVT variations. Apart from the TDC, for proper calibration of the oscillation frequency of the DLL, a phase detector and a charge pump, or a PLL, must be implemented to control the supply voltage of the cells used in the delay chain (see Figure 2.11 for a DLL architecture overview). The drawback is the typical high frequency operation of such architectures which can reach several hundreds of MHz [2.1], [2.117]–[2.119], resulting in an increase in the power consumption and the need for designing custom cells and circuitry which increases system's complexity.

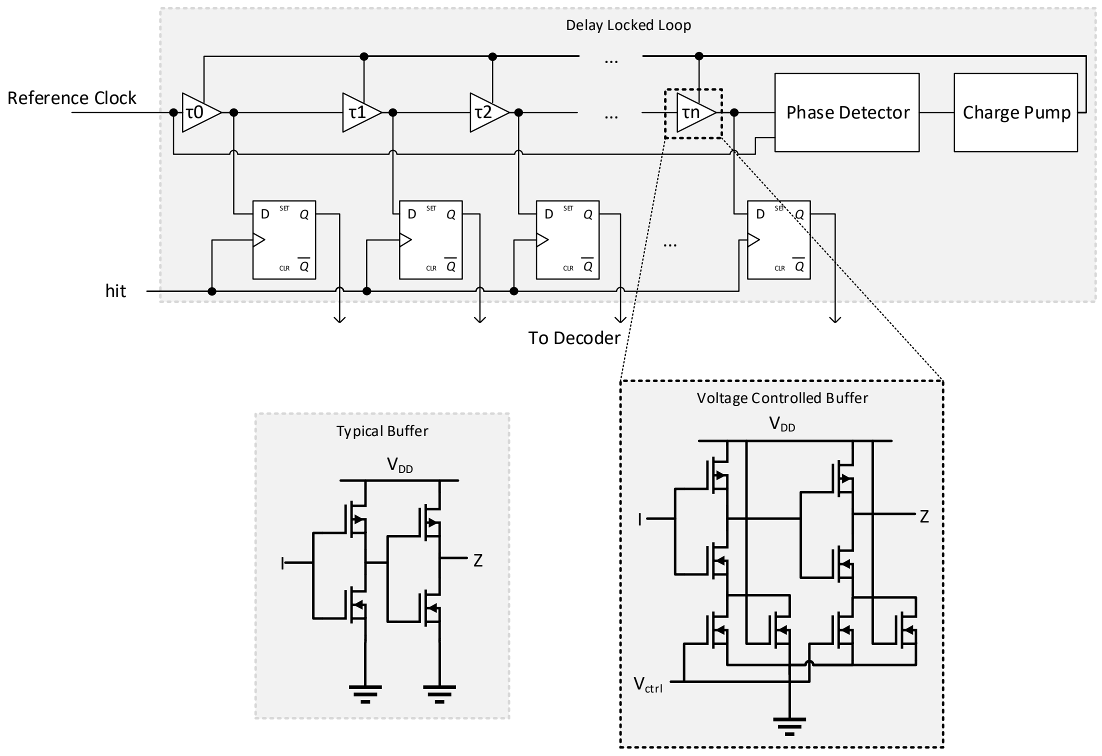


Figure 2.11- General DDL architecture (Single Stage)

The equations used to calculate the fine time measured by the DLL are similar to the ones used in TDLs. However, when the output of the last DLL stage is clocking a cycle counter to increase the DLL range, the time measured can be determined using the following equation:

$$t = N_{cc} * T_{CLK} + N_f * \tau, \qquad (2.15)$$

where t is the time interval measured, N_{cc} is the number of counts on the cycle counter, T_{CLK} is the DLL oscillation period, N_f is the fine value sampled in the DLL, and τ is the DLL resolution, given by equation (2.16) (N_D is the total number of delay elements used to build the DLL).

$$\tau = \frac{T_{CLK}}{2N_D}, \qquad (2.16)$$

The range of resolutions achievable with this architecture is highly dependent on the design of the delay cell and the technology used. In [2.3] an 8.87 ps resolution with 9.8 ps precision was reported. The work by Perktold and Chritiansen [2.118] reported a 5 ps resolution with 3 ps precision, in a 130 nm technology, by adding a calibration buffer stage at the output of each step of the DLL. The buffers were used subdivide each step in four bins, enabling the reduction of the LSB size. This subdivision, together with the PVT compensation, enabled high resolution and precision to be achieved.

Recently, a two-stage DLL TDC, with a similar structure to the one presented in [2.118] (see Figure 2.12), but using resistive interpolation to subdivide the DLL steps to lower sizes was presented [2.1]. The research work reported a 9.3 ps LSB resolution (after calibration) and a 4 ps precision in a 180 nm process technology. Although good resolution and non-linearity values lower than one LSB for both DNL and INL were achieved, the architecture requires the implementation of a PLL for generating an 800 MHz clock and an external reference clock of 50, 100, or 200 MHz to operate.

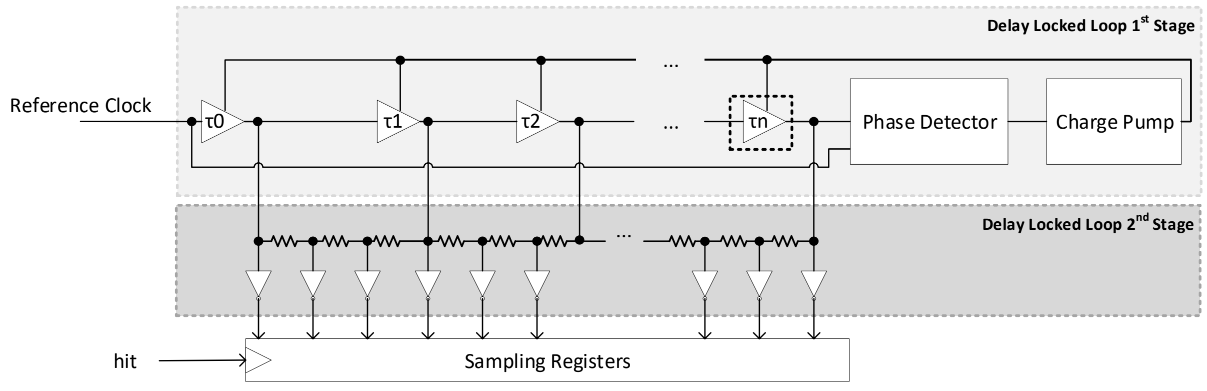


Figure 2.12- Two-Stage DDL architecture

2.2.5. Time-Amplifier (TA) TDCs

A common practice for ASIC-based TDCs is the use of time amplifiers as the input stage for a TDC channel to improve the resolution, similar to the use of Wave-Union launchers as the input stage for TDL TDCs in

FPGA platforms. This enables to achieve lower resolutions since the effective resolution will be the LSB of the TDC divided by the amplification factor [2.120]. A resolution of 980 fs in a 65 nm technology have been reported in [2.120]. Molaei and Hajsadeghi [2.121] also implemented a TA TDC, achieving a 5.3 picoseconds precision in a 180 nm process technology.

2.2.6. Differential Delay Lines

An alternative to improve TDC resolution under the intrinsic propagation delay of a cell is to adopt a differential approach. These types of architectures have a resolution equal to the difference between the delay step of two elements. This is obtained by, for example, delaying both the time event to be measured using a TDL and delaying the clock signal for the registers that are sampling the TDL (usually called 2D TDL or Vernier TDL). This way, the resolution is given by the difference between the cells used in the TDL and the cells used to delay the clock signal (see Figure 2.13 where the resolution of the TDC is given by τ1-τ2). TDCs based on the Vernier principle can also be considered as an interpolative TDC, similar to the TDL and DLL architectures.

Nevertheless, since the resolution of the TDC is given not by a single delay element, but rather by the difference between two interpolative stages, these architectures were considered as differential, according to the taxonomy proposed in [2.122]. Another approach using two ring oscillators with slightly different frequencies could also be adopted, as presented in [2.123] and depicted in Figure 2.16.

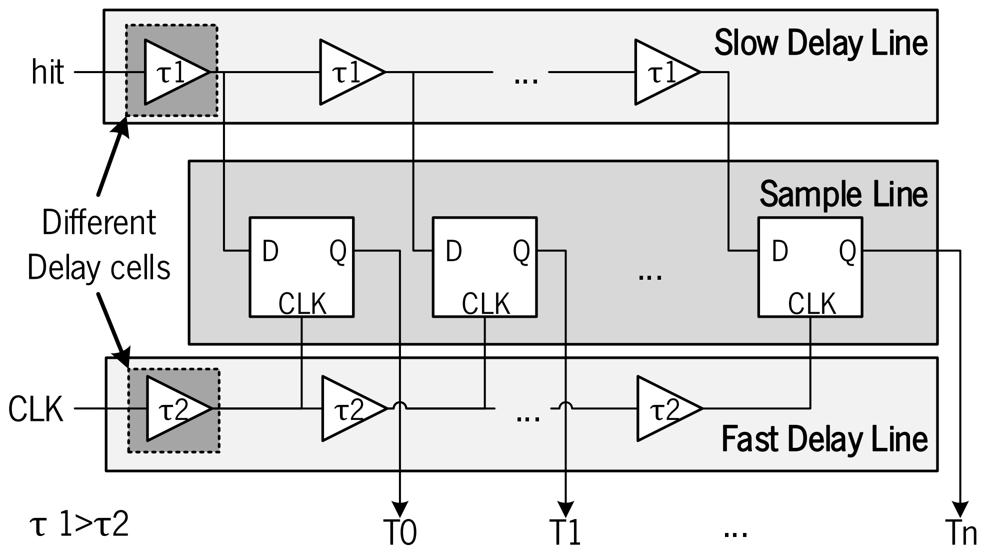


Figure 2.13- Differential Delay line Architecture

1) 2D TDL: On a 2D TDL, depicted in Figure 2.11, the resolution of the TDC is calculated according to equations (2.17) and (2.18). During implementation it must be guaranteed that the propagation delay of the cells used to build the clock delay chain is lower than the propagation delay of the cells used in the hit delay chain. Otherwise, the clock signal will never be able to catch up with the hit signal and the TDC will always return the maximum value. Another issue with this architecture is the length of the delay chains. As in the case of TDLs, the longer the chain is, the higher will be the error due to non-linearities. Furthermore, because it must also be guaranteed that an entire reference clock is covered by the delay line, and because the size of the step in 2D TDLs is lower than in simple TDLs, the delay chains will be longer. This requires more resources and decreases the TDCs precision due to the accumulation of non-linearity errors across the chains.

$$\tau = \tau_1 - \tau_2, \quad (2.17)$$

$$t_{fine} = n * \tau, \quad (2.18)$$

For FPGA implementation, the choice for different cells is limited and therefore, when implementing 2D TDLs, the routing is used to obtain chains with slightly different delays steps. Therefore, getting a uniform delay difference across the two delay chains is a demanding process, hard to replicate. For these reasons, regarding differential TDCs in FPGA platforms, the ring oscillators are often more popular due to its simpler implementation. On the other hand, in ASIC implementation, it is possible to design different delay cells, achieving better resolutions. The research works in [2.124] and [2.125] reported a 30 ps and 5 ps LSB resolution, respectively.

2) Ring Oscillators: Ring oscillator architectures are usually implemented using TDLs in a loop schema [2.112]. Because the parameter responsible for defining the TDC resolution is the difference between the two ring oscillators, the cells' delay mismatch issue is solved [2.112] (see Figure 2.16 and equations (2.19) to (2.22)). The main challenge is to obtain a high accurate and stable oscillation period in order to achieve high performance [2.123]. Two different implementations for ring oscillators TDC were proposed over the last few years for FPGA platforms [2.123], [2.126]. The first one is based on two counters incremented each by one oscillator and a phase detector which samples the counters when the phases of both oscillators align [2.123] (see Figure 2.14). The values on the counters is then added and multiplied by the TDC resolution. The measurement value can be calculated according to equation (2.20):

$$\tau = T_1 - T_2, \quad (2.19)$$

$$T_{ringOscillatorTDC} = (n_1 - 1) * T_1 - (n_2 - 1) * T_2, \quad (2.20)$$

$$t_{maxConv} = \frac{T_1 * T_2}{\tau}, \qquad (2.21)$$

being n_1 the slow and n_2 the fast counters' value respectively. Accordingly, T_1 and T_2 are the slow and fast oscillators' periods. The waveform diagram on Figure 2.15 depicts a typical measurement process for this type of architecture. The main drawback of the architecture is its long conversion time that can reach several hundreds of nanoseconds.

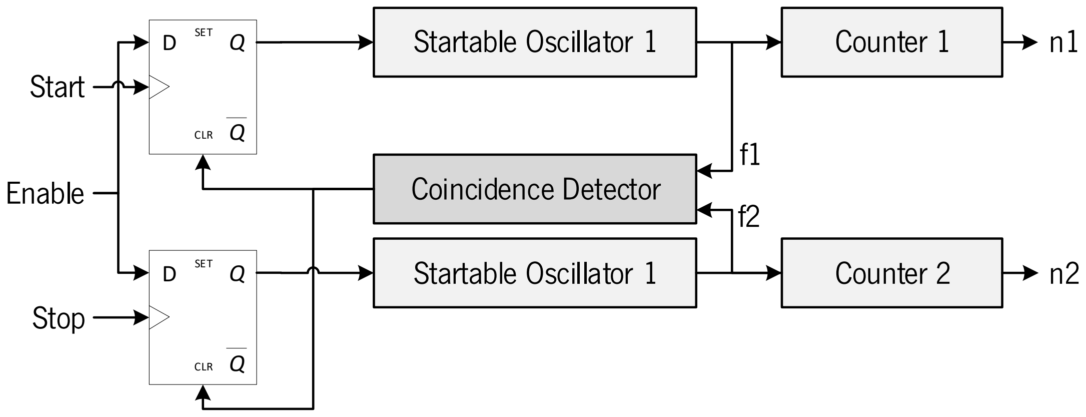


Figure 2.14- Ring Oscillator with two independent counters

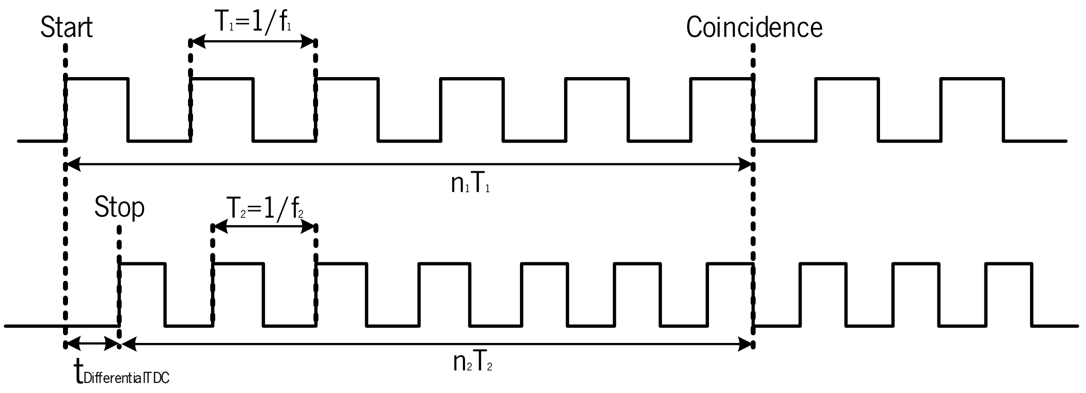


Figure 2.15- Different frequency oscillators Vernier Architecture waveform

The other architecture is based on a single counter and a phase detector. Once the phase of the two oscillators align, the value on the counter is sampled and the measurement value is obtained by multiplying the counter value for the TDCs resolution as demonstrated in equation (2.22):

$$t_{ringOscillatorTDC} = n_{fine} * \tau, \qquad (2.22)$$

being n_{fine} the number of counts in the counter until the fast oscillator is able to overtake the slow one, and τ the resolution of the TDC (given by equation (2.19)) [2.126] (see Figure 2.16).

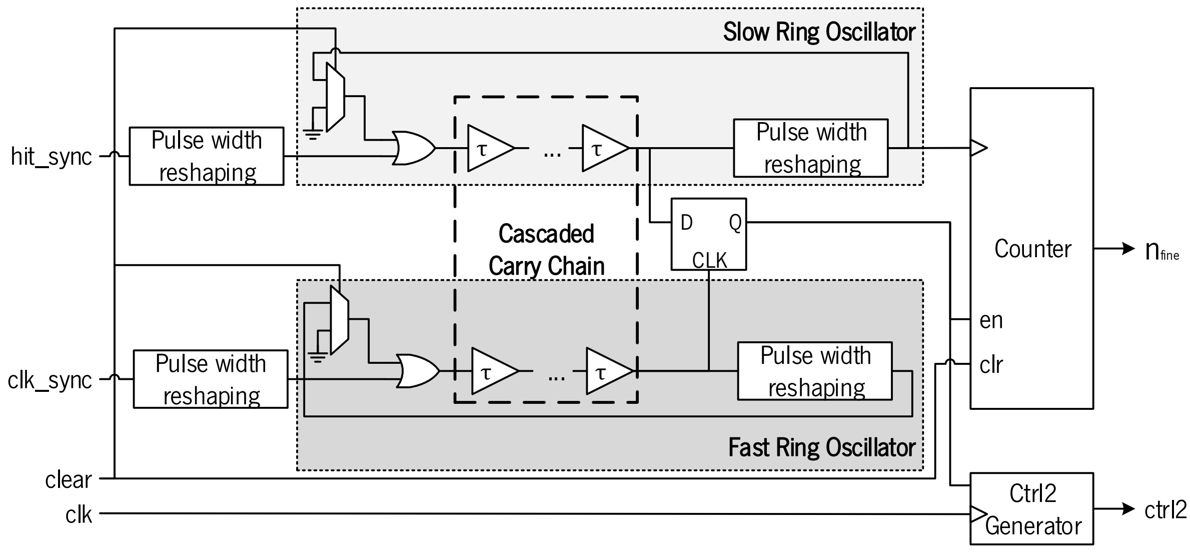


Figure 2.16- Ring Oscillator with single counter

The main drawback of these architectures is the occurrence of pulse shrinking/stretching phenomenon that may happen when implementing ring oscillators that propagate a pulse. If this is not addressed the oscillation behavior will cease. Therefore, a pulse reshaping mechanism, like the one presented in [2.16] and [2.112], needs to be implemented.

Ring oscillators architectures are also known for long measurement dead times. The time required to finalize a conversion can be calculated using equation (2.21). Based on it, depending on the time interval to be measured, the conversion time can reach several microseconds, which limits both the throughput of the TDC and the acceptable input rate. It is possible to decrease the conversion time by reducing the resolution of the TDC or by increasing the oscillation frequency of the ring oscillators. Neither are good solutions since high resolution is usually desirable and maintaining a stable oscillation behavior at high frequencies is not a trivial task.

Ring oscillators topologies in ASIC share the same structure as the ones implemented in FPGA platforms. As most of the ASIC architectures that can be implemented in FPGA, the difference resides on the type of cell used to build the module responsible for defining the TDC resolution, in the case of the ring oscillators. Nguyen et al. [2.127] presented a ring oscillator TDC with 377 ps LSB resolution and 0.8 LSB precision in a 0.18 µm technology.

2.2.7. Pulse Shrinking Architecture

The pulse shrinking phenomenon, cause by the mismatch between the rise and fall times for the cells used in a delay chain [2.128], although being an undesirable effect when trying to propagate a pulse in a ring oscillator, can be used to implement a high resolution TDC. Basically, if a counter is being incremented at each ring oscillator cycle, and if a pulse is propagating and being shrunk every cycle, the number of counted cycles until the oscillation behavior ceases will be proportional to the size of the pulse. Therefore, the resolution is given by the so-called shrinking factor, i.e. the amount the pulse that is being propagated is shrunk every cycle. This shrinking factor is given by the sum of the difference between the rise and fall times of every cell in the looped delay line that is implementing the ring oscillator. These TDCs, although achieving high resolution values, have a high measurement deadtime that is proportional to the pulse size that is being measured. Furthermore, there is an offset associated to the measurement since near the end of the measured, although the pulse is still circulating in the ring, it no longer has the capability of triggering the loop counters clock [2.128]. The measured value can be calculated according to equation (2.23):

$$t_{in} = n * R - t_{offset}, \text{(2.23)}$$

being R the pulse shrinking factor, i.e. the resolution, and t_{offset} the offset size of the pulse circulating on the loop that can no longer trigger the counter's clock (this value has to be obtained by a time consuming experimental procedure or estimated by exhaustive simulations). The main advantage of this architecture is its non-linearity which is bellow half of the LSB.

The shrinking factor of a delay cell can be adjusted by controlling its power supply. Therefore, in FPGA platforms, implementing pulse shrinking architectures is hard since there is no direct mean to control the shrink factor of the ring oscillator. Nevertheless, the work from Chen et al. [2.128] reported a resolution in the range of 110-115 picoseconds with ±1 LSB INL, using a Xilinx XC3S200An FPGA. The authors also proposed a schema to address the offset issue, eliminating the time-consuming process of determining its value through experimental measurements. The proposals to the pulse shrinking architecture changes are depicted in Figure 2.17. Figure 2.18 depicts the waveform diagram of the architecture with the offset canceler mechanism.

Although offering good linearity and relatively high resolutions, the extra complexity of implementation makes this architecture less popular when developing for FPGA platforms since TDLs and Phased clocks can offer the same or even better performances with lower design complexity. In fact, FPGA TDLs have proven to be capable of reaching performance levels similar to the ones achieved by some ASIC pulse shrinking solutions [2.129]–[2.131].

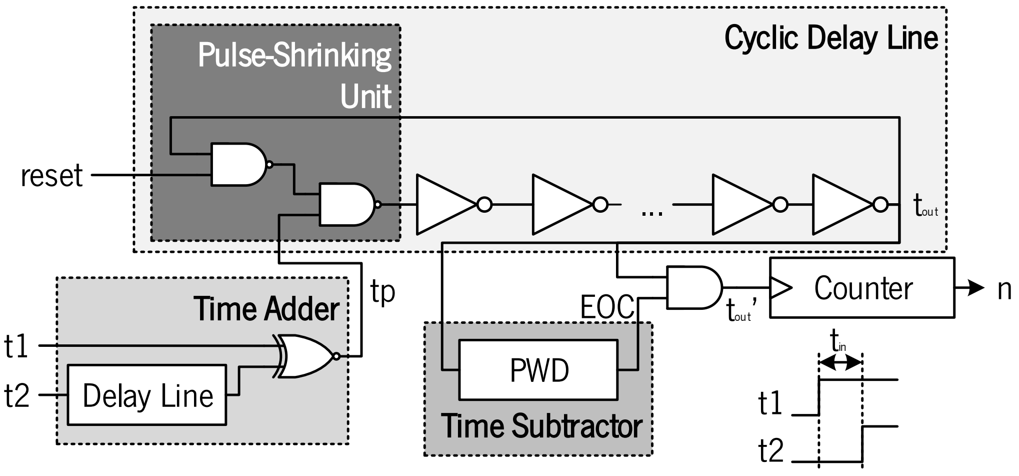


Figure 2.17- Offset canceller pulse-shrinking architecture

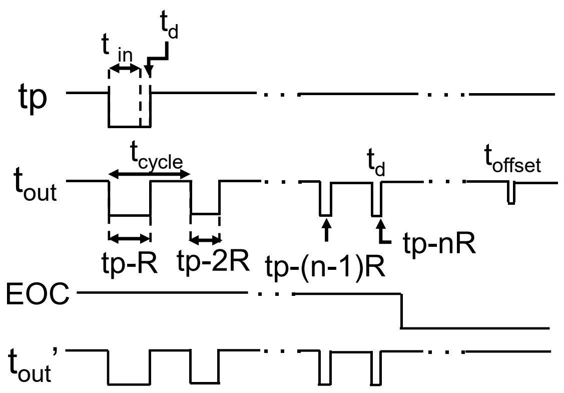


Figure 2.18- Offset canceller pulse-shrinking waveforms

Though pulse shrinking architectures are not suitable for FPGA implementation, these are good solutions when developing for ASIC due to the finer grain control that a custom design cell can has over the shrinking factor, precisely controlling the TDC's resolution. Moreover, the linearity of the TDC is just related with the shrinking factor and not to the individual cells' propagation delays, enabling high performances to be attained.

2.2.8. Summary

Table 2.1 presents a summary on the recent FPGA-based and ASIC-based TDCs, grouped according to the Taxonomy proposed on the journal article J1 [2.122]. Note that a direct comparison between ASIC and FPGA based TDCs cannot be made. Although they may share some applications, usually the goal of these two implementations are not the same. Nevertheless, by the analysis of Table 2.1 it is possible to verify that FPGA-based TDCs' performances are closing the gap to the ASIC-based ones.

2.3. Commercial Devices

Apart from the research that has been done in TDC, it is also important to analyze the available commercial devices. The performance values reported help on understanding the current state of the industry. Table 2.2 presents the most relevant TDC devices available.

2.4. Conclusion

By analyzing the state-of-the-art and the TDC's architectures evolution throughout recent years a set of conclusions can be drawn:

1) First, as technology scales down and FPGA platforms improve, architectures with higher resolutions are expected due to reduction on the cells' propagation delays. However, this will also contribute to enhance the negative effect of PVT variations on the linearity of the TDC. Therefore, calibration mechanisms and methods to reduce PVT variations will assume higher relevance.

2) Second, nowadays ToF applications are requiring multiple TDC channels. For example, in LiDAR applications, capturing larger parts of the scene in a single shot process by having multiple receivers, each with a TDC channel associated, has the advantage of saving time that can be used by the scanning and image processing algorithms. The alternative solution is to sweep a scene point-by-point, which is a much slower process and puts hard time constraints on the image processing algorithms, if a rate of 10-20 frames per second is required. Thus, lower hardware resources TDC architectures are desirable in order to keep both costs and area utilization low, in order to increase systems integration. In FPGA platforms, some research works have already reported 128-channels [2.35] and 256-channels [2.37], [2.38] implementations. However, these works used large FPGA platforms and the resource utilization was on its limits. Recently the research work by Wu and Xu [2.23] proposed an interesting low resource architecture with good linearity and stability. This architecture seems promising for high channel count using low area, low resources platforms and states the need for more architectures focusing on other aspects of a TDC rather than its resolution. Therefore, architectures capable of reducing the length of the

delay chains without compromising system resolution will become more popular and will be the focus of future research works [2.61], [2.97].

3) Third, nowadays market is rapidly changing, and time-to-market constraints are tighter than ever. As stated in [2.4], porting an architecture from one application to another is not an easy task, and requires lengthy manual customizations. Therefore, research regarding reconfigurable and customizable TDCs architectures, is required. Also, tools that can help the designer generate the base core of the TDC would accelerate the development process and contribute for system portability and reutilization. A recent research work [2.55] has addressed this issue with promising results. Due to its characteristics, phased clocks architectures are promising candidates for exploring automated generation of TDCs. Automatically generated TDLs offer an extra challenge due to the non-linearity of the delay-chain. Nevertheless, some research works [2.14], [2.61] have explored a multichain architecture which improves the overall TDL's linearity before calibration. These works are good use cases to test automatic TDC generation tools. With nowadays integration of microprocessors on most FPGA, systems that allow for programmable logic reconfiguration could also be used to achieve automatically generated TDCs. An algorithm to analyze the automatically generated TDL's non-linearity could be implemented on the microprocessor. Depending on the results from the histograms, the chain's hardware could be rearranged automatically in order to improve its linearity and reduce missing codes. In [2.162], a framework is proposed to dynamically control the FPGAs routing with precision. The use of such framework in TDC designs could contribute to achieve high linearity architectures in a simplified way.

4) Fourth, in order to comply with modern application requirements, higher sampling rates architectures are required. Usually TDC architectures based on TDLs or DLL report sampling rates equal to the frequency of the reference clock being used. An alternative to reduce other architectures dead time, such as pulse shrinking and ring oscillators, is to have multiple TDC channels operating in an interleaved schema. However, this solution has a negative impact on resources utilization. Therefore, for applications with high input event rates, these architectures may not present the best solution, being phased clocks and delay lines more appropriated.

5) Fifth, building on the literature analysis, it is expected that applications' requirements will continue to drive TDCs evolution. FPGA-based TDCs have recently reported performance values that can compete with the ones achieved by ASIC TDCs. Therefore, it is expected that FPGA-based TDC would grow in popularity and start to be included on commercial products, rather than just being used in research field experiments or as prototype platforms. A full automated implementation and migration process for FPGA-

based and ASIC-based TDC will certainly increase its popularity, reduce the production cost and ease the use of these systems in a broader range of applications.

The state-of-the-art clearly shows that TDL are the most explored architecture when FPGA platforms are used. On the other hand, in ASIC-based TDCs there is no dominant architecture. Nevertheless, research works reporting the used of DLL, with or without a second stage interpolator to achieve resolutions under the values of the cells' propagation delay, are the most popular architectures. In FPGA-based TDCs, linearity issues are often addressed by introducing a calibration stage to the TDC architecture, either by doing decimation or bin-by-bin calibration. In ASIC-based TDCs, the shielding against PVT variations is achieved through carefully design the delay element and by a DLL schema, locked to a fixed frequency that dynamically adjusts the power supply of the cells.

With the evolution on automotive technology and the appearance of LiDAR sensor for autonomous driving scenarios, it is predictable that TDCs will become more popular, due to their performance on ToF measurements. Although requiring high performance systems, automotive applications have other tighter requirements such as area and power consumption. Therefore, architectures that can combine all these requirements will have the upper hand. In FPGA implementations, this trend can already be identified. The research work by Dinh et al. [2.41] proposes a mixture between a ring oscillator and a second stage interpolator based on a TDL architecture, that enables to reduce hardware resources utilization while maintaining high resolutions. A phased clock architecture with a TDL to cover the time interval in-between phases, proposed in [2.9], is also an indicative to this trend in FPGA platforms.

Although FPGA constraint the TDC design by having the available resources pre-defined, the portability between different FPGAs is greatly enhanced when compare to the ASIC scenario where, if a TDC architecture needs to be ported to another technology, the delay cells must be completely redesigned to agree with the new technology rules. In FPGA-based implementation, it is only necessary to change the name of the cell used to create the delay chain in the HDL (Hardware Description Language) file. Furthermore, being capable of testing an architecture in FPGA and directly migrate it to ASIC improves system testability and reduce the risks associated to the porting. Therefore, research on TDC architectures and technology migration processes would definitely prove advantageous on the design of TDCs for new ToF sensors.

The research on migration of digital FPGA-based TDC architectures to ASIC platforms is scarce, being the research work by Wang et al. [2.34] one of the few architecture proposals that could be directly migrated from FPGA, since the TDC channel was developed using HDL. Thus, digital TDC architectures were studied during this Thesis' research to understand which architecture could be directly migrated to ASIC. Maximum resolution, high sampling rate, power consumption and size were also considered when selecting the TDC architectures, since these requirements are extremely important for LiDAR sensors, which require multiple TDC channels. The selected architectures and design decisions are introduced and discussed in Chapter 3.

Table 2.1 - TDC Literature Review Summary

Ref.-Year	*Technology/ Device*	*LSB*	*Precision*	*DNL*	*INL*	*Measurement Range*	*Dead time*	*Power*	*Nutt Method*	*Resources*
Unit		ps	ps	LSB	LSB	ns	ns	mW		logic units
ASIC TDCs										
[2.3]-12	350nm	8.88	9.8	[-0.9:0.65]	[-2.5:0.84]	4.5	-	85	√	8.88**
[2.124]-00	700nm	30	20	-	±1	3.84	3.84	-	X	9.92**
[2.132]-06	350nm	105	-	-	±0.05	-	-	150	X	8.63**
[2.133]-09	180nm	41	20	-	-	3.4E+03	-	86	√	9**
[2.134]-09	180nm	126 368	-	<0.5	<0.2	[20:4.2E+06] [20:12.3E+06]	-	1.3 1.7	√	0.044**
[2.135]-14	65nm	1.12	0.773	0.6	1.7	0.578	4	15.4	X	0.14**
[2.119]-14	350nm	244	-	±0.5	<0.2	2E+03	-	5	√	0.35**
[2.125]-15	130nm	5	-	0.63	1.47	6.4E+03	100	1.15	√	0.7**
[2.136]-15	350nm	0.61	4.2	-	±7.4	327E+03	1.25E+03	80	X	0.61**
[2.137]-15	65nm	2.64	0.76	-	-	1.8	6.7	0.0035	X	0.03**
[2.1]-16	180nm	15	4	±0.31	±0.67	1.28E+03	20	45	X	0.196**
[2.25]-17	110nm	50	50	-	-	-	17E+03	10	X	-

[2.127]-18	180nm	377	362	1.41	2.31	355	1.5E+03	0.65	X	0.028**
[2.138]-18	65nm	0.45	0.765	0.65	1.25	0.2	20	12.6	X	0.089**
[2.129]-14	350nm	40	2.2	-	±0.6	22	1E+08	0.0016	X	0.025**
[2.118]-14	130nm	5	2.5	[-0.8:0.6]	[-1.2:0.4]	-	-	43	X	-
[2.117]-15	130nm	77	-	-	-	313.6	330	0.004	X	-
[2.139]-16	180nm	15	20	±0.31	±0.67	1,28E+03	<20	45	X	0.195**
[2.140]-17	180nm	586	-	[-0.18:0.05]	-	6.8E+06	-	0.3155	X	0.026**
[2.141]-17	350nm	320	233	±0.68	[-1.23:1.19]	2,55E+03	-	10.9	X	0.152
[2.34]-18	130nm	781	300	±0.05	±0.05	-	-	6.5	X	-
[2.131]-19	180nm	2	1.44	1.5	4.2	130	303	18	√	0.08**
[2.121]-19	180nm	5.3		0.9	2.8	-	34	1.1	X	0.05**

FPGA-based TDCs

Phased Clocks TDCs

[2.33]-14	Artix-7	156	56	±0.32	±1	-	-	-	X	347
[2.28]-14	Virtex-5	625	255	[-0.06:0.07]	[-0.07:0.07]	640	-	-	X	436
[2.142]-15	Spartan-6	1000	2600	[0.13:0.52]	0.39	>2E+07	-	-	√	-
[2.38]-16	Kintex-7	89.3	56.2	[0.44:0.87]	[0.44:0.82]	-	4.3	-	√	-
[2.79]-16	Stratix IV	2.5	15.91	[-1.9:1.66]	[-3.79:6.53]	-	-	-	X	-

[2.143]-16	Kintex-7	280	-	±0.4	<0.05	3,7E+04	6.25	-	√	-
[2.27]-17	Spartan-6	1000	-	-	-	4,096E+03	32	-	√	-
[2.19]-17	Cyclone III	1190	-	-	-	1,95E+03	4.76	-	X	-
[2.144]-17	Kintex-7	780	250	±0.3	-	1E+05	-	-	√	-
[2.145]-17	Artix-7	400	165	<0.3	<0.3	2,1E+05	0	-	√	-
[2.113]-17	Stratix IV	2.5	6.72	[-0.56:0.46]	[-2.98:3.23]	16.5	19.75	212	X	35675
[2.29]-17	Kintex-7	-	280	[0.13:0.31]	-	3,7E+04	-	20	√	1303
[2.146]-18	Zynq-7000	138	73.6	-	-	1.45E+06	-	-	X	-
					TDL TDCs					
[2.108]-08	Cyclone II	30	25	-	-	-	5	-	X	-
[2.147]-09	Cyclone II	20.1	50	-	-	-	0	-	√	-
[2.148]-11	Virtex-6	9.8	14.24	[-1:1.5]	[-2.25:1.61]	1,0E+07	3.3	-	√	-
[2.149]-12	Virtex-5	30	15	[-1:3]	±4	-	30	-	√	-
[2.150]-13	Virtex-6	10	19.6	±2	±2.5	1,0E+05	3.3	-	√	-
[2.115]-13	Spartan-6	1.14	6	-	19.36	1,0E+10	2000	750	√	13.44**
[2.151]-13	Virtex-5	50	29	[-0.47:0.62]	[-0.87:0.68]	-	-	-	X	-
[2.152]-13	Actel A3PE15000	42	16.4	[-1:0.9]	[-1:3.5]	6,553E+05	100	-	√	-
[2.114]-14	Spartan-6	1	6	[-1:2.91]	[-14.1:15.7]	4,28E-01	3	-	X	-

[2.153]-14	Virtex-5	-	16.3	[-0.9:3]	[-1.5:5]	2.86	0.15	-	X	-
[2.91]-14	Kintex-7	22.7	85.7	<3	<4	5,24E+03	30	-	√	-
[2.63]-14	Spartan-3	45	150	-	-	1,0E+09	-	-	√	-
[2.49]-15	Virtex-4	120	65	0.04	0.017	8,0E+06	2.5	-	X	-
[2.154]-15	Virtex-6	347	18	-	[-0.12:0.11]	-	5	2.7	X	-
[2.97]-15	Spartan-3E	30	-	-	-	-	-	-	√	-
[2.39]-15	Virtex-6	23.9	24	[-1:3]	±3	>1E+09	30	-	√	-
[2.104]-15	Kintex-7	8.6	20	-	-	1E+10	200	<5000	√	-
[2.87]-15	Virtex-6	1.7	4.2	[-1:4]	[-9.8:6.2]	6.25	-	-	√	-
[2.31]-15	Virtex-5	38	15	[-1:1.4]	-	-	10	-	√	-
[2.35]-15	Kintex-7	8.7	-	[0:4.6]	-	360	1.47	-	√	-
[2.75]-15	Kintex-7	17.6	15	[-1:0.8]	±0.8	-	-	-	√	-
[2.155]-16	Virtex-4	20.5	7.2	<0.25	<0.25	-	-	-	X	-
[2.21]-16	Kintex-7 UltraScale	3.29	4.2	[0:4]	[-1.5:1.9]	440	4	-	√	-
[2.20]-16	Kintex-7 UltraScale	2.3	3.9	-	-	440	4	-	√	-
[2.78]-16	Kintex-7	1.28	3.1	7	-	20	-	-	√	-
[2.80]-16	Lattice	10	38	[-1:2.7]	[-0.5:9]	1,62E+14	-	-	√	-

	ECP3-150									
[2.14]-16	Kintex7	10.6	8.13	[-1:1.45]	[-1.23:4.3]	-	5	-	√	2218
	Virtex-6	10.1	9.82	[-1:1.18]	[-3.03:2.46]		5			2218
	Spartan-6	16.7	12.75	[-1:1.22]	[-0.7:2.54]		5			1048
[2.9]-16	Kintex-7	1.9	4.5	[-1:4.2]	[-12:5.8]	2,62E+14	87.7	1550	√	47596
[2.156]-16	Kintex-7	23	11	[-1:2.9]	[-1:4.73]	8,0E+09	-	-	√	-
[2.82]-16	Virtex-6	10	12.83	[-1:1.91]	[-2.2:3.93]	20	-	-	√	-
[2.83]-16	Spartan-6	-	25	-	-	4.5	<100	-	√	-
[2.92]-16	Virtex-5	5.8	20	-	-	$45*10^9$	-	-	√	-
[2.42]-16	Zynq-7020	68	-	±0.7	[-0.6:0.5]	420	17	-	√	3470
[2.43]-16	Spartan-6	80	80	<0.5	<0.5	2.5	-	-	X	-
[2.51]-16	Artix-7	40	-	-	-	640	45	-	√	-
[2.58]-16	Virtex-5	10	15	-	-	1,07E+10	-	-	X	256*
[2.157]-17	Virtex-5	7.4	6.8	±0.74	±1.52	5,07E+11	-	1113	X	3372
[2.98]-17	Virtex-7	1.15	3.5	[-0.98:3.5]	[-5.9:3.1]	12	8	-	√	19666
[2.84]-17	Artix-7	33	12.86	±0.6	±0.8	-	10	-	√	1056
[2.158]-17	Virtex-5	60	-	[-0.66:0.65]	[-0.54:0.24]	-	-	-	X	-
[2.77]-17	Zynq-7000	5	5.8	[0:3.4]	-	50	2.63	-	√	8832

[2.74]-17	Kintex-7	2.45	3.9	[-1:5.5]	18.8	1,183E+05	3.6	821	√	8983
[2.45]-17	Virtex-7	10.5	5.11	[-0.38:0.87]	[-1.23:1.02]	-	-	-	√	-
[2.69]-17	Spartan-6	1.95	21.5	-	-	2.14E+09	8	-	√	-
[2.81]-18	Spartan-6	25.6	37	[-0.9:1.23]	[-0.43:2.96]	-	8.69	131	√	415*
[2.65]-18	Kintex-7	3.17	4.3	4.4	[-1.7:2.1]	50	-	-	√	-
[2.41]-18	Spartan-6	19	20	-	-	3	-	-	X	-
[2.48]-18	Cyclone IV	45	18	[-0.5:0.13]	[-0.48:0.37]	7.5	13.3	-	√	-
[2.53]-18	Spartan-6	15	21	-	-	4.18E+06	8	-	√	-
[2.67]-18	Actel APA1000	550	180	±0.2	[-0.37:0.2]	6,4E+03	-	-	√	-
Differential TDCs										
[2.159]-15	Spartan-3AN	23	-	[-0.5:0.43]	-	-	-	-	X	-
[2.15]-15	Actel SmartFusion	63.3	61.7	[-0.55:0.28]	[-0.72:0.63]	5	1410	-	X	-
[2.16]-17	Stratix III	31	35	[-0.08:0.073]	±0.09	-	256	-	√	423
[2.126]-17	Stratix III	20	35	±1	±1	-	-	-	√	-
[2.61]-18	Virtex-7	10.5	14.59	[-0.05:0.08]	[-0.09:0.11]	-	-	-	√	377
	Xilinx UltraScale	5	7.8	[-0.12:0.11]	[-0.15:0.48]					19794

					Pulse Shrinking TDCs					
[2.10]-10	Spartan-3	42	56	[-0.98:0.5]	[-4.17:3.5]	11.5	710	-	X	-
[2.160]-14	Virtex-5	4.56	-	-	-	-	-	-	X	-
[2.128]-16	Spartan-3AN	-	115	-	±1	15	-	-	X	
[2.161]-18	Actel SmartFusion	8.5	42.4	0.36	0.91	10	1042	-	√	-

*-number of slices (for a TDL each slice usually has 8 logic units used in Xilinx FPGAs); **-units in mm2

Table 2.2- TDCs Commercial Devices Summary

Parameter	*Product Name*	*Resolution (ps)*	*Precision (ps)*	*Range*	*#channels*	*System Clock (MHz)*	*Readout rate*	*Interface*	*Reference*
vendor									
Texas Instruments	TDC7201	55	35	Mode1:12 ns - 2 us Mode2:250 ns - 8 ms	2	16	-	SPI	http://www.ti.com/product/TDC7201#features
AMS	AS6500	-	20	0 s - 16 s	4	2-12.5	1.5 MSamples/s	SPI	https://ams.com/as6500
	AS6501	-	10	0 s - 16 s	2	2-12.5	70 MSamples/s	LVDS and SPI	https://ams.com/as6501
	TDC-GPX	-	10	9.8 us	8	40	40 MSamples/s (200 M peak)	28-bit parallel	https://ams.com/tdc-gpx#tab/features

	TDC-GPX2	-	10	0 s - 16 s	4	2-12.5	35 MSamples/s (70 M peak)	Serial LVDS and SPI	https://ams.com/tdc-gpx2
*Maxim integrated**	MAX35101/ MAX35102/ MAX35103	-	20	8 ms	2	-	-	SPI	https://www.maximintegrated.com/en/products/industries/metering-energy-measurement/MAX35101.html/
Cronologic	TimeTagger 4	500	-	8 ms	4	-	48 MHits/s	PCIe 1.1	https://www.cronologic.de/time_measurement/timetag/timetagger42g/
	HPTDC	25 to 12,800		419 us	8	78.125	4 MHits/s	PCIe 2.2	https://www.cronologic.de/time_measurement/tdc/hptdc/
	xTDC4	13	-	218 us (14 ms extended)	4	-	48 MHits/s	PCIe 1.1	https://www.cronologic.de/time_measurement/tdc/xtdc4/

*for ultrasonic heat meters and flow meters markets

References

[2.1] J. Mauricio, D. Gascón, D. Ciaglia, S. Gómez, G. Fernández, and A. Sanuy, “MATRIX: a 15 ps resistive interpolation TDC ASIC based on a novel regular structure,” J. Instrum., vol. 11, no. 12, pp. C12047–C12047, Dec. 2016.

[2.2] R. A. Dias et al., “Real-Time Operation and Characterization of a High-Performance Time-Based Accelerometer,” J. Microelectromechanical Syst., vol. 24, no. 6, pp. 1703–1711, Dec. 2015.

[2.3] J.-P. Jansson, “A stabilized multi-channel CMOS time-to-digital converter based on a low frequency reference,” University of Oulu, 2012.

[2.4] 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 Trans. Electron Devices, vol. 63, no. 1, pp. 235–251, Jan. 2016.

[2.5] R. Szplet, R. Szymanowski, and D. Sondej, “Measurement Uncertainty of Precise Interpolating Time Counters,” IEEE Trans. Instrum. Meas., pp. 1–9, 2019.

[2.6] J.-P. Jansson, A. Mantyniemi, and J. Kostamovaara, “Multiplying delay locked loop (MDLL) in time-to-digital conversion,” in 2009 IEEE Intrumentation and Measurement Technology Conference, 2009, pp. 1226–1231.

[2.7] Z. Jachna, R. Szplet, P. Kwiatkowski, and K. Różyc, “Permanently calibrated interpolating time counter,” Meas. Sci. Technol., vol. 26, no. 1, p. 015006, Jan. 2015.

[2.8] R. Szplet, P. Kwiatkowski, Z. Jachna, and K. Rozyc, “Precise three-channel integrated time counter,” in 2015 Joint Conference of the IEEE International Frequency Control Symposium & the European Frequency and Time Forum, 2015, pp. 575–578.

[2.9] R. Szplet, P. Kwiatkowski, Z. Jachna, and K. Rozyc, “An eight-channel 4.5-ps precision timestamps-based time interval counter in FPGA chip,” IEEE Trans. Instrum. Meas., vol. 65, no. 9, pp. 2088–2100, 2016.

[2.10] R. Szplet and K. Klepacki, “An FPGA-Integrated Time-to-Digital Converter Based on Two-Stage Pulse Shrinking,” IEEE Trans. Instrum. Meas., vol. 59, no. 6, pp. 1663–1670, Jun. 2010.

[2.11] P. Kwiatkowski and R. Szplet, “Time-to-Digital Converter with Pseudo-Segmented Delay Line,” in 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 2019, pp. 1–6.

[2.12] R. Plassche, “Specifications of converters,” in CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters, Boston, MA: Springer US, 2003, pp. 57–64.

[2.13] S. Cova and M. Bertolaccini, “Differential linearity testing and precision calibration of multichannel time sorters,” Nucl. Instruments Methods, vol. 77, no. 2, pp. 269–276, Jan. 1970.

[2.14] J. Y. Won and J. S. Lee, “Time-to-Digital Converter Using a Tuned-Delay Line Evaluated in 28-, 40-, and 45-nm FPGAs,” IEEE Trans. Instrum. Meas., vol. 65, no. 7, pp. 1678–1689, Jul. 2016.

[2.15] J. Zhang and D. Zhou, “A new delay line loops shrinking time-to-digital converter in low-cost FPGA,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 771, pp. 10–16, 2015.

[2.16] K. Cui, Z. Ren, X. Li, Z. Liu, and R. Zhu, “A High-Linearity, Ring-Oscillator-Based, Vernier Time-to-Digital Converter Utilizing Carry Chains in FPGAs,” IEEE Trans. Nucl. Sci., vol. 64, no. 1, pp. 697–704, 2017.

[2.17] M. Arkani, “A High Performance Digital Time Interval Spectrometer: An Embedded, FPGA-Based System With Reduced Dead Time Behaviour,” Metrol. Meas. Syst., vol. 22, no. 4, pp. 601–619, Dec. 2015.

[2.18] Q. Guo, R. Feng, Y. Wu, and N. Yu, “Measurement of the AFDX switch latency based on FPGA,” in 2016 IEEE International Conference on Aircraft Utility Systems (AUS), 2016, pp. 45–49.

[2.19] D. N. Grigoriev, P. V. Kasyanenko, E. A. Kravchenko, A. G. Shamov, and A. A. Talyshev, “A 32-channel 840Msps TDC based on Altera Cyclone III FPGA,” J. Instrum., vol. 12, no. 8, 2017.

[2.20] C. Liu, Y. Wang, P. Kuang, D. Li, and X. Cheng, “A 3.9 ps RMS resolution time-To-digital converter using dual-sampling method on Kintex UltraScale FPGA,” 2016 IEEE-NPSS Real Time Conf. RT 2016, pp. 1–3, 2016.

[2.21] 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 Trans. Nucl. Sci., vol. 63, no. 5, pp. 2632–2638, 2016.

[2.22] R. Szplet, “Time-to-Digital Converters,” in Design, Modeling and Testing of Data Converters, P. Carbone, S. Kiaei, and F. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 211–246.

[2.23] J. Wu and J. Xu, “A Novel TDC Scheme: Combinatorial Gray Code Oscillator Based TDC for Low Power and Low Resource Usage Applications,” in 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), 2019, pp. 1–7.

[2.24] J. Kalisz, “Review of methods for time interval measurements with picosecond resolution,” Metrologia, vol. 41, no. 1, pp. 17–32, Feb. 2004.

[2.25] F. Cossio, “A mixed-signal ASIC for the readout of Gas Electron Multiplier detectors design review and characterization results,” in 2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), 2017, pp. 33–36.

[2.26] D. Calvo, “1 ns time to digital converters for the KM3NeT data readout system,” in AIP Conference Proceedings, 2014, vol. 1630, no. 2014, pp. 98–101.

[2.27] Z. Li et al., “Development of an integrated four-channel fast avalanche-photodiode detector system with nanosecond time resolution,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 870, no. November 2016, pp. 43–49, 2017.

[2.28] A. Balla et al., “The characterization and application of a low resource FPGA-based time to digital converter,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 739, pp. 75–82, 2014.

[2.29] Y. Sano, Y. Horii, M. Ikeno, O. Sasaki, M. Tomoto, and T. Uchida, “Subnanosecond time-to-digital converter implemented in a Kintex-7 FPGA,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 874, no. February, pp. 50–56, 2017.

[2.30] H. Huang and W. Chou, “Hysteresis Switch Adaptive Velocity Evaluation and High-Resolution Position Subdivision Detection Based on FPGA,” IEEE Trans. Instrum. Meas., vol. 64, no. 12, pp. 3387–3395, Dec. 2015.

[2.31] H. H. Fan, P. Cao, S. Bin Liu, and Q. An, “TOT measurement implemented in FPGA TDC,” Chinese Phys. C, vol. 39, no. 11, 2015.

[2.32] W. Yonggang, C. Xinyi, L. Deng, Z. Wensong, and L. Chong, “A linear time-over-threshold digitizing scheme and its 64-channel DAQ prototype design on FPGA for a continuous crystal PET detector,” IEEE Trans. Nucl. Sci., vol. 61, no. 1, pp. 99–106, 2014.

[2.33] T. Xiang et al., “A 56-ps multi-phase clock time-to-digital convertor based on Artix-7 FPGA,” in 2014 19th IEEE-NPSS Real Time Conference, 2014, pp. 1–4.

[2.34] J. Wang et al., “Development of a time-to-digital converter ASIC for the upgrade of the ATLAS Monitored Drift Tube detector,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 880, pp. 174–180, Feb. 2018.

[2.35] C. Liu and Y. Wang, “A 128-Channel, 710 M Samples/Second, and Less Than 10 ps RMS Resolution Time-to-Digital Converter Implemented in a Kintex-7 FPGA,” IEEE Trans. Nucl. Sci., vol. 62, no. 3, pp. 773–783, 2015.

[2.36] W. Pan, G. Gong, and J. Li, “A 20-ps time-to-digital converter (TDC) implemented in field-programmable gate array (FPGA) with automatic temperature correction,” IEEE Trans. Nucl. Sci., vol. 61, no. 3, pp. 1468–1473, 2014.

[2.37] Z. Song, Y. Wang, and J. Kuang, “A 256-channel, high throughput and precision time-to-digital converter with a decomposition encoding scheme in a Kintex-7 FPGA,” J. Instrum., vol. 13, no. 05, pp. P05012–P05012, May 2018.

[2.38] Y. Wang, P. Kuang, and C. Liu, “A 256-channel multi-phase clock sampling-based time-to-digital converter implemented in a Kintex-7 FPGA,” in Conference Record - IEEE Instrumentation and Measurement Technology Conference, 2016, vol. 2016-July.

[2.39] B. Qi et al., “A compact readout electronics for the ground station of a quantum communication satellite,” IEEE Trans. Nucl. Sci., vol. 62, no. 3, pp. 883–888, 2015.

[2.40] W.-S. Choong, F. Abu-Nimeh, W. W. Moses, Q. Peng, C. Q. Vu, and J.-Y. Wu, “A front-end readout Detector Board for the OpenPET electronics system,” J. Instrum., vol. 10, no. 08, pp. T08002–T08002, Aug. 2015.

[2.41] V. L. Dinh, X. T. Nguyen, and H.-J. Lee, “A New FPGA Implementation of a Time-to-Digital Converter Supporting Run-Time Estimation of Operating Condition Variation,” in 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1–4.

[2.42] N. Franch, O. Alonso, J. Canals, A. Vila, A. Herms, and A. Dieguez, “A low cost fluorescence lifetime measurement system based on SPAD detectors and FPGA processing,” in 2016 Conference on Design of Circuits and Integrated Systems (DCIS), 2016, pp. 1–6.

[2.43] L.-Y. Hsu and J.-L. Huang, “A multi-channel FPGA-based time-to-digital converter,” in 2016 IEEE 21st International Mixed-Signal Testing Workshop (IMSTW), 2016, pp. 1–4.

[2.44] H. Y. T. To et al., “A Novel Programmable On-chip Voltage Droop Detector for FPGA Applications,” in 2016 IEEE 66th Electronic Components and Technology Conference (ECTC), 2016, pp. 2009–2015.

[2.45] H. Chen, Y. Zhang, and D. D.-U. Li, “A Low Nonlinearity, Missing-Code Free Time-to-Digital Converter Based on 28-nm FPGAs With Embedded Bin-Width Calibrations,” IEEE Trans. Instrum. Meas., vol. 66, no. 7, pp. 1912–1921, Jul. 2017.

[2.46] K. Katoh et al., “A Small Chip Area Stochastic Calibration for TDC Using Ring Oscillator,” J. Electron. Test., vol. 30, no. 6, pp. 653–663, Dec. 2014.

[2.47] D. R. E. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, “An Experimental Evaluation and Analysis of Transient Voltage Fluctuations in FPGAs,” IEEE Trans. Very Large Scale Integr. Syst., pp. 1–14, 2018.

[2.48] G. Cao, H. Xia, and N. Dong, “An 18-ps TDC using timing adjustment and bin realignment methods in a Cyclone-IV FPGA,” Rev. Sci. Instrum., vol. 89, no. 5, p. 054707, 2018.

[2.49] F. Nogrette et al., “Characterization of a detector chain using a FPGA-based time-to-digital converter to reconstruct the three-dimensional coordinates of single particles at high flux,” Rev. Sci. Instrum., vol. 86, no. 11, p. 113105, Nov. 2015.

[2.50] J. Jung, Y. Choi, K. bom Kim, S. Lee, and H. Choe, “An improved time over threshold method using bipolar signals,” Phys. Med. Biol., vol. 63, no. 13, p. 135002, Jun. 2018.

[2.51] E. Venialgo et al., “An order-statistics-inspired, fully-digital readout approach for analog SiPM arrays,” in 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD), 2016, pp. 1–5.

[2.52] J. Michel et al., “Electronics for the RICH detectors of the HADES and CBM experiments,” J. Instrum., vol. 12, no. 01, pp. C01072–C01072, Jan. 2017.

[2.53] E. Arabul, J. Rarity, and NaimDahnoun, “FPGA based fast integrated real-time multi coincidence counter using a time-to-digital converter,” in 2018 7th Mediterranean Conference on Embedded Computing (MECO), 2018, pp. 1–4.

[2.54] A. T. Eshghi, S. Lee, M. K. Sadoughi, C. Hu, Y.-C. Kim, and J.-H. Seo, “Generic high resolution PET detector block using 12×12 SiPM array,” Smart Mater. Struct., vol. 26, no. 10, p. 105037, Oct. 2017.

[2.55] N. Lusardi, A. Palmucci, and A. Geraci, “Fully-migratable TDC architecture for FPGA devices,” in 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD), 2016, pp. 1–3.

[2.56] S. Grzelak, L. Wydzgowski, J. Czokow, D. Chaberski, and M. Zielinski, “High precision ΔE effect measurement with the use of ultrasonic-wave-time-of-flight method,” Prz. Elektrotechniczny, vol. 1, no. 11, pp. 85–88, Nov. 2016.

[2.57] W. Pan, G. Gong, Q. Du, H. Li, and J. Li, “High resolution distributed time-to-digital converter (TDC) in a White Rabbit network,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 738, pp. 13–19, Feb. 2014.

[2.58] N. Lusardi, A. Geraci, J. Marjanovic, and M. Gustin, “High-resolution TDL-TDC system for MTCA.4 standard,” in 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD), 2016, pp. 1–4.

[2.59] S. Grzelak, M. Kowalski, J. Czoków, and M. Zieliński, “High Resolution Time-Interval Measurement Systems Applied To Flow Measurement,” Metrol. Meas. Syst., vol. 21, no. 1, pp. 77–84, Mar. 2014.

[2.60] B. Neumeier and D. Schmitt-Landsiedel, “Online Condition Measurement of High Power Solid State Laser Cutting Optics using Ultrasound Signals,” Phys. Procedia, vol. 56, pp. 1252–1260, 2014.

[2.61] H. Chen and D. D.-U. Li, “Multichannel, Low Nonlinearity Time-to-Digital Converters Based on 20 and 28 nm FPGAs,” IEEE Trans. Ind. Electron., vol. 66, no. 4, pp. 3265–3274, Apr. 2019.

[2.62] T. Polzer, F. Huemer, and A. Steininger, “Measuring metastability using a time-to-digital converter,” in 2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2017, pp. 116–121.

[2.63] R. Szplet, P. Kwiatkowski, K. Rozyc, M. Sawicki, and Z. Jachna, “Modular time interval counter,” in 2014 European Frequency and Time Forum (EFTF), 2014, pp. 494–497.

[2.64] M. Pałka et al., “Multichannel FPGA based MVT system for high precision time (20 ps RMS) and charge measurement,” J. Instrum., vol. 12, no. 08, pp. P08001–P08001, Aug. 2017.

[2.65] Y. Wang, Q. Cao, and C. Liu, “A Multi-Chain Merged Tapped Delay Line for High Precision Time-to-Digital Converters in FPGAs,” IEEE Trans. Circuits Syst. II Express Briefs, vol. 65, no. 1, pp. 96–100, Jan. 2018.

[2.66] P. Deng et al., “Readout Electronics of T0 Detector in the External Target Experiment of CSR in HIRFL,” IEEE Trans. Nucl. Sci., vol. 65, no. 6, pp. 1315–1323, 2018.

[2.67] D. Yang et al., “Readout electronics of a prototype time-of-flight ion composition analyzer for space plasma,” Nucl. Sci. Tech., vol. 29, no. 4, p. 60, Apr. 2018.

[2.68] T. Polzer, F. Huemer, and A. Steininger, “Refined metastability characterization using a time-to-digital converter,” Microelectron. Reliab., vol. 80, pp. 91–99, Jan. 2018.

[2.69] E. Arabul, A. Girach, J. Rarity, and N. Dahnoun, “Precise multi-channel timing analysis system for multi-stop LIDAR correlation,” in 2017 IEEE International Conference on Imaging Systems and Techniques (IST), 2017, pp. 1–6.

[2.70] H. Li, T. Xue, G. Gong, and J. Li, “The integration of FPGA TDC inside White Rabbit node,” J. Instrum., vol. 12, no. 04, pp. P04020–P04020, Apr. 2017.

[2.71] S. Grzelak, J. Czoków, M. Kowalski, and M. Zieliński, “Ultrasonic Flow Measurement with High Resolution,” Metrol. Meas. Syst., vol. 21, no. 2, pp. 305–316, Jun. 2014.

[2.72] F. Huemer, T. Polzer, and A. Steininger, “Using a Duplex Time-to-Digital Converter for Metastability Characterization of an FPGA,” in 2018 IEEE 21st International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2018, pp. 141–146.

[2.73] A. Aguilar et al., “Timing Results Using an FPGA-Based TDC with Large Arrays of 144 SiPMs,” IEEE Trans. Nucl. Sci., vol. 62, no. 1, pp. 12–18, Feb. 2015.

[2.74] 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 Trans. Nucl. Sci., vol. 64, no. 10, pp. 2713–2718, Oct. 2017.

[2.75] Y. Wang and C. Liu, “A nonlinearity minimization-oriented resource-saving time-to-digital converter implemented in a 28 nm Xilinx FPGA,” IEEE Trans. Nucl. Sci., vol. 62, no. 5, pp. 2003–2009, 2015.

[2.76] Q. Shen et al., “A multi-chain measurements averaging TDC implemented in a 40 nm FPGA,” 2014 19th IEEE-NPSS Real Time Conf. RT 2014 - Conf. Rec., pp. 6–8, 2015.

[2.77] 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,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 847, no. September, pp. 61–66, 2017.

[2.78] Q. Cao, Y. Wang, and C. Liu, “A Combination of Multiple Channels of FPGA Based Time-to-Digital Converter for High Time Precision,” in Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD) 2016, 2016.

[2.79] P. Chen, Y. Y. Hsiao, and Y. S. Chung, “A high resolution FPGA TDC converter with 2.5 ps bin size and -3.79~6.53 LSB integral nonlinearity,” in Proceedings of the 2nd International Conference on Intelligent Green Building and Smart Grid, IGBSG 2016, 2016, pp. 2–6.

[2.80] C. Ugur, S. Linev, J. Michel, T. Schweitzer, and M. Traxler, “A novel approach for pulse width measurements with a high precision (8 ps RMS) TDC in an FPGA,” J. Instrum., vol. 11, no. 1, 2016.

[2.81] A. Tontini, L. Gasparini, L. Pancheri, and R. Passerone, “Design and Characterization of a Low-Cost FPGA-Based TDC,” IEEE Trans. Nucl. Sci., vol. 65, no. 2, pp. 680–690, Feb. 2018.

[2.82] J. Y. Won, S. Il Kwon, H. S. Yoon, G. B. Ko, J. W. Son, and J. S. Lee, “Dual-Phase Tapped-Delay-Line Time-to-Digital Converter with On-the-Fly Calibration Implemented in 40 nm FPGA,” IEEE Trans. Biomed. Circuits Syst., vol. 10, no. 1, pp. 231–242, 2016.

[2.83] S. Burri, H. Homulle, C. Bruschini, and E. Charbon, “LinoSPAD: a time-resolved 256×1 CMOS SPAD line sensor system featuring 64 FPGA-based TDC channels running at up to 8.5 giga-events per second,” Opt. Sens. Detect. IV, vol. 9899, p. 98990D, 2016.

[2.84] J. Zheng, P. Cao, D. Jiang, and Q. An, “Low-Cost FPGA TDC With High Resolution and Density,” IEEE Trans. Nucl. Sci., vol. 64, no. 6, pp. 1401–1408, 2017.

[2.85] S. Guo, Y. Wang, N. Li, J. Diao, and L. Chen, “Multi-chain time interval measurement method utilizing the dedicated carry chain of FPGA,” in 2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC), 2017, no. 1, pp. 489–492.

[2.86] D. Chaberski, R. Frankowski, M. Zieliński, and Ł. Zaworski, “Multiple-tapped-delay-line hardware-linearisation technique based on wire load regulation,” Meas. J. Int. Meas. Confed., vol. 92, pp. 103–113, 2016.

[2.87] Q. Shen et al., “A 1.7 ps equivalent bin size and 4.2 ps RMS FPGA TDC based on multichain measurements averaging method,” IEEE Trans. Nucl. Sci., vol. 62, no. 3, pp. 947–954, 2015.

[2.88] N. Lusardi, J. W. N. Los, R. B. M. Gourgues, G. Bulgarini, and A. Geraci, “Photon counting with photon number resolution through superconducting nanowires coupled to a multi-channel TDC in FPGA,” Rev. Sci. Instrum., vol. 88, no. 3, 2017.

[2.89] N. Lusardi, A. Abba, F. Caponio, and A. Geraci, “Quantization noise in non-homogeneous calibration table of a TCD implemented in FPGA,” in 2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2014, no. i, pp. 1–5.

[2.90] Y. Wang, C. Liu, X. Cheng, and D. Li, “Spartan-6 FPGA based 8-channel time-to-digital converters for TOF-PET systems,” in 2015 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC 2015, 2016, pp. 1–3.

[2.91] J. Torres et al., “Time-to-Digital Converter Based on FPGA With Multiple Channel Capability,” Nucl. Sci. IEEE Trans., vol. 61, no. 1, pp. 107–114, 2014.

[2.92] D. Chaberski, “Time-to-digital-converter based on multiple-tapped-delay-line,” Measurement, vol. 89, pp. 87–96, Jul. 2016.

[2.93] H. Homulle, F. Regazzoni, and E. Charbon, “200 MS/s ADC implemented in a FPGA employing TDCs,” Proc. 2015 ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays, pp. 228–235, 2015.

[2.94] S. Y. Wang, J. Wu, S. H. Yao, and W. C. Chang, “A Field-Programmable Gate Array (FPGA) TDC for the Fermilab SeaQuest (E906) experiment and its test with a novel external wave union launcher,” IEEE Trans. Nucl. Sci., vol. 61, no. 6, pp. 3592–3598, 2014.

[2.95] M. Pałka et al., “A novel method based solely on field programmable gate array (FPGA) units enabling measurement of time and charge of analog signals in positron emission tomography (PET),” Bio-Algorithms and Med-Systems, vol. 10, no. 1, pp. 41–45, 2014.

[2.96] T. Chujo et al., “Experimental verification of timing measurement circuit with self-calibration,” in 19th Annual International Mixed-Signals, Sensors, and Systems Test Workshop Proceedings, 2014, vol. 1, pp. 1–6.

[2.97] R. Narasimman, A. Prabhakar, and N. Chandrachoodan, “Implementation of a 30 ps resolution time to digital converter in FPGA,” in 2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV), 2015, pp. 12–17.

[2.98] 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,” IEEE Trans. Nucl. Sci., vol. 64, no. 12, pp. 2951–2957, Dec. 2017.

[2.99] A. Aguilar et al., “Optimization of a Time-to-Digital Converter and a coincidence map algorithm for TOF-PET applications,” J. Syst. Archit., vol. 61, no. 1, pp. 40–48, 2015.

[2.100] A. Aguilar et al., “Time of flight measurements based on FPGA using a breast dedicated PET,” J. Instrum., vol. 9, no. 5, 2014.

[2.101] A. Aguilar et al., “Time of flight measurements based on FPGA and SiPMs for PET-MR,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 734, no. PART B, pp. 127–131, 2014.

[2.102] N. Lusardi, F. Garzetti, G. Bulgarini, R. B. M. Gourgues, J. W. N. Los, and A. Geraci, “Single photon counting through multi-channel TDC in programmable logic,” in 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD), 2016, pp. 1–4.

[2.103] 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 Trans. Nucl. Sci., vol. 63, no. 5, pp. 2617–2621, 2016.

[2.104] N. Lusardi and A. Geraci, “8-Channels high-resolution TDC in FPGA,” in 2015 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC 2015, 2016, pp. 1–2.

[2.105] Y.-C. Chen, H.-C. Chang, and H. Chen, “Two-Dimensional Multiply-Accumulator for Classification of Neural Signals,” IEEE Access, vol. 6, pp. 19714–19725, 2018.

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

[2.107] R. Machado, J. Cabral, and F. Alves, “Designing Synchronizers for Nutt-TDCs,” in 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), 2019, pp. 1–6.

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

[2.109] C. Ugur, G. Korcyl, J. Michel, M. Penschuk, and M. Traxler, “264 Channel TDC Platform applying 65 channel high precision (7.2 psRMS) FPGA based TDCs,” in 2013 IEEE Nordic-Mediterranean Workshop on Time-to-Digital Converters (NoMe TDC), 2013, pp. 1–5.

[2.110] N. Lusardi, M. Luciani, and A. Geraci, “Single-chain 4-channels high-resolution multi-hit TDC in FPGA,” in 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD), 2016, pp. 1–4.

[2.111] J. Kuang, Y. Wang, Q. Cao, and C. Liu, “Implementation of a high precision multi-measurement time-to-digital convertor on a Kintex-7 FPGA,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 891, no. February, pp. 37–41, May 2018.

[2.112] K. Cui, X. Li, Z. Liu, and R. Zhu, “Toward Implementing Multichannels, Ring-Oscillator-Based, Vernier Time-to-Digital Converter in FPGAs: Key Design Points and Construction Method,” IEEE Trans. Radiat. Plasma Med. Sci., vol. 1, no. 5, pp. 391–399, Sep. 2017.

[2.113] P. Chen, Y. Hsiao, Y. Chung, W. X. Tsai, and J. Lin, “A 2.5-ps Bin Size and 6.7-ps Resolution FPGA Time-to-Digital Converter Based on Delay Wrapping and Averaging,” IEEE Trans. Very Large Scale Integr. Syst., vol. 25, no. 1, pp. 114–124, Jan. 2017.

[2.114] R. Szplet, D. Sondej, and G. Grzeda, “Subpicosecond-resolution time-to-digital converter with multi-edge coding in independent coding lines,” in 2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014, pp. 747–751.

[2.115] R. Szplet, Z. Jachna, P. Kwiatkowski, and K. Rozyc, “A 2.9 ps equivalent resolution interpolating time counter based on multiple independent coding lines,” Meas. Sci. Technol., vol. 24, no. 3, p. 035904, Mar. 2013.

[2.116] G. Grzęda and R. Szplet, “Time interval measurement module implemented in SoC FPGA device,” Int. J. Electron. Telecommun., vol. 62, no. 3, pp. 237–246, Sep. 2016.

[2.117] I. Diehl et al., “Readout ASIC for fast digital imaging using SiPM sensors: Concept study,” in 2015 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2015, pp. 1–3.

[2.118] L. Perktold and J. Christiansen, “A multichannel time-to-digital converter ASIC with better than 3 ps RMS time resolution,” J. Instrum., vol. 9, no. 01, pp. C01060–C01060, Jan. 2014.

[2.119] T. Watanabe and H. Isomura, “All-digital ADC/TDC using TAD architecture for highly-durable time-measurement ASIC,” in 2014 IEEE International Symposium on Circuits and Systems (ISCAS), 2014, pp. 674–677.

[2.120] J.-C. Lai and T.-Y. Hsu, “Cost-Effective Time-to-Digital Converter Using Time-Residue Feedback,” IEEE Trans. Ind. Electron., vol. 64, no. 6, pp. 4690–4700, Jun. 2017.

[2.121] H. Molaei and K. Hajsadeghi, “A 5.3-ps, 8-b Time to Digital Converter Using a New Gain-Reconfigurable Time Amplifier,” IEEE Trans. Circuits Syst. II Express Briefs, vol. 66, no. 3, pp. 352–356, Mar. 2019.

[2.122] R. Machado, J. Cabral, and F. S. Alves, “Recent Developments and Challenges in FPGA-Based Time-to-Digital Converters,” IEEE Trans. Instrum. Meas., vol. 68, no. 11, pp. 4205–4221, Nov. 2019.

[2.123] M. Maamoun, I. S. Arami, R. Beguenane, A. Benbelkacem, and A. Meraghni, “A 3ps Resolution Time-to-digital Converter in Low-cost FPGA for Laser Rangefinder,” in Proceedings of the World Congress on Engineering, 2017, vol. I, no. figure 2, pp. 7–11.

[2.124] P. Dudek, S. Szczepanski, and J. V. Hatfield, “A high-resolution CMOS time-to-digital converter utilizing a Vernier delay line,” IEEE J. Solid-State Circuits, vol. 35, no. 2, pp. 240–247, Feb. 2000.

[2.125] C. T. Ko, K. P. Pun, and A. Gothenberg, “A 5-ps Vernier sub-ranging time-to-digital converter with DNL calibration,” Microelectronics J., vol. 46, no. 12, pp. 1469–1480, 2015.

[2.126] K. Cui, Z. Liu, R. Zhu, and X. Li, “FPGA-based high-performance time-to-digital converters by utilizing multi-channels looped carry chains,” in 2017 International Conference on Field Programmable Technology (ICFPT), 2017, pp. 223–226.

[2.127] V. Nguyen, D. Duong, Y. Chung, and J.-W. Lee, “A Cyclic Vernier Two-Step TDC for High Input Range Time-of-Flight Sensor Using Startup Time Correction Technique,” Sensors, vol. 18, no. 11, p. 3948, Nov. 2018.

[2.128] C.-C. Chen, C. Hwang, Y. Lin, and G. Chen, “Note: All-digital pulse-shrinking time-to-digital converter with improved dynamic range,” Rev. Sci. Instrum., vol. 87, no. 4, p. 046104, Apr. 2016.

[2.129] 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 Trans. Circuits Syst. II Express Briefs, vol. 61, no. 3, pp. 163–167, Mar. 2014.

[2.130] Yue Liu et al., “A 6ps resolution pulse shrinking Time-to-Digital Converter as phase detector in multi-mode transceiver,” in 2008 IEEE Radio and Wireless Symposium, 2008, pp. 163–166.

[2.131] R. Enomoto, T. Iizuka, T. Koga, T. Nakura, and K. Asada, “A 16-bit 2.0-ps Resolution Two-Step TDC in 0.18-um CMOS Utilizing Pulse-Shrinking Fine Stage With Built-In Coarse Gain Calibration,” IEEE Trans. Very Large Scale Integr. Syst., vol. 27, no. 1, pp. 11–19, Jan. 2019.

[2.132] P. Fischer, I. Peric, M. Ritzert, and T. Solf, “Multi-Channel Readout ASIC for ToF-PET,” in 2006 IEEE Nuclear Science Symposium Conference Record, 2006, pp. 2523–2527.

[2.133] P. Fischer, I. Peric, M. Ritzert, and M. Koniczek, “Fast Self Triggered Multi Channel Readout ASIC for Time- and Energy Measurement,” IEEE Trans. Nucl. Sci., vol. 56, no. 3, pp. 1153–1158, Jun. 2009.

[2.134] T. Watanabe and T. Terasawa, “An all-digital ADC/TDC for sensor interface with TAD architecture in 0.18-um digital CMOS,” in 2009 16th IEEE International Conference on Electronics, Circuits and Systems - (ICECS 2009), 2009, pp. 219–222.

[2.135] K. Kim, W. Yu, and S. Cho, “A 9 bit, 1.12 ps Resolution 2.5 b/Stage Pipelined Time-to-Digital Converter in 65 nm CMOS Using Time-Register,” IEEE J. Solid-State Circuits, vol. 49, no. 4, pp. 1007–1016, Apr. 2014.

[2.136] P. Keranen and J. Kostamovaara, “A Wide Range, 4.2 ps(rms) Precision CMOS TDC With Cyclic Interpolators Based on Switched-Frequency Ring Oscillators,” IEEE Trans. Circuits Syst. I Regul. Pap., vol. 62, no. 12, pp. 2795–2805, Dec. 2015.

[2.137] W. Yu, K. Kim, and S. Cho, “A 0.22 ps rms Integrated Noise 15 MHz Bandwidth Fourth-Order ΔΣ Time-to-Digital Converter Using Time-Domain Error-Feedback Filter,” IEEE J. Solid-State Circuits, vol. 50, no. 5, pp. 1251–1262, May 2015.

[2.138] A. I. Hussein, S. Vasadi, and J. Paramesh, “A 450 fs 65-nm CMOS Millimeter-Wave Time-to-Digital Converter Using Statistical Element Selection for All-Digital PLLs,” IEEE J. Solid-State Circuits, vol. 53, no. 2, pp. 357–374, Feb. 2018.

[2.139] J. Mauricio, D. Gascon, D. Ciaglia, S. Gomez, G. Fernandez, and A. Sanuy, “MATRIX: A novel two-dimensional resistive interpolation 15 ps time-to-digital converter ASIC,” in 2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD), 2016, pp. 1–3.

[2.140] A. Pokhara, J. Agrawal, and B. Mishra, “Design of an all-digital, low power time-to-digital converter in 0.18μm CMOS,” in 2017 7th International Symposium on Embedded Computing and System Design (ISED), 2017, pp. 1–5.

[2.141] J. Wu, W. Zhang, X. Yu, Q. Jiang, L. Zheng, and W. Sun, “A hybrid time-to-digital converter based on residual time extraction and amplification,” Microelectronics J., vol. 63, pp. 148–154, May 2017.

[2.142] T. Suwada, F. Miyahara, K. Furukawa, M. Shoji, M. Ikeno, and M. Tanaka, “Wide dynamic range FPGA-based TDC for monitoring a trigger timing distribution system in linear accelerators,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 786, pp. 83–90, 2015.

[2.143] Y. Sano, M. Tomoto, Y. Horii, O. Sasaki, T. Uchida, and M. Ikeno, “Development of a sub-nanosecond time-to-digital converter based on a field-programmable gate array,” J. Instrum., vol. 11, no. 03, pp. C03053–C03053, Mar. 2016.

[2.144] Y. Sano et al., “Performances of typical high energy physics applications in flash-based field-programmable gate array under gamma irradiation,” J. Instrum., vol. 12, no. 04, pp. C04002–C04002, Apr. 2017.

[2.145] M. Buchele, H. Fischer, F. Herrmann, and C. Schaffer, “The ARAGORN front-end - FPGA based implementation of a time-to-digital converter,” 2016 IEEE Nucl. Sci. Symp. Med. Imaging Conf. Room-Temperature Semicond. Detect. Work. NSS/MIC/RTSD 2016, vol. 2017-Janua, pp. 2–4, 2017.

[2.146] Y. Jia, C. Wang, and H. Shi, “Multi-channel high precision time digital converter system based on equivalent pulse counting,” in 2018 Chinese Control And Decision Conference (CCDC), 2018.

[2.147] J. Wu, “An FPGA wave union TDC for time-of-flight applications,” in 2009 IEEE Nuclear Science Symposium Conference Record (NSS/MIC), 2009, pp. 299–304.

[2.148] H. Menninga, C. Favi, M. W. Fishburn, and E. Charbon, “A multi-channel, 10ps resolution, FPGA-based TDC with 300MS/s throughput for open-source PET applications,” in 2011 IEEE Nuclear Science Symposium Conference Record, 2011, pp. 1515–1522.

[2.149] L. Zhao, X. Hu, S. Liu, J. Wang, and Q. An, “A 16-channel 15 ps TDC implemented in a 65 nm FPGA,” in 2012 18th IEEE-NPSS Real Time Conference, 2012, pp. 1–5.

[2.150] M. Fishburn, L. H. Menninga, C. Favi, and E. Charbon, “A 19.6 ps, FPGA-Based TDC With Multiple Channels for Open Source Applications,” IEEE Trans. Nucl. Sci., vol. 60, no. 3, pp. 2203–2208, Jun. 2013.

[2.151] Y.-H. Chen, “A high resolution FPGA-based merged delay line TDC with nonlinearity calibration,” in 2013 IEEE International Symposium on Circuits and Systems (ISCAS2013), 2013, pp. 2432–2435.

[2.152] Q. Xi, F. Changqing, Z. Deliang, Z. Lei, L. Shubin, and A. Qi, “A low dead time vernier delay line TDC implemented in an actel flash-based FPGA,” Nucl. Sci. Tech., vol. 24, no. 4, p. 40403, 2013.

[2.153] N. Dutton et al., “Multiple-event direct to histogram TDC in 65nm FPGA technology,” Ph.D. Res. Microelectron. Electron. (PRIME), 2014 10th Conf., pp. 1–5, 2014.

[2.154] J. P. Caram, J. Galloway, and J. S. Kenney, “Harmonic ring oscillator time-to-digital converter,” Proc. - IEEE Int. Symp. Circuits Syst., vol. 2015-July, pp. 161–164, 2015.

[2.155] R. Frankowski, M. Gurski, and P. Płóciennik, “Optical methods of the delay cells characteristics measurements and their applications,” Opt. Quantum Electron., vol. 48, no. 3, p. 188, Mar. 2016.

[2.156] Y. Yao, Z. Wang, H. Lu, L. Chen, and G. Jin, “Design of time interval generator based on hybrid counting method,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 832, pp. 103–107, 2016.

[2.157] 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.

[2.158] Y. H. Chen, “A counting-weighted calibration method for a field-programmable-gate-array-based time-to-digital converter,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 854, no. February, pp. 61–63, 2017.

[2.159] M. Abbas and K. Khalil, “A 23ps resolution Time-to-Digital converter implemented on low-cost FPGA platform,” ISSCS 2015 - Int. Symp. Signals, Circuits Syst., pp. 0–3, 2015.

[2.160] C. Chen, S. Meng, Z. Xia, G. Fang, and H. Yin, “Pulse shrinking time-to-digital converter for UWB application,” J. Electron., vol. 31, no. 3, pp. 180–186, 2014.

[2.161] J. Zhang and D. Zhou, “An 8.5-ps Two-Stage Vernier Delay-Line Loop Shrinking Time-to-Digital Converter in 130-nm Flash FPGA,” IEEE Trans. Instrum. Meas., vol. 67, no. 2, pp. 406–414, Feb. 2018.

[2.162] E. Bergeron, M. Feeley, M.-A. Daigneault, and J. P. David, “Using dynamic reconfiguration to implement high-resolution programmable delays on an FPGA,” in 2008 Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference, 2008, pp. 265–268.

3. FPGA-based TDC Development

In order to research and understand the numerous theoretical and experimental topics related with TDC, two architectures were selected for implementation from the architectures discussed on the previous Chapter. This had the objective of studying the best TDC design practices and identify architectures that could be potentially used in LiDAR sensor applications. In this chapter the process of designing, implementing and testing of the selected architectures is presented. The Chapter starts by introducing the FPGA platform used, a Zybo development board with a Zynq7000 System-on-Chip (SoC). Since one of the objectives of this Thesis is to achieve an architecture that can be seamless ported between platforms and technologies, only fully digital architectures were considered for implementation. Moreover, TDCs are intrinsically dependent on the hardware used. FPGA platforms have predefined resources which vary with the architecture and vendor. Characteristics such as power supply voltage are shared among the entire FPGA device, thus the typical ASIC architectures based on DLLs to shield the TDC against PVT variations were not considered since they cannot be implemented in FPGAs.

The target application, the LiDAR sensor, requires high resolution (<1 ns for a 15 cm depth resolution) and precision, and low area. The architecture reporting the best resolutions in FPGA are based on multiple TDL. In modern FPGA devices, the typical propagation delay of a LUT element is about 250 ps. Moreover, modern FPGA also have dedicated carry blocks (Carry4 cells) for high speed calculations, with propagation delays under 30 ps, that can also be used to implement the Basic Delay Block (step) of TDLs. Therefore, there is no need for multiple TDLs to achieve a resolution under the intrinsic propagation delay of the available cells. Furthermore, the possibility of using a single TDL will help to meet the low area requirement. Regarding the high precision requirement, TDLs are known for having low linearity which decreases the overall TDC precision (without calibration), especially in FPGA platforms. As discussed in the previous Chapter, calibration techniques like decimation and bin-by-bin calibration can be implement in FPGA with low hardware resources. When developing for ASIC implementation, bin-by-bin calibration can also be used as an option for post-measurement calibration, however it will occupy a considerable chip area. Nevertheless, it can be implemented in software by the acquisition system if the output of the TDC is given in absolute counts. Moreover, in ASIC technology the process mismatch between TDL's cells is not as critical as with the FPGA-based TDL's Carry4 cells.

Decimation is also a solution to address the TDL non-linearity issue that can be implemented in ASIC with no extra hardware requirement, i.e., no impact on chip area, at the cost of resolution. TDLs can be implemented both in FPGA platforms and in ASIC technology and will have performances and resolutions that fulfil the requirements of the LiDAR sensor application. Consequently, the TDL architecture was selected for implementation since it can provide the required seamless portability.

With Flash LiDAR sensors applications in mind, another TDC solution based on Gray-Code counters was also explored, since the main advantage of this architecture targets is its low resources consumption. Furthermore, the architecture is able to satisfy the minimum required resolution and can also be implemented in ASIC technology.

3.1. The Zynq FPGA Platform

The Zynq-7000 all-programmable SoC is a development platform from Xilinx with an integrated Artix-7 based FPGA (Programmable Logic) and at least one Arm Cortex-A9 processors (Processing System) [3.1]. The ZYBO board, depicted in Figure 3.1, is one of the development boards equipped with this SoC. An overview of the SoC is depicted in Figure 3.2. The platform's SoC is fabricated in 28 nm technology and targets hardware and software co-design, while fastening development cycles and enabling the design complexity to be reduced. Furthermore, the platform has full design flow stack development support using the Vivado Design Suite framework, for PL development, and Xilinx SDK environment, to program the PS and integrate the developed hardware with the co-designed software. Moreover, Xilinx has AXI buses interfaces that can be instantiated, facilitating the integration of hardware and software components [3.2]. Several IPs are provided by Xilinx that contribute to accelerate the development process. However, apart from the basic structure of the AXI-Lite interface, no other IP was used during the development of the FPGA-based TDC prototypes.

To develop the FPGA-based TDC prototypes, the Zybo Z7-10 development board was used. The platform is equipped with a dual-core Cortex-A9 processor, at 667 MHz operating frequency, a 1 GB, 32-bit bus, DDR3L memory, at 1066 MHz, and 5 Pmod™ ports for expansion, from which eight are used for processor I/O [3.1]. Regarding the PL, the main difference between this FPGA and the Xilinx 7-Series Artix, are the dedicated ports and buses that connect the PL to the PS. All the PS peripheral controllers that are not connected to the Mux I/O can be routed through the PL to one of the Pmod ports using the Extended Mux I/O (EMIO) interface [3.2].

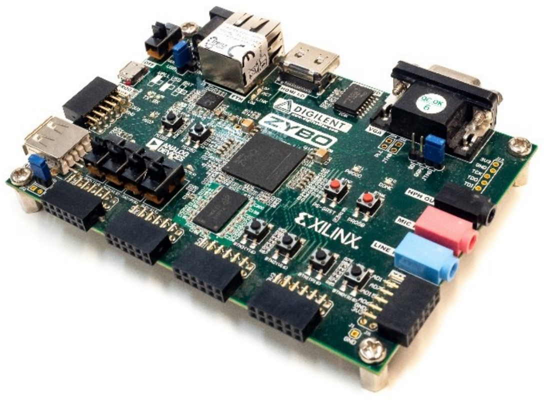


Figure 3.1- Zybo Development Board

3.1.1. Processing System (PS): The Cortex-A9

The Arm Cortex-A9 is a processor based on the Arm v7-A architecture that includes advanced Single Instruction Multiple Data (SIMD) and provides support for integer and floating-point vector operations [3.3]. It has support for full virtual memory implementation, due to its Level 1 cache subsystem, and has an instruction cache and branch prediction unit, to improve performance [3.3]. An overview of the main blocks composing the Cortex-A9 processor is depicted in Figure 3.2.

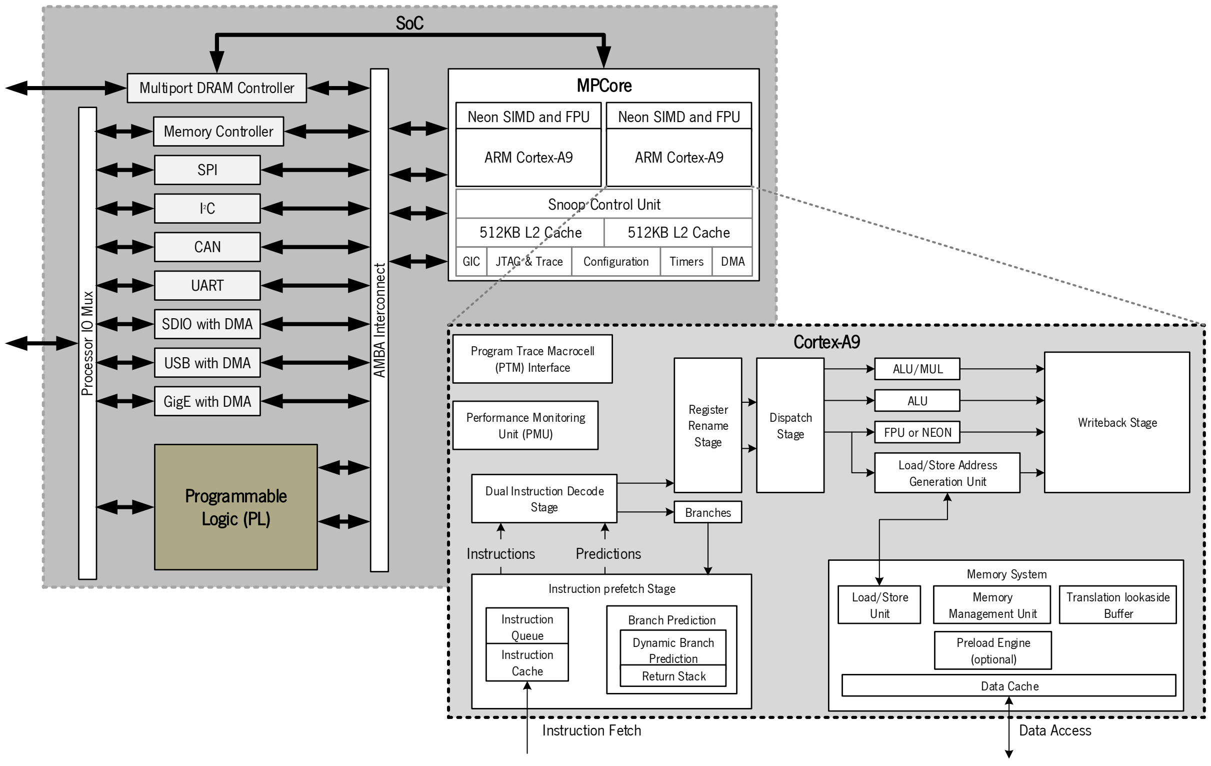


Figure 3.2 - Zynq SoC Overview

The internal communication is achieved through an AMBA 3 AXI interface bus. This bus is used to communicate with all the processor peripherals, like the memory controller units, communication peripherals (I2C, SPI, UART, USB, CAN, Ethernet), and general purpose I/O ports [3.1], [3.2], to name some. Furthermore, this bus is can also be used to implement the communication between the PS and the PL, enabling the processor to interface with new custom peripherals implemented in the PL, and/or to optimize some software tasks by implementing hardware acceleration functions in the PL [3.2].

3.1.2. Programmable Logic (PL)

Since the TDC implementation is closely coupled to the available technology, knowing how the FPGA blocks are organized and which elements implement those blocks is crucial to support future implementation decisions. In Xilinx 7-Series FPGAs, the main logic resource is a Configurable Logic Block (CLB) that is connected to a switching matrix, giving it access to the available FPGA's routing resources. The CLBs are organized in columns having two Slices each (see Figure 3.3). These Slices can be of type L or M (SLICEL and SLICEM, respectively), and a CLB can be composed of two SLICELs or one SLICEL and one SLICEM. The Slices are composed by four 6-input LUTs, eight storing elements (flip-flops – FF), wide-function multiplexers, and one Carry4 cell. The LUTs can be configured either as a 6-input LUT with one output, or as two 5-input LUTs with one output each. Furthermore, in the case of SLICEM, the LUTs can also be used to implement 32-bit distributed RAMs and shift registers. Four of the Slice's storing elements must be edge -triggered D-type flip-flops. The remaining four Slice storing elements can be configured as edge-triggered D-type flip-flops or latches, with a set or reset signal. However, if these storing elements are configured as latches, the first four flip-flops of the Slice cannot be used. Moreover, the storing elements must keep the same configuration inside the same Slice, i.e. if one of the storing elements is configured as a flip-flop with asynchronous reset, then the other storing elements can only implement the same type of flip-flop, since the control signals (clock, enable, set/reset) are shared inside a Slice.

The Carry4 cell is provided to enable fast arithmetic operations (addition and subtraction). Each CLB as two identical 4-bit carry chains, one per slice. In order to increase the number of inputs supported by the carry element, multiple carries from different slices can be cascaded using the COUT and CIN ports (See Figure 3.3). The CYINIT input is used as the CIN bit in the first carry of a carry chain or to select between the add operation (0) and the subtract operation (1). The carry element outputs the result of the addition/subtraction on O0 to O3 while the carry out of each bit can be accessed through CO0 to CO3 outputs, the last one, the most significant bit, is also connected to COUT to be used to cascade the carry chain.

3.2. TDC Design Flow and General Architecture

The first step when designing any digital system is to analyze the application requirements to understand which constraints and limitations must be addressed. According to the problem description presented in Chapter 1, a typical LiDAR sensor application requires resolutions below 7 cm and a range near 180 m. These requirements represent a time resolution for the ToF measurement better than 467 ps and a measurement range of approximately 1.34 µs. In order to address all the aforementioned constraints, the block diagram presented in Figure 3.4 was developed to guide the implementation of the TDC. The system's architecture was designed to guarantee a modular and flexible implementation, thus, all the blocks represented in Figure 3.4 can operate in standalone and be reused in other digital designs. This enabled the use of the same design structure to implement both FPGA-based TDC architectures, by only changing the fine measurement module.

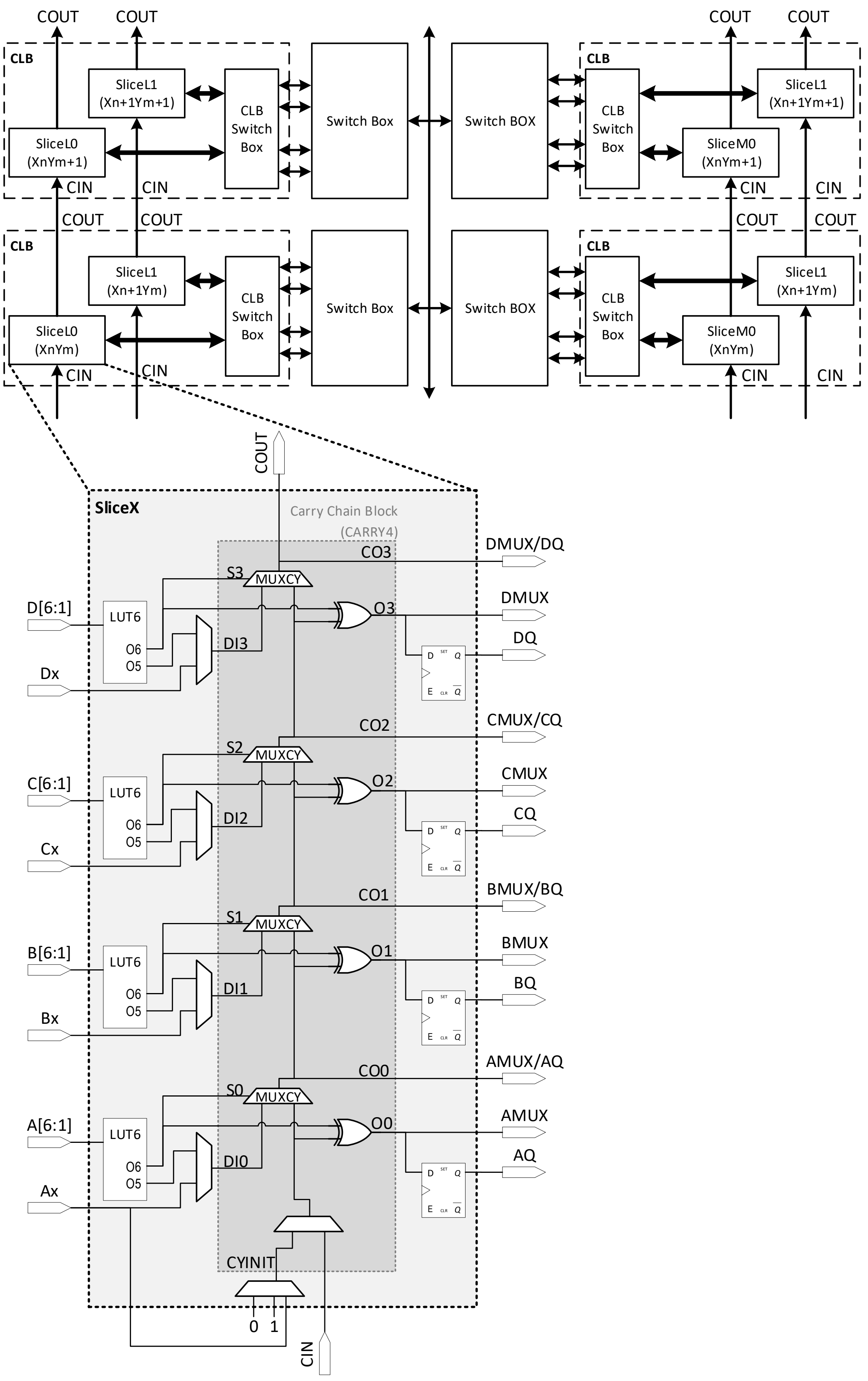


Figure 3.3- CLB disposition overview (top) and Slice detailed view (bottom)

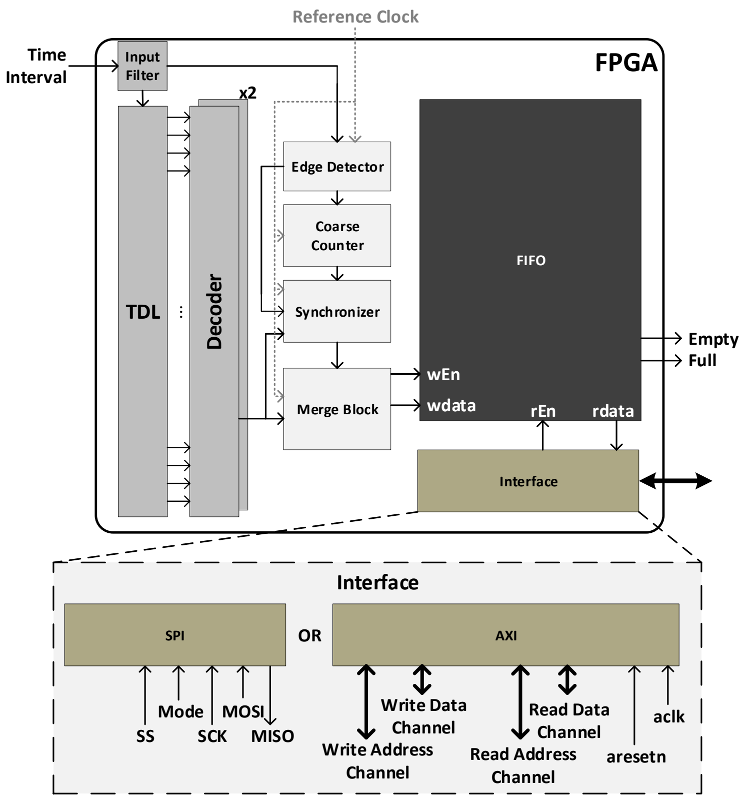


Figure 3.4- TDL TDC Architecture Overview

The proposed TDC architecture is composed by two measurement units, one for fine time counting, which enables higher resolution to be achieved, and the other for coarse time counting, which secures large dynamic range. Because the two measurement units operate asynchronously, a synchronization module is required to ensure that the values used from the fine and coarse measurement modules are correct. Two different synchronizers were developed during this Thesis. Further details on these modules and on its implementation will be presented in Section 3.3. A merge block was also implemented to combine the two measurement values and generate a set of control signals. The interface to the TDC is done through a storage module implemented using a FIFO (dual-port RAM) which is also used as the clock domain crossing mechanism between the TDC and the implemented interface. Two different interfaces were developed. One to interface the TDC with the PS Arm processor on the FPGA platform used, which is based on the AXI-Lite protocol. A SPI slave interface was also developed to enable the TDC system to be used by other external microcontrollers. The two communication protocols are mutually exclusive, either the TDC is implemented using the AXI-Lite or using the SPI slave interface.

With the general TDC architecture defined, it is now important to understand the multiple steps and resources involved in its implementation. These steps are closely coupled with the development tools used, in this case, the Vivado Design Suite framework. An overview of the design flow used and required resources can be seen in Figure 3.5. The digital flow adopted by Vivado has four main phases: System design entry, RTL Synthesis, Place & Route, and bitstream generation. System simulation can be done in-between each of the design steps.

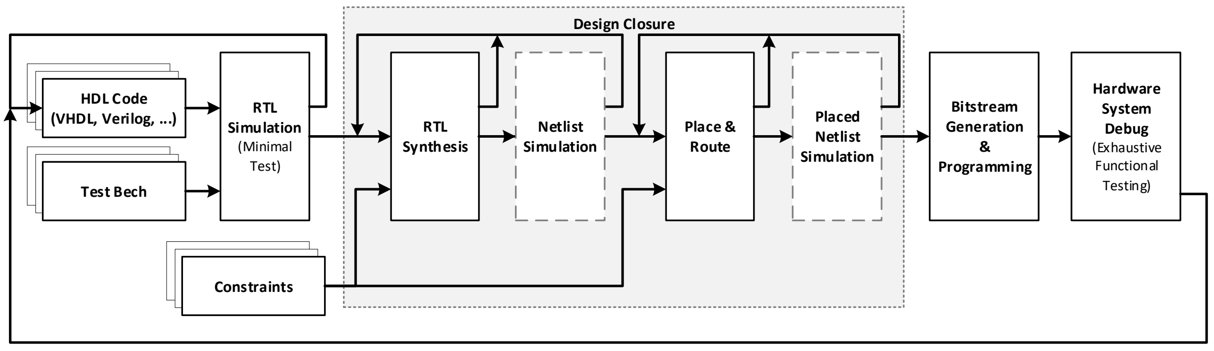

Design Closure
HDL Code (VHDL, Verilog, ...)
Test Bech
RTL Simulation (Minimal Test)
Constraints
RTL Synthesis
Netlist Simulation
Place & Route
Placed Netlist Simulation
Bitstream Generation & Programming
Hardware System Debug (Exhaustive Functional Testing)

Figure 3.5- FPGA Design Flow

The description of the system can be done using VHDL, Verilog or using both HDLs. In this Thesis Verilog was used to implement the digital design of the TDC. Once the description is completed, it is possible to generate an RTL (Register Transfer Level) view of the implemented system and perform the behavioral simulation. The generated RTL view represents a generic implementation of the described system. However, this is a technology independent view of the system and thus, the subsequent steps may introduce multiple changes due to technology mapping and optimization algorithms. Moreover, this step allows for non-synthesizable code to be used which, although helpful for debugging, must be used only for simulation purposes and never to describe a functionality that is intended to be implemented in the final system. Nevertheless, the generation of the RTL view can be considered equivalent to the analysis phase during synthesis in the ASIC design flow.

After, a behavioral simulation can be performed to assess our design and guarantee that it is functioning as intended. This is particularly useful to test the system's state-machines and sequential behavior. Vivado can be configured to directly interface different simulators. During this Thesis the default Vivado simulator was used.

Once the RTL is validated through behavioral simulation, the synthesis step is responsible to map the RTL code to the technology available on the selected platform. Since the Zybo Z7 board is mainly composed by LUTs, Storage elements and Carry arithmetic, all the combinatorial circuitry is converted to a logic expression and implemented using LUTs. Analogously, all registers triggered by a clock are mapped to storing elements. If any non-synthesizable code exists in the hardware description, a warning or error will occur stopping the synthesis process. At the end of the synthesis step a new RTL description is obtained.

This RTL description is no longer technology independent. Thus, functional and timing simulations are now possible. However, timing simulations prior the implementation step do not have information regarding placement and routing of the logic elements used. Therefore, these simulations only take into account the propagation delays on sequential storing elements and consider the propagation delay throughout combinatorial logic as ideal. Nevertheless, functional simulation in this step is important to guarantee that the optimization and technology mapping process did not change the intended behavior of the digital design.

With the design synthesized and its functional behavior validated, the cells instantiated by the technology dependent RTL must be placed and routed. This step is done during the Place & Route phase (also known as implementation). Timing constraints are important throughout the entire design flow since the tool's optimization algorithms make use of them to decide if a part of the design should be replicated or if buffers should be added to a given output. However, in the implementation phase, these constraints are very important, since these are the major drivers when deciding the optimal spot to place a logic element and which routing box and path must be selected. The lack of timing constraints may lead to design malfunction, solely due to arbitrary placement and routing. Upon completion of the implementation phase, functional and timing simulation of the placed and routed digital design can be performed.

The main difference between the post-synthesis timing simulation and the post-implementation timing simulation is that, in the later, the propagation delay of the combinatorial circuits and the routing delays are also considered during simulation. This timing information is represented and described in a Standard Delay Format (SDF) file that is generated by Vivado during the implementation phase. All timing simulations are made considering the worst-case timing scenarios by default. Apart from timing constraints, physical constraints must also be defined to map the input and output ports of the design to the FPGA physical locations. At the end of the implementation phase, power, timing and resources utilization reports are made available to further analyze the final design result.

The last step on the FPGA-based digital design flow is the generation of the bitstream used to program the FPGA. During this step a set of design rules, Design Rules Check (DRC), are made to validate the design. The output of this step is a *.bit* file that is used to configure the FPGA device according to the developed design.

Vivado also offers an IP integrator tool that enables multiple IPs to be instantiated, connected and validate. This functionality is especially useful to integrate the created designs with the processing system with minimal effort, since all the intermediary modules required are automatically generated by the framework. To make use of this functionality, the design must be encapsulated. More details regarding the created IP from the implemented design and the use of the IP integrator functionality will be presented in Section 3.3.4.

3.3. TDL TDC

The TDL was one of the selected architectures to be implemented in FPGA due to its intrinsic digital nature, attractiveness for attaining a full autonomous migration for ASIC platforms in a later stage of this Thesis, and achievable high resolutions, due to the fast carry blocks available on Xilinx FPGAs. Although being a pure digital system, the design of a TDC demands for additional steps and considerations during implementation, mainly to avoid unwanted optimizations, automatically done by the framework in the various steps of the design flow. This section will describe in detail the design and implementation of the multiple modules presented on Figure 3.4.

3.3.1. Architecture Design

The main block of the TDC system is the fine measurement module, since the major performance metrics of the TDC are defined or highly influenced by it. It is also the module that distinguishes the design from any other digital system design. Although a TDL is used as the base architecture for the fine measurement module, some changes were made to the typical approach in order to minimize resources utilization. The applications that this Thesis targets has both start and stop signals asynchronous to the reference clock. Usually, in such scenarios, two fine TDL measurement channels are implemented, one for measuring the start signal arrival time and another for the stop signal. In this Thesis, a fine measurement module designed with a single TDL for capturing both start and stop time interval is proposed. Using this approach, only the second stage sampling block and the decoder block must be replicated. Figure 3.6 depicts an overview of the RTL of the implemented fine measurement module.

The decision of using a single TDL adds a timing constraint to the time interval to be measured. There must be a minimum time interval, equal to at least one reference clock cycle, between the start and stop

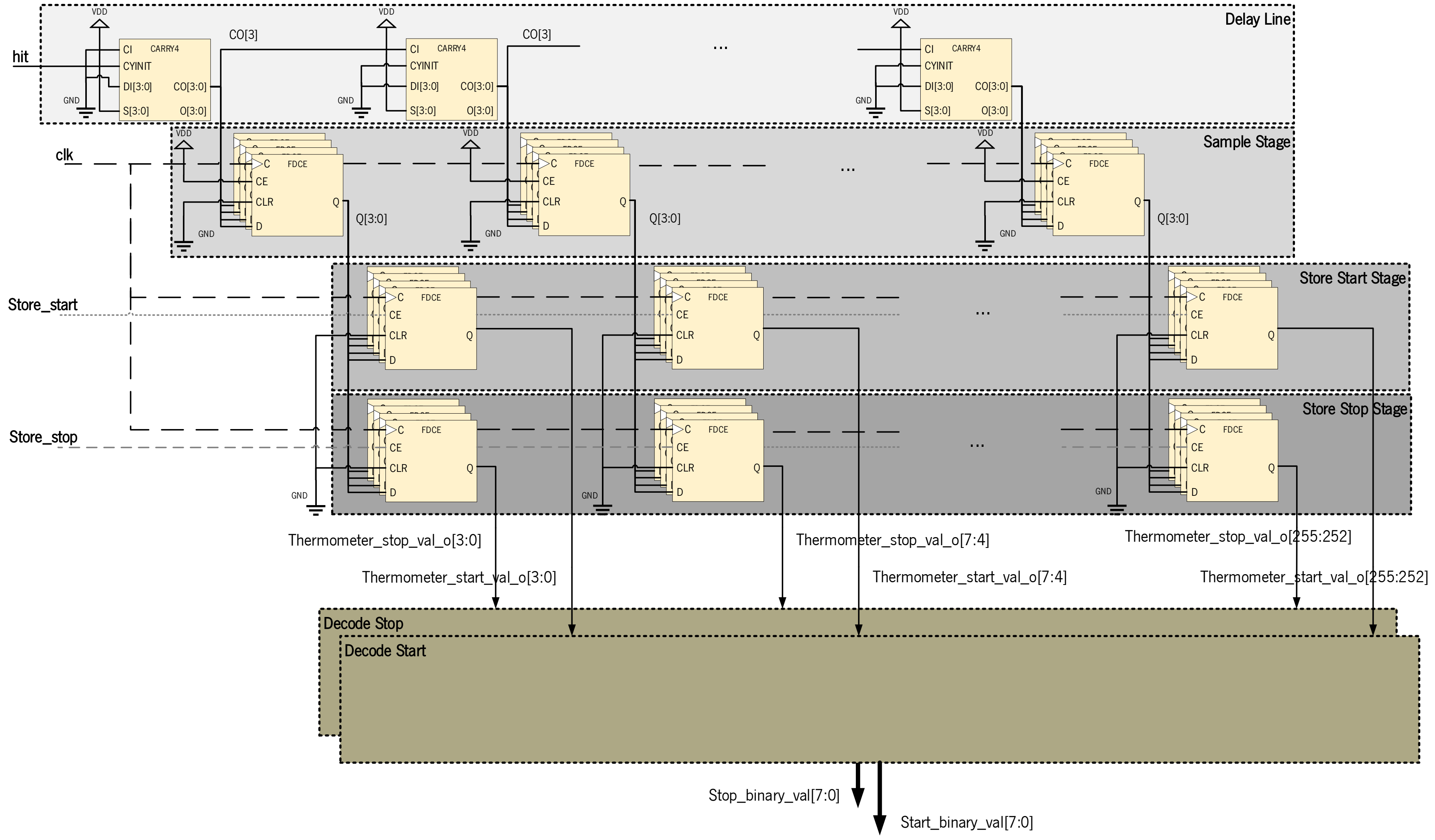


Figure 3.6- TDL RTL Overview

signal in order to properly capture both start and stop time of arrival. Otherwise, multiple transitions will appear in the sampled TDL thermometer code, jeopardizing the measurement result.

The main drawback of TDL architectures implemented in FPGA platforms is its poor linearity, due to process variation, which results in multiple steps having zero propagation delay, as mentioned before, this linearity issue is usually solved using decimation or bin-by-bin calibration. As already explained in Chapter 2, decimation sacrifices the TDC maximum achievable resolution, thus bin-by-bin calibration was the selected approach in this Thesis. This calibration technique can either be implemented directly in hardware or by software. Since one of the goals of this Thesis is to understand how to efficiently port a TDC design from a prototype platform to ASIC, and because the calibration tables required to implement bin-by-bin calibration require large chip areas to implement in ASIC platforms, it was decided that any sort of post-measurement calibration would be done by software and not directly implemented on hardware. Thus, bin-by-bin calibration tables, for the start and stop signals propagation delays, was built in software using the results obtained from a code density test with 100 thousand samples. The values received from the TDC are then used as index to address the calibration tables and the values returned are used in the final ToF measurement calculation.

The principle of operation of the proposed TDC is as follows: the arrival of the start signal generates a rising edge that starts propagating throughout the delay chain, creating a 1-to-0 pattern on the last propagated step; on the following reference clock rise edge, the first sampling stage stores the state of the delay chain. Simultaneously, an edge detector module generates a start signal event. The first sampling stage is always enabled, updating the TDL state at each reference clock cycle; in order to secure a stabilized value for the decoding stage, a second sampling stage, enabled by the start signal event during one clock cycle, is also implemented. Furthermore, this double sampling method reduces the probability of metastability. A second sampling stage for the stop signal was implemented in an analogous way; two reference clock cycles after a start or stop signal, the second sampling stage has a stable value that can be used by the decoder to obtain the equivalent binary state of the TDL from the sampled thermometer code; the decoder module is a purely combinatorial priority encoder that converts the thermometer code sampled from the TDL to a binary value. This value corresponds to the position of the last step of the delay chain at logic level ‘1’. This approach has good performance and shields the TDC against bubbles since only the last ‘1’-to’0’ transition is considered. Further details regarding the implementation process to reduce bubble occurrence are discussed in Section 3.3.2; the start signal event generated by the edge detector module also enables the coarse counter module, which starts incrementing at every reference clock cycle until the stop signal event is generated, disabling it and storing the value in a second set of registers; the process of obtaining the stop time interval is analogous to the start, but with the detection of a '0'-to-'1' pattern being propagated in the delay chain. Because the pattern to search is different, the decoder module must be different to the one used for the start signal. A waveform diagram exemplifying a typical measurement is shown in Figure 3.7.

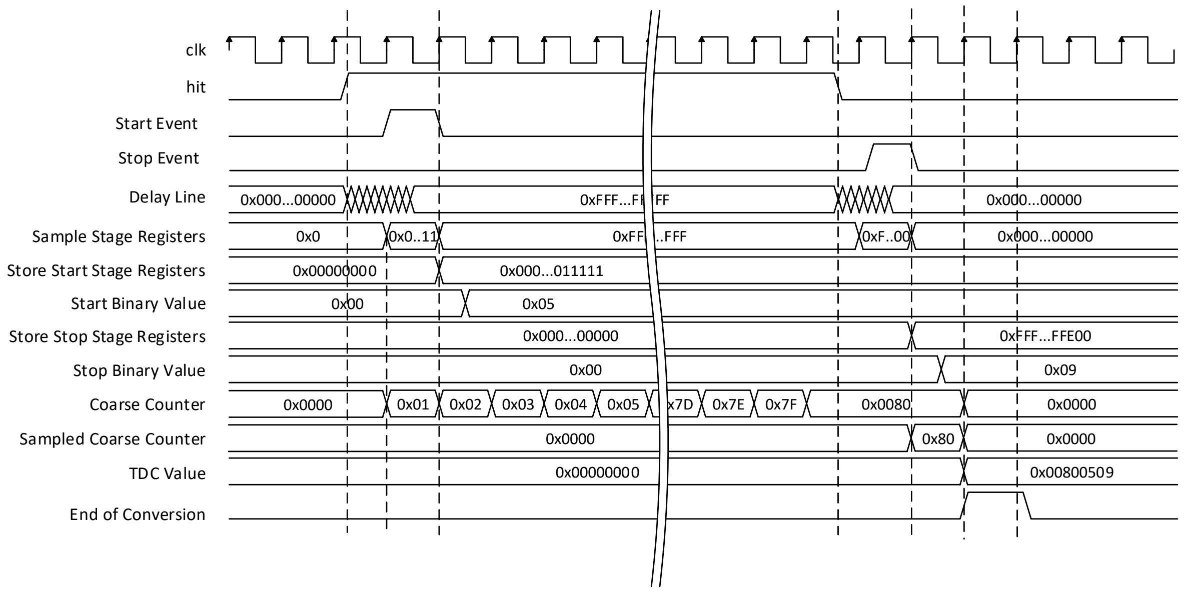


Figure 3.7- Typical TDL TDC Operation Waveform

The time interval to be measured, hereafter denoted as hit signal, is not directly connected to the TDL. Instead, an input stage is implemented to guarantee that no time interval measurement is accepted until the conversion of the previous one is finished. After reset, the input stage follows the hit signal. When a hit rise edge is detected, the input stage propagates it to the TDL and waits a falling edge transition of the hit signal. This event signalizes the stop signal.

Upon the arrival of the stop signal, the input stage keeps the signal propagating to the TDL at a low logic level until receiving an end-of-conversion (EOF) signal, generated by the merge block. This EOF signal indicates that the time interval has successfully been measured and stored in the FIFO memory. To avoid situations where the hit signal might already be at a high logic level when the EOF signal is generated, the input stage checks if the input hit signal is at '0'. If this is the case, the input stage starts bypassing the hit signal to the TDL, otherwise, the input stage waits for the hit signal to return to '0', while keeping '0' at the input of the TDL. Only after does the circuit bypass the hit signal to the TDL again.

Figure 3.8 depicts the RTL view of the input stage and the waveform diagram demonstrating its normal operation and the scenario in which partial time measurement could have been done if the hit signal check was not performed.

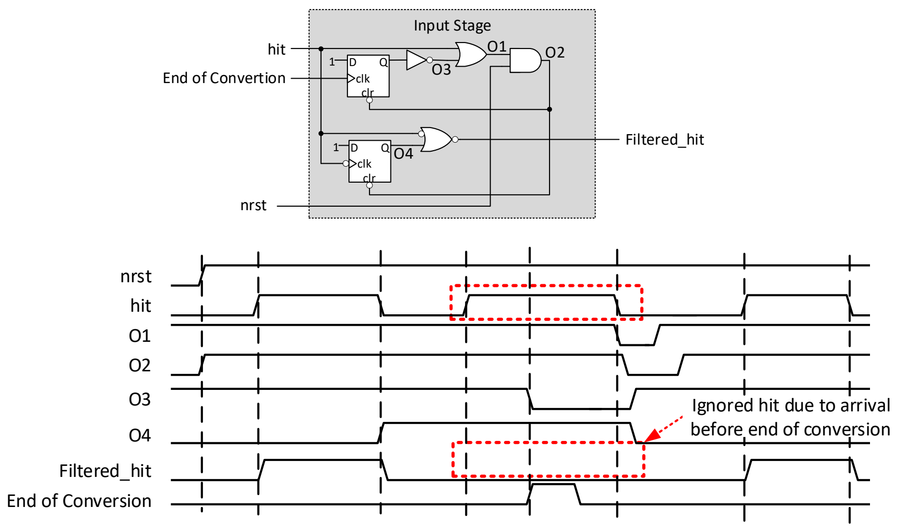


Figure 3.8- Input Stage Schematic (top) and Operation Waveforms (bottom)

The input stage also increases the flexibility of the TDL stage. Throughout the design of the TDC, it was assumed that the time interval to measure (hit signal) would have a pulse shape, being the rising edge of the hit signal used as the start event and the falling edge as the stop event. However, multiple applications require the generation of a pulse for the start event and another pulse for the stop event, being the relevant information the time between pulses and not the pulse's duration. To target such applications while using the same architecture, it is enough to add a D-type flip-flop with asynchronous reset at the beginning of the input stage, maintaining the remaining modules unaltered. The start event could be used to set the flip-flop while the stop event would be responsible for resetting it. Thus, a pulse would be generated with a width proportional to the time interval between the start and stop pulses. This would introduce an error on the measurement due to different propagation of the clock and reset signals on the flip-flop. However, this would be just an offset error correctable by software.

The coarse counter module is a 16-bit binary counter with enable, which is incremented by one at every reference clock cycle. The module has a second set of registers to store the counting value when a stop event is detected.

The synchronizer block is mandatory to guarantee the proper operation between the asynchronous and synchronous part of the TDC system. As already mentioned, since the hit signal is asynchronous to the reference clock, scenarios where the time of arrival of the hit signal violates the setup or hold time of the storing elements of the TDC will occur (see Figure 3.9). In such cases, the state of the TDL and coarse counter may differ (due to metastability on the counting registers or just incorrect counting and sampling, as a result of delay mismatch on the signals' routing). Metastability on the coarse counting registers usually results in measurement errors (several nanoseconds) multiple of the reference clock period. However, these types of errors are not frequent. The most frequent errors occur due to the different insertion delay of the hit signal's routing to the delay line and the enable pin of the counting registers, resulting in a measurement error equal to ±1 period of the reference clock.

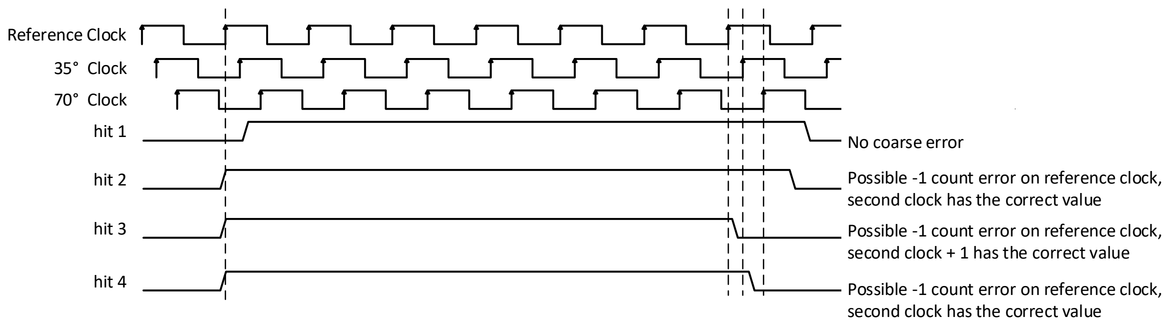

Reference Clock
35° Clock
70° Clock
hit 1
hit 2
hit 3
hit 4
No coarse error
Possible -1 count error on reference clock, second clock has the correct value
Possible -1 count error on reference clock, second clock + 1 has the correct value
Possible -1 count error on reference clock, second clock has the correct value

Figure 3.9- Synchronization Error Scenarios

To correct the coarse measurement, according to the value sampled by the TDL, a synchronization block with two extra coarse counters and a decider block were designed. The coarse counters are clocked by a PLL that outputs two clock signals with the same frequency of the reference clock but that are shifted in phase. This phase shift ensures that at least one counter has a stable counting at the arrival of the hit signal. Usually, this type of synchronization is implemented using a 180° phased clock (because only one signal is asynchronous to the system and thus there are only two possible synchronization error scenarios). However, since in the designed system both the start and stop signals are asynchronous to the system, there are six possible synchronization error scenarios. On Figure 3.9, only the start metastability scenarios are depicted for simplicity (for the stop signal, the scenarios are a mirror of the presented ones). So, to correctly identify every possible synchronization error, two extra counters are required.

The PLL was configured to output two clocks with a 35° and 70° phase difference regarding the reference clock. The start and stop values sampled by the TDL are used as reference to identify the moment of arrival of the hit signal regarding the reference clock. Depending on that value, the correct coarse counter is selected to be further used by the Merge module. The block diagram of the implemented synchronizer is presented in Figure 3.10.

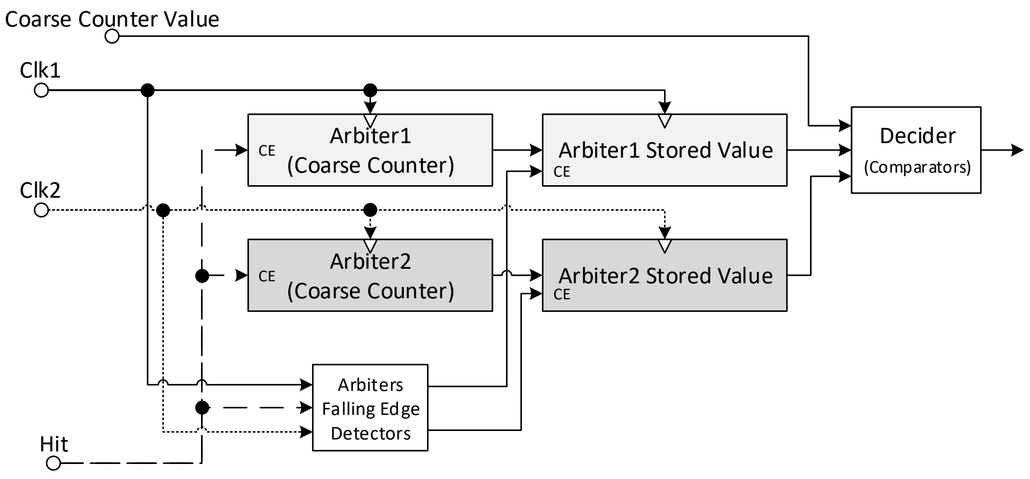


Figure 3.10- First Version of the Synchronizer Block

Later during this Thesis research, a second version of the synchronizer block was developed, aiming to reduce the number of resources and help structure the design process of the synchronizer. The main issue found with the first version of the synchronizer was its dependency on the placement and routing of the design, which demanded for experimental measurements to be done in order to understand if the synchronization boundaries were correctly defined. In order to systematize the design process, a design flow was created to assure a correct definition of the synchronization window (see Figure 3.11). Furthermore, if the coarse counter schema using the hit signal as the counting register's enable is switched to a free-running counting schema, using the hit signal as the second stage coarse registers sampling signal, then only one extra counter is required to identify all possible synchronization errors.

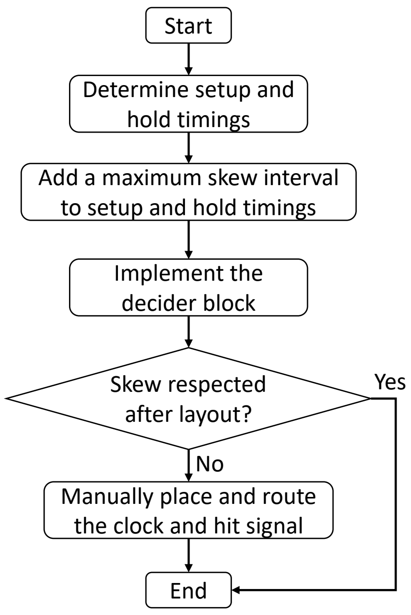

Start
Determine setup and hold timings
Add a maximum skew interval to setup and hold timings
Implement the decider block
Skew respected after layout?
Yes
No
Manually place and route the clock and hit signal
End

Figure 3.11– Design flow for proper synchronization window definition

In terms of storing elements, the number of resources used is the same since this approach required two coarse second stage sampling units (one to store the coarse value at the start event and another at the stop event). However, because only an extra counter is needed a 180° phase can be used, eliminating the need for a PLL.

Two major constraints must be assured independently of the synchronization mechanism used: the hit signal's skew and the clock signal skew between the coarse and fine measurement blocks must be as small as possible. Because multiple bits are being sampled in the coarse counter, the skew between the counting registers and the storing registers should be matched. The process of defining the synchronization windows is as follows: first, the limits must be defined with a value larger than the setup and hold time of the storing elements, being the lower and upper limits tied to the TDL values sampled moments before and after the reference clock signal rising edge respectively; after, a timing margin must be added to these limits to consider the signal skew when routing the hit and clock signals to the different systems insertion points. This margin will define the maximum allowable skew when placing and routing the design. The TDL values defined for the upper and lower limits of the synchronization window are used on the decider block to determine which coarse counter to use. Thus, in a normal TDC operation, if the value sampled by the TDL is inside the synchronization window, the 180° phased coarse counter sampled value is used. Otherwise, the value sampled from the main coarse counter is used. The block diagram of the decider module used in the second version of the synchronizer is depicted in Figure 3.12 (top), while the synchronization window process definition is depicted at the bottom of Figure 3.12 . When using the second version of the synchronizer, errors equal to ±1 reference clock period may occur in scenarios where one of the coarse values used is from the main counter and the other one is from the 180° phased counter. Those cases are identified in Table 3.1 with the respective correction factors that should be applied.

The merge module is responsible for combining the fine and coarse measurement values into a 32-bit word, and to generate the control signals for controlling the TDC state and to write to the FIFO memory. As already mentioned, the thermometer-to-binary decoding stage is a pure combinatorial module. The size of the module is proportional to the number of bits of the thermometer code to convert. When large thermometer codes are used, the module will be composed of multiple combinatorial levels, increasing the time required to perform the decoding. Thus, to prevent for scenarios where the time needed to correctly decode the thermometer code is longer than the reference clock period, the merge block implements a configurable cycle counter, to wait until the binary values outputted by the decoding stages are stable. Only when this counter expires the values are merged, setting a write enable flag to signalize the FIFO that a new value is ready to be stored. The write enable flag is also used as the end-of-conversion flag, to reset the input stage, making the TDC ready to perform a new measurement.

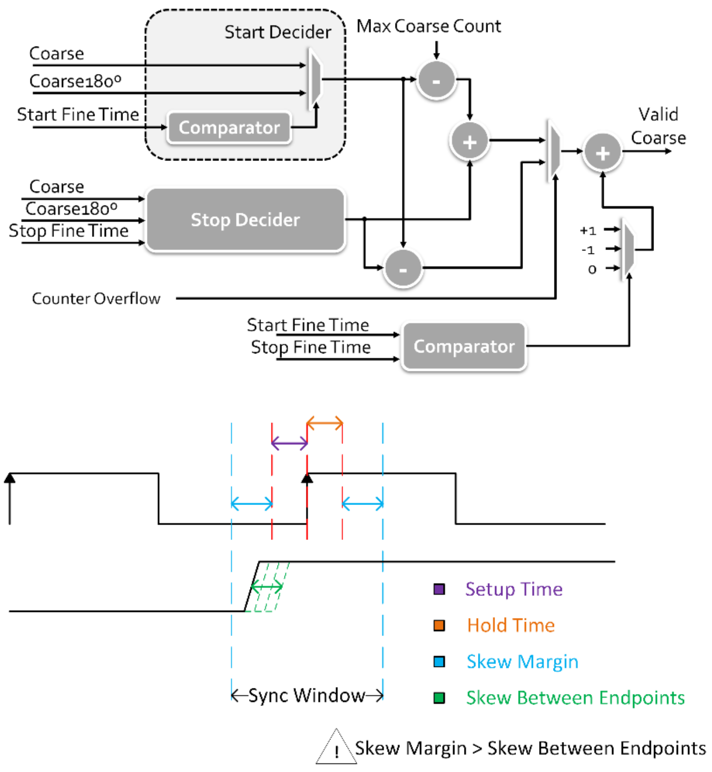


Figure 3.12- Second Version of the Synchronizer and Synchronization Window

Table 3.1- Synchronizer Correction Factors

Condition	*Correction Factor*
TDL Start Code > Sync Window Upper Value & TDL Stop Code < Sync Window Upper Value	-1 Coarse Count
TDL Start Code < Sync Window Upper Value & TDL Stop Code > Sync Window Upper Value	+1 Coarse count

3.3.2. Implementation Notes

The implementation of a TDC using a full HDL approach demands for additional Verilog constructs to be defined in order to control how the tool processes the files. The first thing to consider are the optimizations done by the tools upon synthesis. It is mandatory to mark the TDL components with the *dont_touch* keyword as illustrated in Figure 3.13. Otherwise, since the input of the delay line is always equal to the output, the synthesis will omit the delay chain, which in turn will also omit all sampling registers, rendering the TDL module useless. It was decided that both the TDL and the first and second sampling stages would be directly instantiated using the FPGA's platform available cells. However, the sampling stages could have been implemented using the traditional Verilog *always* construct. The remainder modules of the TDC do not required special considerations and can be optimized by the tools if necessary. To maintain the modular structure of the TDC, the *keep_hierarchy* attribute was defined at every module instantiation on the top module of the TDC. This indicates the synthesis tool that any optimization process must be done inside the boundaries of the module, prohibiting different modules to be merged during optimization. Although preventing some inter modules optimizations, which may lead to area optimized designs, this approach enables to keep a modular architecture, guaranteeing that the replacement of a module in the design will not affect the remaining ones, which is a much-appreciated characteristic during prototyping phase.

```
genvar i;
generate
    for(i=0; i < `NUM_STAGES/4; i=i+1)
    begin : delay_cell
        if(i==0)
        begin : delay_cell
        (* dont_touch = "TRUE" *) CARRY4 delay_cell(
            .CO(tdl_val_w[3:0]),
            .CI(1'b0),
            .CYINIT(hit_i),
            .DI(4'b0000),
            .S(4'b1111),
            .O()
            //.Z(tdl_val_w[i]),
            //.I(hit_i)
        );
        end
        else
        begin : delay_cell
        (* dont_touch = "TRUE" *) CARRY4 delay_cell(
            .CO(tdl_val_w[4*(i+1)-1:4*i]),
            .CI(tdl_val_w[4*i-1]),
            .CYINIT(1'b0),
            .DI(4'b0000),
            .S(4'b1111),
            .O()
            //.Z(tdl_val_w[i]),
            //.I(tdl_val_w[i-1])
        );
        end
    end
endgenerate
```

Figure 3.13- TDL Generation

Another important parameter to consider when implementing TDL is the real propagation delay of the basic delay cell used. To function properly, the TDL must have a length capable of surpassing the period of the reference clock in any scenario (best-, typical- or worst-case execution). The simulation tools only give information regarding the worst-case scenario. There is no information regarding the propagation delay of the FPGA elements on Xilinx's documentation. Thus, in order to determine the typical-case propagation delay, an experiment was made. The test consisted on creating a TDL with the maximum possible size allowed, which is only limited by the number of rows available on the FPGA device (one hundred in the case of the Zybo Z7-10). Since every row has a CLB with a Carry4 element in it (which has 4 carry elements), a four hundred steps (one carry element per step) TDL was built. The TDL was sampled by a set of registers which was storing the sampled thermometer values at every reference clock cycle to a FIFO memory. The integrated Arm processor was used, with an AXI interface to read the FIFO memory at the maximum allowable rate, and display the hexadecimal code read. Since the AXI-Lite is a 32-bit interface, 13 reads were made per measurement to obtain the full 400-bit thermometer code. With the test system implemented, one of the FPGA's PLL blocks was configured to output a 50% duty-cycle square-wave at a frequency doubling the one used in the reference clock. The output of the PLL was mapped to a FPGA's output pin and physically connected to the input pin of the TDL. With this configuration, it was possible to capture a full square-wave cycle within the sampled thermometer code. Knowing the number of cells needed to comprise a full PLL square-wave period, by dividing the period for this number, the average cell delay could be obtained. Accordingly, to the performed tests the average propagation delay of a carry element on the Zybo Z7-10 platform was 17.9 ps. In the timing simulation results, presented on Section 3.6.1 however, each carry element should have a propagation delay of 28.5 ps, which almost doubles the real value.

Considering the real average delay of 17.9 ps, in order to fully cover a reference clock with a 4 ns period (250 MHz), a TDL with a minimum of 224 steps must be implemented. Thus, a 256 step TDL was implemented. The extra steps were added to account for temperature variations and for faster Carry4 cells resultant from process variations.

3.3.3. Layout Considerations

While controlling synthesis is crucial in a full HDL TDC design, once the design is correctly implemented and mapped to the available technology in the FPGA, some additional constraints must be applied during placement and routing not only to guarantee proper TDC functioning, but also to improve the achievable performance.

The first layout consideration is regarding the placement of the TDL delay elements. Since the FPGA is divided into clock regions, and the routing of the clock has an impact on the performance of the delay line, the TDL must be placed inside a single clock region, if possible. Since 256 steps were implemented, the carry delay chain can only be propagated upwards, each CLB has four carry elements grouped up in a Carry4 block, and the Zybo Z7-10 platform has 50 rows per clock region, it is not possible to keep the TDL inside the same clock region.

For this reason, an ultra-wide bin is expected around the 200th step. This could be avoided if LUT elements were used to implement the TDL, since they do not have the restriction of only enable a propagating chain upwards. However, LUTs have higher propagation delay (approximately 123 ps in the Zybo Z7-10 according to the worst-case timing simulations), which would greatly reduce the TDCs resolution. Moreover, LUTs do not have a dedicated routing like the one connecting all the carry blocks in a column, instead LUTs use the route boxes. This would ultimately result in longer propagation delays due to longer routing paths and worse linearity performance as result of the non-uniform routing across the multiple TDL steps. Thus, having some ultra-wide bins was considered preferable, since the issue can be easily targeted by a calibration mechanism.

The routing between the output of the carry elements and the sampling stage registers is also an important factor to achieve higher linearity across the delay chain. While it is enough to constraint the propagation delay between the first and second sampling stages to one reference clock cycle, between the delay chain and the first sampling stage, the routing must be uniform across all steps and as small as possible. Therefore, the first sampling stage registers must be placed inside the same Slice as the carry elements they are sampling.

FPGAs have a highly optimized clock tree structure. Thus, inside the same clock region the routing skew has typical values under 30 ps. However, the same cannot be said when multiple clock regions are used since the insertion delays from the clock input pin to the distribution buffers of each region varies, thus increasing the skew between clock regions.

Controlling the clock signal in the TDC architecture presented is not as critical as controlling the hit signal. As mentioned during the presentation of the synchronization module, the hit signal insertion delay to the TDL and enable pins of the coarse counter registers must be closely match. Otherwise, the synchronization window will suffer a shift equal to the hit signal skew. The insertion delay of the hit signal can be controlled by adding LUTs configured as buffers to delay the signal on the fastest paths or by rerouting it.

3.3.4. Interface

To interface the TDC and store the measurement values, a FIFO module was implemented. Apart from performing a clock domain crossing mechanism, the FIFO module is also useful to temporarily store some measurement values when the processor's reading rate is lower than the acquisition rate of the TDC. The signals from the FIFO module were designed to enable the simple exchange of communication protocols, increasing the system modularity. In fact, the FIFO can also be addressed directly, reading the 32-bit output in parallel, with no communication protocol in-between the TDC and the processor reading from it.

The FIFO module is composed by a dual port RAM memory and two smaller modules to generate the write and read address pointers and the full and empty flags. Because the empty and full flags are generated by comparing the write and read pointers and these are generated in two different clock regions, a clock domain crossing module was implemented, using a double register method. Figure 3.14 depicts an overview of the FIFO implementation (top) and the write pointer and full flag module (bottom).

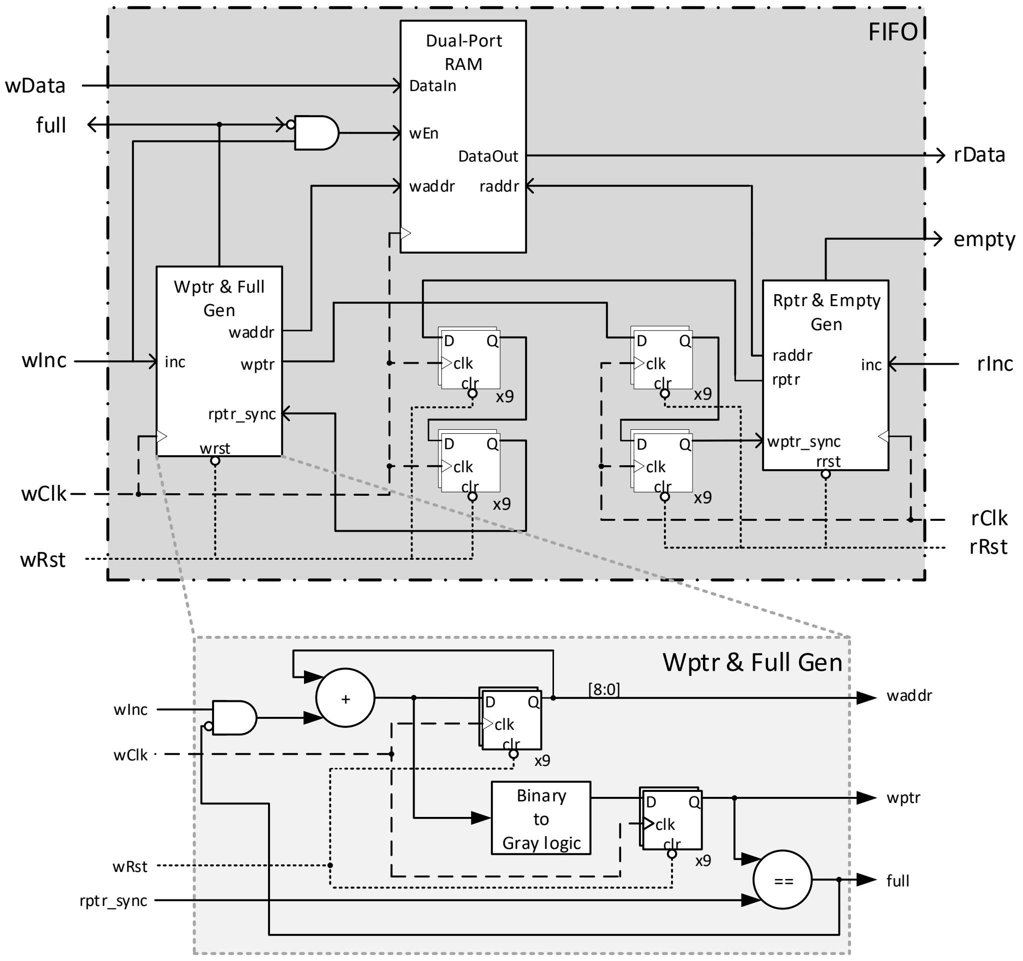


Figure 3.14- FIFO Module Overview and FIFO write pointer and Full Flag Generation module

The write and read address pointers generators are similar. The structure of the implemented write pointer generator is depicted at the bottom of Figure 3.14. The system is based on a (n+1)-bit binary counter, to address the 2^n FIFO memory positions. The extra bit is used to calculate the empty and full flags. If the *n* least significant bits of the read and write pointers are the same, then the logic XOR of the n+1 bit from those pointers define whether the FIFO is full or empty (empty when the MSB are the same and full when they differ). Before crossing the write pointer to the read clock domain, the value is converted from binary to Gray-code. The same is done when passing the read pointer to the write clock domain. This is done to avoid multiple bit state changes when the pointers' values are updated, increasing the system's robustness.

The FPGA Processing System's Arm Cortex-A9 uses an AXI bus to communicate with internally mapped peripherals. Using the Vivado framework it is possible to automatically generate a 32- or 64-bit AXI4 slave or master interface, with a default state-machine implemented, to encapsulate custom made IPs and automatically map them into the processor's peripheral address space. This functionality was used to automatically generate a 32-bit AXI-Lite slave interface. By default, four registers were instantiated in the AXI-Lite state machine to communicate with the processor. Only two of those registers are used, one to read the next value from the TDC's FIFO and another used by the processor to send commands to the implemented TDC IP. Since the AXI interface presented in the Arm Cortex-A9 processor is still a legacy AXI3 version, it was necessary to implement a bridge between it and the AXI4 interface encapsulating the TDC peripheral. This bridge may be generated by the Vivado framework or manually instantiated by the user when using the IP Integrator tool.

With the TDC IP implemented and mapped into the processor's memory, the last step was to develop a software application to read the values from the TDC peripheral. The Xilinx Software Development Kit (XSDK) is integrated in Vivado framework and enables the development of embedded application, grants access to the automatically generated Board Support Package (BSP) of the implemented system and provides a full debugging environment.

The algorithm for the software application to develop is as follows: first, the processor sends a read command to the TDC FIFO by writing to the memory position of register 1 of the TDC's AXI interface. Then the processor reads from the address of TDC's register 0, which has the updated value from the TDC FIFO. The 32-bit value received is then decoded to obtain the coarse, fine start and fine stop measurement values (see Figure 3.15 and Figure 3.16).

The TDC AXI peripheral was mapped in memory from address 0x43C00_0000 to 0x43C00_FFFF, being the base address represented by the macro *XPAR_TDC_0_S00_AXI_BASEADDR* in Figure 3.16. *TDC_S00_AXI_SLV_REG0_OFFSET* and *TDC_S00_AXI_SLV_REG1_OFFSET* are macros representing the offset address of the TDC AXI registers (0 in the case of register 0 and 1 for the register 1).

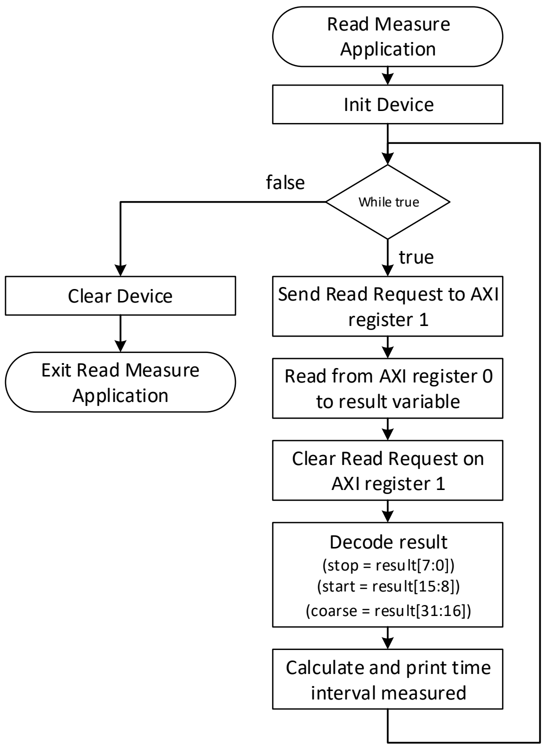

Read Measure Application
Init Device
While true
false
true
Clear Device
Exit Read Measure Application
Send Read Request to AXI register 1
Read from AXI register 0 to result variable
Clear Read Request on AXI register 1
Decode result
(stop = result[7:0])
(start = result[15:8])
(coarse = result[31:16])
Calculate and print time interval measured

Figure 3.15- TDC Read Application Flow

```c
#include <stdio.h>
#include "platform.h"
#include "xparameters.h"
#include "tdc.h"
#include "xil_io.h"
#include "xil_printf.h"
#define LSB 17.9
#define COARSE_PERIOD_NS 2500


int main()
{
    init_platform();
    uint32_t start = 0,stop = 0,coarse = 0;
    uint32_t result = 0;
    uint32_t measuredTime = 0;

    print("Press any key to strat the measurement\n\r");
    scanf("%d");
    for(;;)
    {
        //AXI READ
        TDC_mWriteReg(XPAR_TDC_0_S00_AXI_BASEADDR,TDC_S00_AXI_SLV_REG1_OFFSET,1);
        result = TDC_mReadReg(XPAR_TDC_0_S00_AXI_BASEADDR,TDC_S00_AXI_SLV_REG0_OFFSET);
        TDC_mWriteReg(XPAR_TDC_0_S00_AXI_BASEADDR,TDC_S00_AXI_SLV_REG1_OFFSET,0);
        //TDC VALUE DECODE
        start = (result&0xff00)>>8;
        stop = (result&0xff);
        coarse = (result&0xffff0000)>>16;
        //TDC VALUE CALCULATION
        measuredTime = (coarse*COARSE_PERIOD_NS)+(calibrationTable[start])-(calibrationTable[stop]);
        xil_printf("Measure\nDebug; %x\nCoarse; %d\nStart; %d\nStop; %d\nTime Interval; %d\n", result, coarse, start, stop, measuredTime);
    }
    cleanup_platform();
    return 0;
}
```

Figure 3.16- TDC Read Application

The coarse value was multiplied by the reference clock frequency and the values from the fine start and stop measurements are used to address the calibration table. The final measurement result can be obtained using equation (3.1):

$$t = coarse_{counts} * \frac{1}{T_{CLK}} + (start_{calibrated} - stop_{calibrated}), \qquad (3.1)$$

3.4. Gray-Code TDC

The Gray-code oscillator architecture was first proposed in [3.4]. The architecture reported a mean bin size of 256 ps and 271 ps, for the two TDC channels implemented, using 8-LUTs and 8 flip-flops per channel, being its main components a gray counter and a 2-bit oscillations counter.

This architecture was developed in order to achieve high resolution for a low-power and low-resource system. Typically, a counter is composed by a combinatorial stage, which calculates the next value in the counting schema, and by a sampling stage, responsible for latching the value of the counter at each clock cycle, assuring a stable value for the combinatorial stage, so that the next value can be correctly calculated and latched in the next clock cycle. This is of extreme importance for binary counters in which multiple bits can change from one counting value to the next, activating multiple combinatorial paths at the same time. For example, in a 4-bit value, during the increment from seven (0111) to eight (1000), all the bits change. If no latching stage is present, i.e. the output of the combinatorial stage is directly connected to its inputs, there is the risk of having random values, and therefore a random counting sequence, due to different propagation delays in the counter's datapath. However, if a Gray-code counting schema is used, this problem is avoided. In the original Gray-code, only one-bit changes from one state to the next one. Thus, the Gray-code can be configured in a loop, without the latching stage, since there is no risk of missing codes or making a random counting sequence. This enables the implementation of a counter with a resolution that is no longer limited by the system clock used to sample the state of the counter. Instead, the maximum achievable resolution is given by the propagation delay of the counter's signals trough the Datapath (cells' propagation delay plus routing delays).

The Gray-code counter implemented in [3.4] is based on the a 5-bit reflected binary code (RBC) schema. In such schema, the first bit of the Gray-code does not have a dependence on itself. As the Gray-code has only 5-bits, a single 5-input LUT is enough to calculate each bit next state. However, because the counter also needs a bit to enable the counter, 6-input LUTs are used to implement it. Apart from the Gray-code counter, a 3-bit cycle counter was also included, which counts the number of full counts performed by the Gray-code counter. This mechanism was implemented to extend the range of the TDC, enabling a lower clock frequency to be used in the final system. From the three bits, only two are used. The decision of which bits should be used is done depending on the output of the Gray-code TDC. This guarantees that the selected bits are not metastable.

The architecture was implemented in a Kintex-7, a Xilinx 28 nm technology FPGA. In this platform, as seen in Section 3.1, each CLB has two Slices with four 6-input LUTs, a 4-bit Carry block and eight flip-flops each. Therefore, a single TDC channel can be implemented in a single CLB, allowing for multiple channel implementation even on small FPGAs. The research reports step delays in-between 100 ps and 500 ps, which, for a mean step delay of 256 ps corresponds to a DNL in the range of -0.61 LSB to +0.95 LSB. To obtain these results, the building cells of the TDC were manually placed inside the same CLB, in a specific order. By the analysis of the steps' delays for the two TDC channels presented in [3.4], it is possible to notice that, for some steps, the same TDC step in different TDC channels has delay differences greater than 200 ps, leading to the conclusion that placement influences TDC channel performance. Finally, to improve linearity, a four measurement per input pulse, followed by averaging, is proposed in [3.4]. This method, although enabling better linearity with no extra resource usage, decreases the system throughput. The Gray-code architecture was studied during this Thesis research, since it could be interesting for Flash LiDAR applications due to its very low resource utilization. In the following sections, a proposal for improving the base Gray-code TDC architecture linearity and scalability based on controlling the routing propagation delay is presented.

3.4.1. Architecture Design

Based on the wire load regulation principle presented in [3.5], and the base TDC architecture presented by Wu and Xu in [3.4], an improved linearity Gray-code TDC architecture was designed during this Thesis research. Furthermore, the followed approach enabled the improvement of the TDC architecture scalability, by reducing the channels mismatch when implementing multiple TDC channels. Thus, improved performance is obtained reducing the need for calibration circuitry or post-measurement calibration software routines. The base block diagram of the proposed architecture is depicted in Figure 3.17, and Figure 3.18 present the logic equations to calculate each Gray-code bit. The core of the TDC channel, presented in Figure 3.19, is a pure combinatorial 5-bit Gray-code counter that, when enabled, starts looping through the 32 possible values. The Gray counter is enabled on the rise edge of the hit signal. To avoid that the counter continues to oscillate undefinably, the input stage presented in Figure 3.17 was used to guarantee that the enable for the Gray counter has a maximum duration of one reference clock cycle. After the arrival of a hit signal, on the next clock rise edge, the value of the Gray-code counter is sampled and simultaneously, *hit_r* (the output of the Input Stage register) is cleared thus stopping the Gray counter. When the value sampled is different from zero, a store signal is generated to sample the value of a free-running coarse counter, used to increase the measurement range of the TDC. The Gray-Code value sampled is also stored in a second set of registers on the second rising edge of the reference clock. Like in the TDL case, this is done to reduce the probability of metastability on the sampling of measurement value from the Gray counter and to secure a stable value for the next operations (the first set of registers samples the Gray-code at every clock period and therefore, if the value was not stored, the measurement value would be lost at the second rise edge of the clock after the hit's signal arrival).

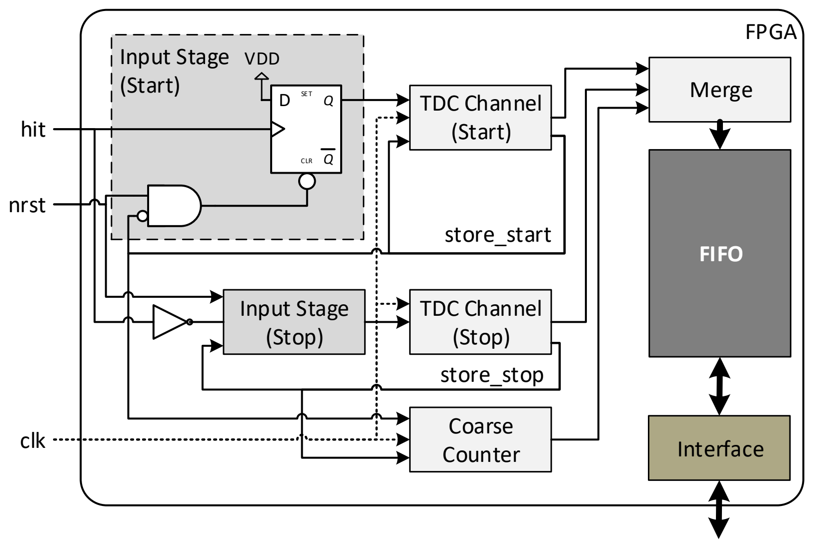


Figure 3.17- Gray-Code Architecture Overview

```
assign gray_w[4] = ((gray_w[4] & !(!gray_w[3] & !gray_w[2] & !gray_w[1] & !gray_w[0])) |
                    (!gray_w[4] & gray_w[3] & !gray_w[2] & !gray_w[1] & !gray_w[0])) & en_i;

assign gray_w[3] = ((!gray_w[4] & (gray_w[3] | (gray_w[2] & !gray_w[1] & !gray_w[0]))) |
                    (gray_w[4] & (gray_w[3] & (!gray_w[2] | gray_w[1] | gray_w[0])))) & en_i;

assign gray_w[2] = ((!gray_w[4] & ((!gray_w[3] & gray_w[1] & !gray_w[0]) | (gray_w[2] & (!gray_w[1] | gray_w[0])))) |
                    (gray_w[4] & ((gray_w[3] & gray_w[1] & !gray_w[0]) | (gray_w[2] & (!gray_w[1] | gray_w[0]))))) & en_i;

assign gray_w[1] = ((!gray_w[4] & ((gray_w[1] & !gray_w[0]) | (gray_w[0] & ((!gray_w[3] & !gray_w[2]) |
                    (gray_w[3] & gray_w[2]))))) | (gray_w[4] & ((gray_w[1] & !gray_w[0]) | (gray_w[0] &
                    ((!gray_w[3] & gray_w[2]) | (gray_w[3] & !gray_w[2])))))) & en_i;

assign gray_w[0] = ((!gray_w[4] & ((!gray_w[1] & ((!gray_w[3] & !gray_w[2]) | (gray_w[3] & gray_w[2]))) |
                    (gray_w[1] & ((!gray_w[3] & gray_w[2]) | (gray_w[3] & !gray_w[2]))))) | (gray_w[4] &
                    ((!gray_w[3] & ((!gray_w[2] & gray_w[1]) | (gray_w[2] & !gray_w[1]))) | (gray_w[3] &
                    ((gray_w[2] & gray_w[1]) | (!gray_w[2] & !gray_w[1])))))) & en_i;
```

Figure 3.18- 5-bit Gray-Code Logic Equations

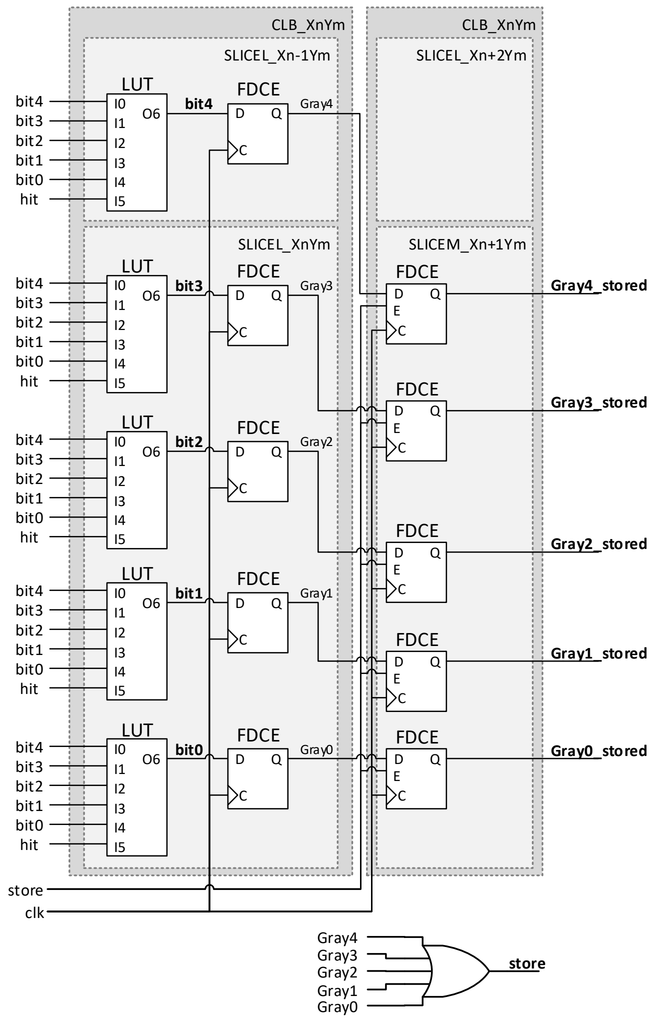


Figure 3.19- Gray-Code channel RTL View

A second TDC channel was implemented to measure the arrival time of the hit's signal falling edge. This way, it was possible to measure the time of an impulse. A waveform, with the typical TDC's channel signals, during a measurement procedure, is depicted in Figure 3.20, and in Figure 3.21 the state machine of a measurement process is presented. Differently to the architecture presented by Wu and Xu [3.4], no bit extension was used in this implementation of the TDC channel. Therefore, the maximum counting value, assuming the 250 ps average delay per stage reported for the 7-Series Xilinx FPGA, is 8 ns. This constrains the minimum clock of the system to 125 MHz, which, for modern FPGAs, is a constraint easy to meet. Although this decision forces the use of a faster clock, which can lead to higher power consumption, it also enables to save three LUTs, three flip-flops and one Carry4 element per TDC channel.

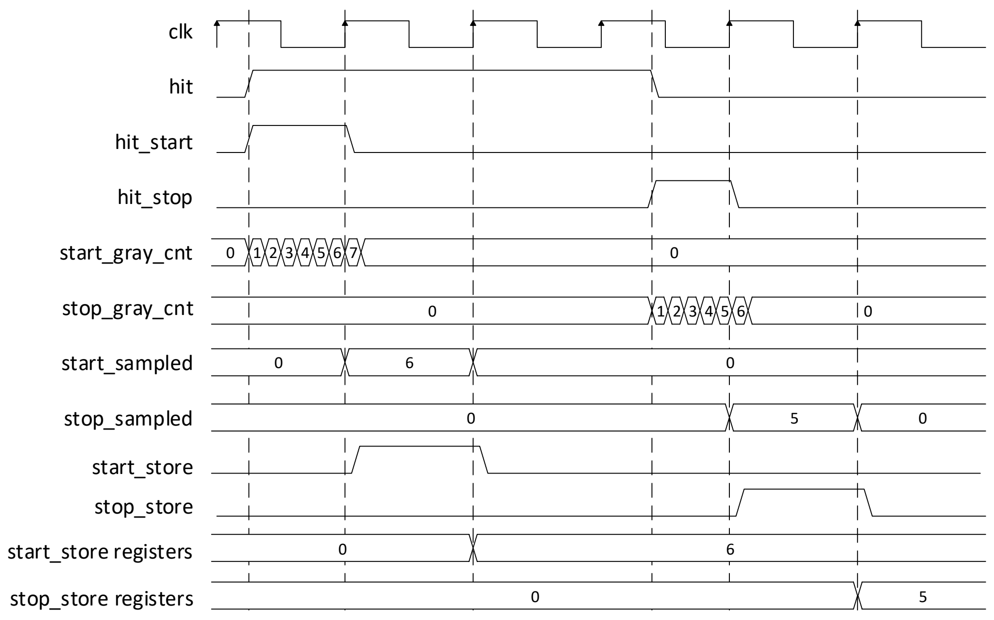
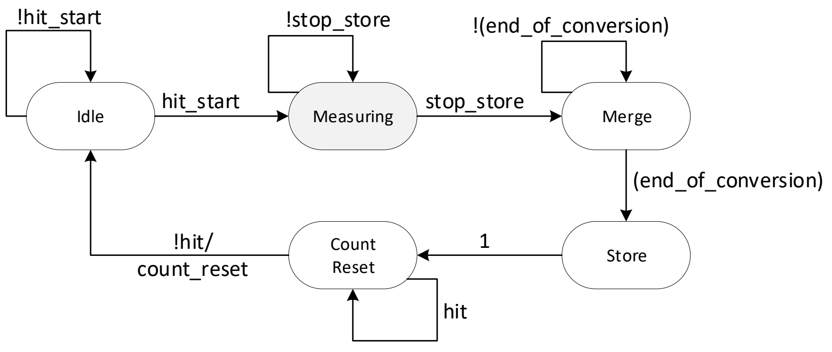


Figure 3.20- Gray-Code TDC Typical Measurement Waveform



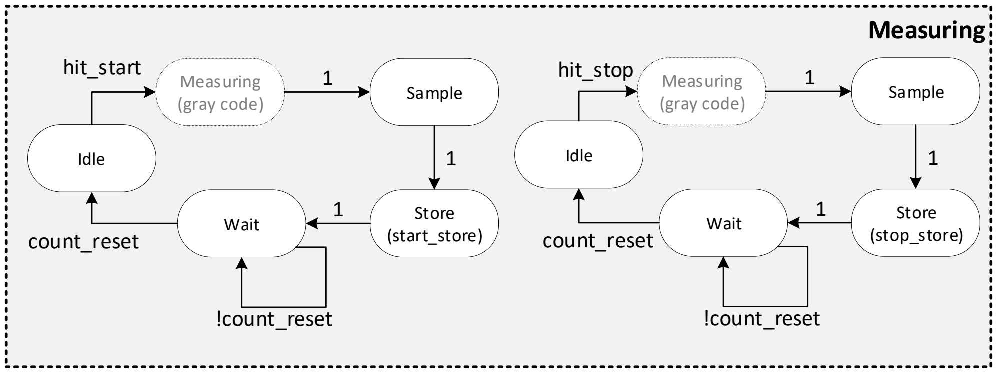


Figure 3.21- Gray-Code TDC State-Machine

Since all the modules developed for the TDL architecture were designed targeting portability and modularity, the new Gray-code TDC channel was integrated with the synchronizer, merge, FIFO and AXI-Lite Interface modules automatically. It was only necessary to substitute the TDL Channel module by the Gray-code TDC module.

3.4.2. Implementation notes and Layout Considerations

Gray's Counter oscillator Datapath Analysis

Based on the results reported in [3.5], the propagation delay of a cell in FPGA platforms can be partially controlled by controlling the load of the cell. This can be done by adding dummy buffers to the output of a cell, to increase its load, or by increasing/decreasing the size of the routing, which will increase/decrease the parasitic capacitance of the net, changing the load at the output of the cell. The second option is more suitable for smaller adjustments, either to increase or reduce the propagation delay, but demands for a great knowledge on the FPGA routing resources, making its implementation more complex. Furthermore, the propagation delays obtained from the tool, which are dependent on the wire loads, are always the worst-case scenario. Nevertheless, the second option does not require extra resource usage, apart from the routing resources which are required for both methods. By assuring a close match between the propagation delays in the worst-case scenario, it is possible to assume that the typical conditions would be similar as well. Moreover, as the TDC channel can be confined to a single CLB, the voltage and temperature conditions should be similar in all the five LUTs used to build the Gray-code oscillator. Therefore, exploring the routing possibilities during the layout of the Gray-code TDC may lead to an implementation with higher linearity with no extra hardware cost, improving the overall system's performance.

By analyzing the pattern on the 5-bit Gray-code it is possible to conclude that only 8 out of the 24 datapath connections affect the size of the steps. Namely the paths from the output of LUT0 to the input of all the other LUTs (four connections) and the output of all the other LUTs to the inputs of LUT0 (another four connections, see Table 3.2 and Figure 3.19). This greatly reduces the effort of implementing the linearity correction through routing. The remaining 16 datapath will not affect the delay of the steps (as long as the propagation delay of these paths do not exceed two clock cycles), and therefore can be automatically routed by the framework.

Manual Routing to Control Datapath's delay

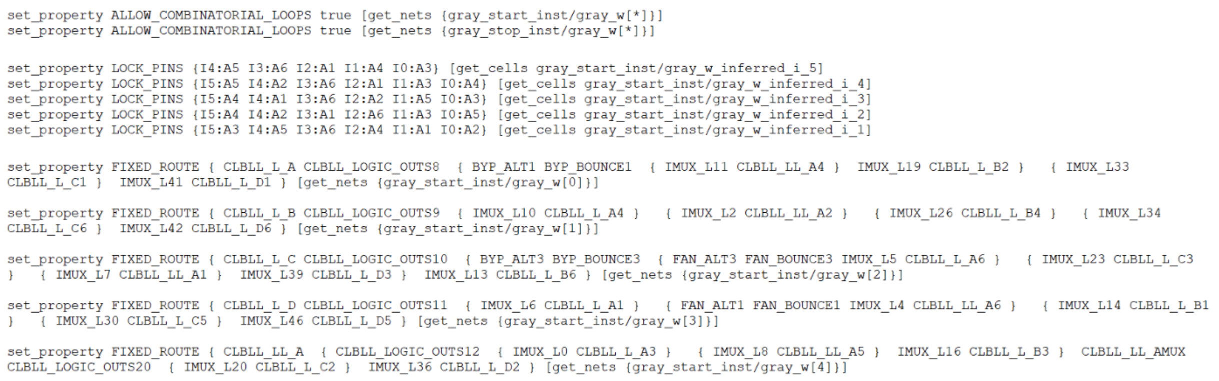
The manual routing process can be done in Vivado design tool using the implemented design graphical interface, or by creating a file with the set of physical constraints annotated in a Xilinx Design Constraints (XDC) format. If the graphical interface is used, the manual routing is done by entering the implementation design view and selecting the routing resources option of the Device tab. Then the routings on the nets

Table 3.2- Gray-Code Datapath Delay Analysis

Current Gray Value					**Next Gray Value**					**Propagation delay**
bit4	bit3	bit2	bit1	bit0	bit4*	bit3*	bit2*	bit1*	bit0*	
0	0	0	0	0	0	0	0	0	1	Tphit + tpLUT0
0	0	0	0	1	0	0	0	1	1	Tpbit0 + tpLUT1
0	0	0	1	1	0	0	0	1	0	Tpbit1 + tpLUT0
0	0	0	1	0	0	0	1	1	0	Tpbit0 + tpLUT2
0	0	1	1	0	0	0	1	1	1	Tpbit2 + tpLUT0
0	0	1	1	1	0	0	1	0	1	Tpbit0 + tpLUT1
0	0	1	0	1	0	0	1	0	0	Tpbit1 + tpLUT0
0	0	1	0	0	0	1	1	0	0	Tpbit0 + tpLUT3
0	1	1	0	0	0	1	1	0	1	Tpbit3 + tpLUT0
0	1	1	0	1	0	1	1	1	1	Tpbit0 + tpLUT1
0	1	1	1	1	0	1	1	1	0	Tpbit1 + tpLUT0
0	1	1	1	0	0	1	0	1	0	Tpbit0 + tpLUT2
0	1	0	1	0	0	1	0	1	1	Tpbit2 + tpLUT0
0	1	0	1	1	0	1	0	0	1	Tpbit0 + tpLUT1
0	1	0	0	1	0	1	0	0	0	Tpbit1 + tpLUT0
0	1	0	0	0	1	1	0	0	0	Tpbit0 + tpLUT4
1	1	0	0	0	1	1	0	0	1	Tpbit1 + tpLUT0
1	1	0	0	1	1	1	0	1	1	Tpbit0 + tpLUT1

of the TDC must be eliminated. This will enable to enter the assign routing mode option, an interface that guides the user through the different available resources to connect two endpoints. By using a trial and error process, different routing options can be explored to achieve the desired net delay.

In order to be able to fix the manual routing done on a pure combinatorial path, the LUTs need to be fixed with placement constraints and its inputs need to be locked. Otherwise the tool may change the order of the input ports from one run to another, leading to scenarios where it is impossible to respect the defined routing constraints [3.6], resulting in an implementation with unconnected nets. With the LUTs manually placed and its inputs locked, the manual routing can be safely defined. The mandatory constraints used in the proposed TDC channel implementation are presented in Figure 3.22.



```
set_property ALLOW_COMBINATORIAL_LOOPS true [get_nets {gray_start_inst/gray_w[*]}]
set_property ALLOW_COMBINATORIAL_LOOPS true [get_nets {gray_stop_inst/gray_w[*]}]

set_property LOCK_PINS {I4:A5 I3:A6 I2:A1 I1:A4 I0:A3} [get_cells gray_start_inst/gray_w_inferred_i_5]
set_property LOCK_PINS {I5:A5 I4:A2 I3:A6 I2:A1 I1:A3 I0:A4} [get_cells gray_start_inst/gray_w_inferred_i_4]
set_property LOCK_PINS {I5:A4 I4:A1 I3:A6 I2:A2 I1:A5 I0:A3} [get_cells gray_start_inst/gray_w_inferred_i_3]
set_property LOCK_PINS {I5:A4 I4:A2 I3:A1 I2:A6 I1:A3 I0:A5} [get_cells gray_start_inst/gray_w_inferred_i_2]
set_property LOCK_PINS {I5:A3 I4:A5 I3:A6 I2:A4 I1:A1 I0:A2} [get_cells gray_start_inst/gray_w_inferred_i_1]

set_property FIXED_ROUTE { CLBLL_L_A CLBLL_LOGIC_OUTS8  { BYP_ALT1 BYP_BOUNCE1  { IMUX_L11 CLBLL_LL_A4 }  IMUX_L19 CLBLL_L_B2 }   { IMUX_L33
CLBLL_L_C1 }  IMUX_L41 CLBLL_L_D1 } [get_nets {gray_start_inst/gray_w[0]}]

set_property FIXED_ROUTE { CLBLL_L_B CLBLL_LOGIC_OUTS9  { IMUX_L10 CLBLL_L_A4 }   { IMUX_L2 CLBLL_LL_A2 }   { IMUX_L26 CLBLL_L_B4 }   { IMUX_L34
CLBLL_L_C6 }  IMUX_L42 CLBLL_L_D6 } [get_nets {gray_start_inst/gray_w[1]}]

set_property FIXED_ROUTE { CLBLL_L_C CLBLL_LOGIC_OUTS10  { BYP_ALT3 BYP_BOUNCE3  { FAN_ALT3 FAN_BOUNCE3 IMUX_L5 CLBLL_L_A6 }   { IMUX_L23 CLBLL_L_C3
}   { IMUX_L7 CLBLL_LL_A1 }  IMUX_L39 CLBLL_L_D3 }  IMUX_L13 CLBLL_L_B6 } [get_nets {gray_start_inst/gray_w[2]}]

set_property FIXED_ROUTE { CLBLL_L_D CLBLL_LOGIC_OUTS11  { IMUX_L6 CLBLL_L_A1 }   { FAN_ALT1 FAN_BOUNCE1 IMUX_L4 CLBLL_LL_A6 }   { IMUX_L14 CLBLL_L_B1
}   { IMUX_L30 CLBLL_L_C5 }  IMUX_L46 CLBLL_L_D5 } [get_nets {gray_start_inst/gray_w[3]}]

set_property FIXED_ROUTE { CLBLL_LL_A  { CLBLL_LOGIC_OUTS12  { IMUX_L0 CLBLL_L_A3 }   { IMUX_L8 CLBLL_LL_A5 }  IMUX_L16 CLBLL_L_B3 }  CLBLL_LL_AMUX
CLBLL_LOGIC_OUTS20  { IMUX_L20 CLBLL_L_C2 }  IMUX_L36 CLBLL_L_D2 } [get_nets {gray_start_inst/gray_w[4]}]
```

Figure 3.22- Gray-Code TDC Constraints

The definition of the routing and placement constraints in the Gray-code TDC not only enabled high linearity to be achieved but also higher performance uniformity across multiple TDC channels. When implementing multiple TDC channel in a complex system, the allocated routing resources on each TDC channel can vary significantly due to the channels positioning and proximity to other sub-systems' logic inside the FPGA. Since the constraints created to a single TDC channel can be applied to different channels directly, system scalability and channel homogeneity can be assured. A more detailed discussion regarding the achieved results and test methodology used is presented in Section 3.6.

3.4.3. Interface

The interface between the Gray-code TDC architecture and the Arm processor is similar to the one used for the TDL TDC. The main difference is just on the application side. On the TDL case, the values obtained from the TDC IP were decoded into coarse, start and stop counts, and these values were directly used to calculate the time interval or to address the calibration table. In the Gray-code, after decoding the values on the AXI frame read, the start and stop fine values must be converted from Gray to binary before being used to calculate the measured time interval value.

3.5. Serial Peripheral Interface (SPI) Interface

Although the adoption of the AXI interface is advantageous when using Xilinx SoC platforms, this interface is not available on every processor. Since the objective of this Thesis is to study an efficient way to port the TDC IP between platforms and technologies, an alternative interface was developed in order to allow the TDC to be implemented as a standalone IP, accessible by multiple processor and microcontroller families. Serial Peripheral interface (SPI) is broadly used in embedded microcontrollers and can operate at an acceptable data rate for the given application [3.7]. Thus, an SPI slave interface was designed to convert the 32-bit parallel output from the TDC's FIFO memory to a serial output, enabling the reduction of the number of pads required at the cost of the maximum sampling rate.

SPI is a de facto standard that can be implemented with a 3- or 4-wire interface. In a 4-wire interface, the signals involved in the communication are: the clock signal (SCLK), generated by the SPI master; the master-out slave-in (MOSI) signal; the master-in slave-out (MISO) signal; and the chip select signal (CS). The chip select is used to select which slave is active, i.e. connected to the master, during a given transmission. The SPI protocol does not specify the number of bits per transmission and enables four different operation modes, which define when the data should be read and updated. The operation mode is limited to the ones supported by the device acting as the Slave.

To support the most common SPI interfaces designed for 8-bit frames transmission, the SPI slave module implemented processes the data sent or received as an 8-bit package. Thus, in order to get a complete 32-bit TDC measurement value, four sequential SPI reads are required. Moreover, the SPI slave interface only supports mode 0, i.e. the data lines are sampled at the rising edge of the SPI clock signal and at the falling edge the data is shifted out. The SPI slave interface was designed according to the state machine presented in Figure 3.23. For debugging purposes, a set of eight dummy registers were implemented to test the SPI read and write operations. These registers are available for future implementations, if the need for configurable parameters arise. A complete read operation consists of a minimum of two data package transmissions. The first data package is the register's address from which the SPI master wants to read. The FIFO was mapped to address zero being the eight dummy registers mapped to the addresses from 1 to 9. The second data package is the data sent by the slave or written by the master. It is only possible to write to the SPI slave interface if the selected address is in the dummy registers address range, otherwise the operation is ignored. When reading from the FIFO, after writing the address zero to the SPI slave interface, four read operations must be performed in order to get a complete TDC measurement value. Only after these, the FIFO read pointer incremented. There is no limit for the consecutive reads that can be done since the FIFO read pointer will automatically reset once the last memory position is reached.

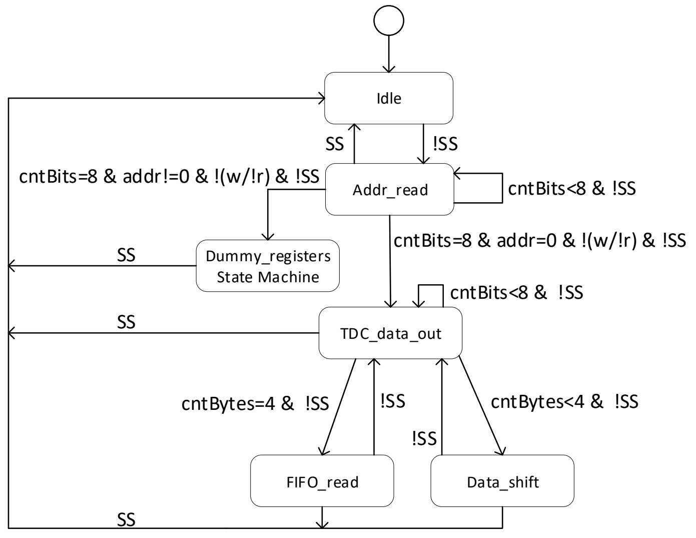


Figure 3.23- SPI Slave State Machine

3.6. Simulation results

To test the proper behavior of the implemented architectures, a set of simulations were developed. The simulation consists of a set of hit signals generated in sequence at random instants. Since one of the core modules behavior is based on the propagation time of the FPGA logic, timing simulations must be done. However, these timing simulations were used only to validate the proper functioning, and not to extract information regarding delay times (worst-case timing models are used in the simulation resulting in propagation delay values that are far from the real values).

3.6.1. TDL TDC Simulation

A simulation of the typical measurement functioning of the TDL TDC is depicted in Figure 3.24. Upon the arrival of the hit signal, it is possible to verify the signal propagation throughout the delay chain (represented by the variable *tdl_val_w[255:0]*). The value is sampled in the next reference clock and the binary decoded value becomes stable after a few nanoseconds. The same happens on the falling edge of the hit signal. It is possible to see that a new hit signal arrives before the register *tdc_val_w[31:0]* gets updated with the previous measurement value. Thus, this start event is ignored due to the implemented hit filter module, preventing erroneous measurement values. The value stored in *tdc_val_w[31:0]* corresponds to the concatenation of the fine stop value (9b), fine start value (53), and the counter value

(0045), indicating that the merge operation is also correct. Thus, it can be concluded that the implemented TDL architecture is working properly.

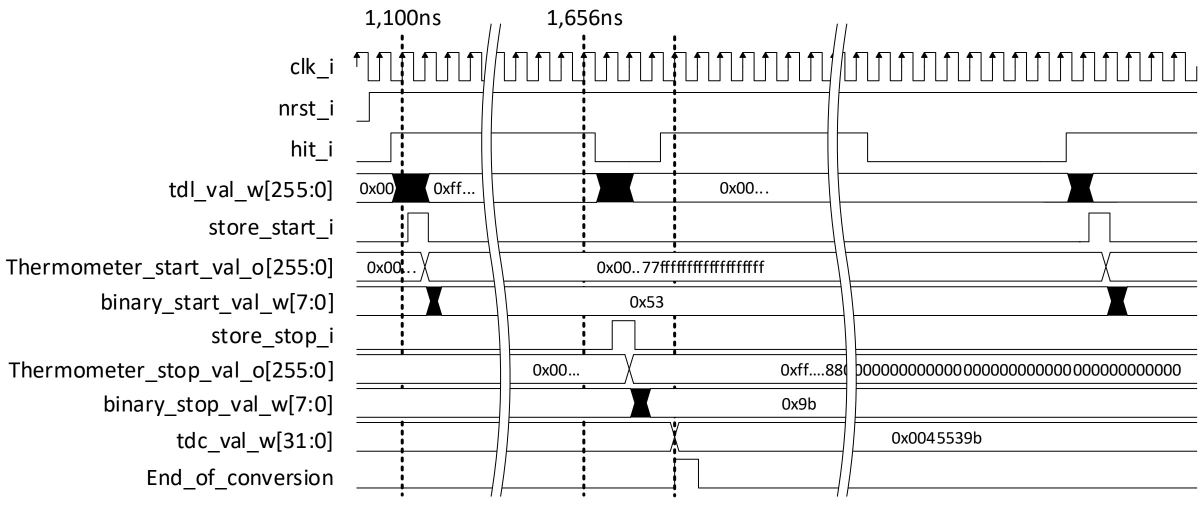


Figure 3.24- TDL TDC Timing Simulation

Analyzing the thermometer code in detail, it is possible to see the occurrence of bubbles (077ffff...). If the hit signal propagation is analyzed with more detail, it is possible to verify that the timing simulation has a limitation when displaying the value of the delay line (see Figure 3.25). The four different carries on the Carry4 element are updated at the same time. Thus, the propagation of the hit signal in the simulation is done in steps of 114 ps (equivalent to four carry cells in the worst-case scenario). Thus, the bubble occurrences in simulation highlight for the mismatch in the routings between the carries and the storing elements. The clock skew between storing elements is also a cause for this behavior. However, since a priority encoder is being used, the bubbles in the code can be safely ignored, and the output of the decoding stage maintains a valid value.

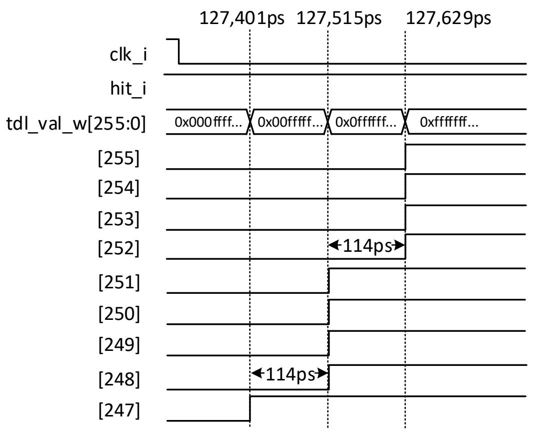


Figure 3.25- Detailed View of the Carry4 Propagation Delay

According to the simulation, a Carry4 element has 114 ps propagation delay, which corresponds to an average value of 28.5 ps per carry cell. In reality, the delay of each carry is more than 10 ps lower. The code density test performed, and presented in the Section 3.7, showed an average carry propagation delay of 17.2 ps, enforcing the statement that timing simulation, when studying TDCs, should be used to test proper system functioning, but not to extract relevant timing simulation.

3.6.2. Gray-code TDC Simulation

The same testbench used in the TDL TDC timing simulation was used to test the Gray-code TDC. The functional timing simulation results are presented in Figure 3.26. The behavior of the architecture is similar to the one of the TDL. The main difference is related with the handling of the hit signal. Although the hit signal is at logic level one during multiple reference clock cycles, the Gray-code fine measurement stage only counts until the next reference clock rise transition (in the TDL scenario the hit signal was always being propagated by the TDL). After it, the Gray-code counter is disabled. This behavior matches with the architecture described in Section 3.4, validating the implemented fine measurement stage. Furthermore, since the obtained result from the fine stage measurement does not need to be decoded, the system is ready for another measurement one reference clock cycle after the hit signal falling edge event.

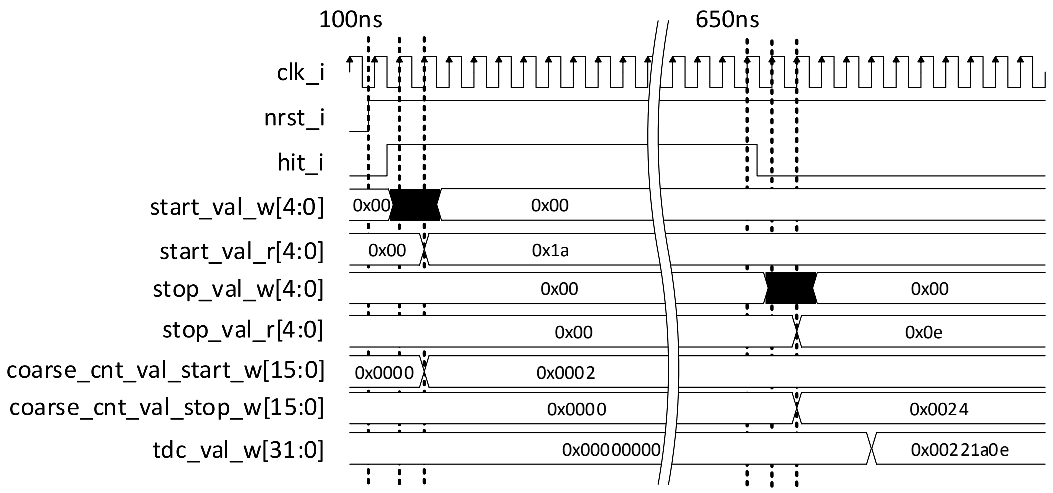


Figure 3.26- Gray Code TDC Timing Simulation

Analyzing the Gray-code start channel in detail it is also possible to verify that the correct Gray sequence is being generated (see Figure 3.27). In this architecture, contrarily to what happened in the TDL case, the steps do not have the same propagation delay. Instead, the delay varies and repeats itself in the pattern identified before (in Section 3.4). Thus, if the routing timings is subtracted to the times obtained from the simulation, the worst-case scenario of the LUT's propagation delay can be calculated according to equation (3.2).

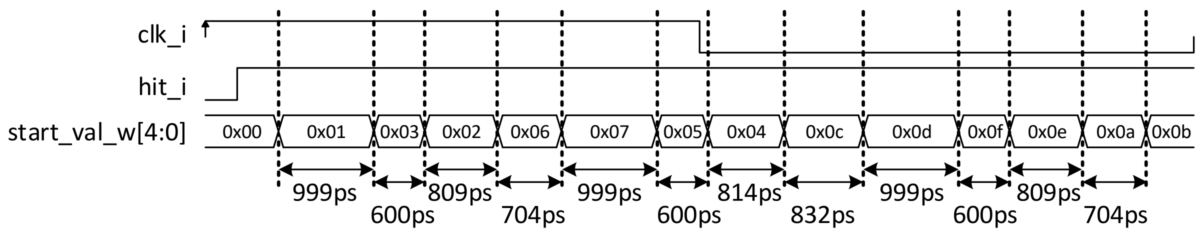


Figure 3.27- Detailed View of the Gray-Code Sequence Generation and step size

$$t_{LUTi} = t_{PD} - t_{ROUTEi}, \qquad (3.2)$$

where t_{LUTi} is the LUT propagation delay, t_{ROUTEi} is the propagation delay of the LUT's output wire to the LUT which will change state for the next count, and t_{PD} is the total propagation delay, extracted from the simulation.

Taking the LUT responsible for generating the least significant Gray-code bit and the 1 (00001) to 2 (00011) transition as an example, the LUT propagation delay would be equal to 123 ps (999 ps - 876 ps, see Figure 3.27 and Table 3.3). As in the TDL scenario, 123 ps is the worst-case propagation delay for all the LUTs. These results conform with the information on the Xilinx datasheet stating that the LUT's propagation delay is independent of the truth table being implemented. Since the first LUT has one less output connection than the remaining LUTs, but all of them have the same propagation delay, one can also conclude that the LUT's propagation delay is independent of its output load (at least for the timing simulation). Furthermore, the routing resources appear as the main delay source, further supporting the statement made previously regarding the impact of controlling it to achieve better linearity. The experimental results presented in Section 3.7 study the differences between manual and automatic routing and its impact on the TDC channel linearity and scalability.

3.7. TDC Performance Assessment

The TDC architectures were deployed in Xilinx Zybo Z7 development board (depicted in Figure 3.1). The Tektronix AFG1022 arbitrary waveform generator was used to generate the time intervals to assess the developed TDCs. Figure 3.28 depicts the test setup used during all the tests performed. It is composed

Table 3.3- Routing Delays for the Gray-Code TDC

STOP CHANNEL	*MANUAL ROUTING*					*AUTOMATIC ROUTING*				
	LUT4	*LUT3*	*LUT2*	*LUT1*	*LUT0*	*LUT4*	*LUT3*	*LUT2*	*LUT1*	*LUT0*
BIT0	**665**	**691**	**686**	**876**	-	**664**	**701**	**696**	**295**	-
BIT1	700	198	193	475	**477**	700	198	193	475	**477**
BIT2	911	732	737	162	**580**	909	730	735	1080	**165**
BIT3	394	306	307	711	**709**	394	306	307	711	**709**
BIT4	296	916	914	518	**513**	297	612	609	330	**514**

by the development board, the waveform generator and a host PC running a MATLAB script to analyze the data read from the board. The TDC IP with the AXI interface and the integrated Arm processor was used (the SPI interface was built to target the ASIC solution). For each architecture, a code density test was performed to extract the real delay of each step of the fine interpolation stage. The results from these tests were used to calculate the non-linearity of the fine measurement module, namely, the DNL and INL.

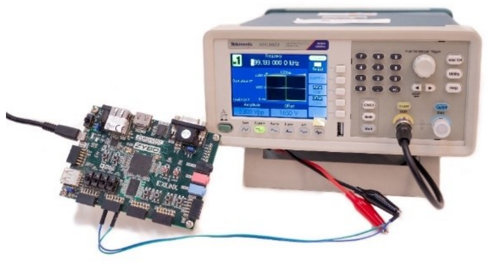


Figure 3.28- FPGA Test Setup

A total of 100 thousand measurements were made to reduce probabilistic errors. A single-shot precision test was also performed, with 100 thousand samples. Then, to reduce the influence of the errors introduced by the arbitrary waveform generator, a 10 measurements average was made, and the precision recalculated. To understand the impact of a calibration mechanism in the TDCs' performance, a post-measurement software bin-by-bin calibration was applied to the 100 thousand samples collected during the single-shot precision test. The calibration tables were created based on the results from the code density test. Then, the single-shot precision and average precision were recalculated after applying the calibration. All the tests were performed at ambient temperature of 25°C and with a power supply of 3.3V. The following assessment results are presented by TDC architecture.

3.7.1. TDL TDC Code Density Test

In order to evaluate the real delay distribution across the implemented TDL, a code density test was performed. The waveform generator was configured to output a square wave signal at a frequency unrelated with the 250 MHz reference clock used, thus creating a sliding window effect on the sampling steps of the TDC, which, in an ideal scenario, would have the same probability to be sampled. The selected frequency was 999133 Hz.

The code density test for the start and stop events propagation is presented in Figure 3.29. Regarding the start signal, it is possible to notice that no step was captured prior to the 22nd step. The last step through which the start event was able to propagate was the 254th. Thus, the start signal can propagate through a total of 232 steps in one clock period. Since the reference clock used is operating at 250 MHz, the average delay of each step when propagating the start signal is 17.2 ps. Notice that almost half of the steps have zero delay. This is mainly caused due to process variation and it is the main reason for the mandatory implementation of a calibration mechanism in FPGA-based TDL TDCs. Since the proposed architecture uses the same TDL to measure both time events, a similar behavior was expected for the stop signal propagation. This can be observed when analyzing the results from the stop signal code density test. The ultra-wide steps are the same (for example, the 200th step) and the number of steps with zero propagation delay is also similar. The main difference is regarding the first bin to be sampled that, in the stop propagation scenario is the 10th step. Since a rising edge (start signal) and a falling edge (stop signal) have different propagation behaviors, this difference was expected. Thus, the average delay per step when propagating a stop signal is 16.4 ps.

The values obtained from the code density test were used to create two calibration tables. Each of the rows in filled with the cumulative sum of the delays of the TDL steps until that row position, i.e. at the 70th row, the value would be equal to the sum of the delays of the first 70 steps of the delay line and so on.

3.7.2. TDL TDC Linearity

The DNL and INL of the TDL were calculated using the data obtained from the code density test, according to the equations (2.5) and (2.6) presented on Chapter 2 (considering 17.2 ps and 16.4 ps as the LSB for start and stop respectively). Figure 3.30 depicts the non-linearity for the start and stop events propagation. For the start event propagation, the maximum DNL is equal to 3.3 LSB (56.16 ps), while for the stop event, a maximum DNL of 3.7 LSB (60.84 ps) was obtained. The INL ranges between -3.8 and 1.8 LSB when propagating the start event, and -3.7 and 2 LSB for the stop event.

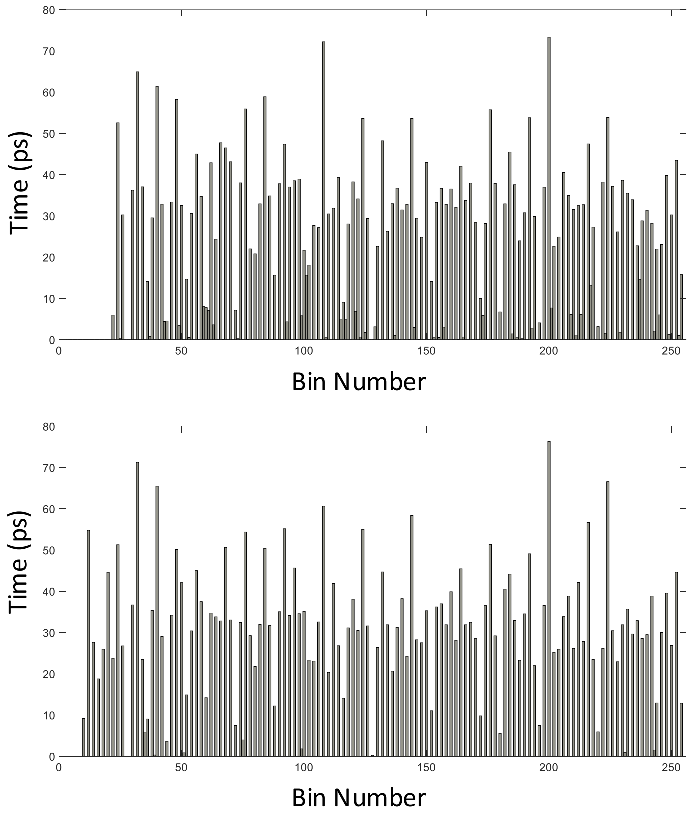


Figure 3.29- TDL TDC Code Density Test for Start (top) and Stop (bottom) event propagation

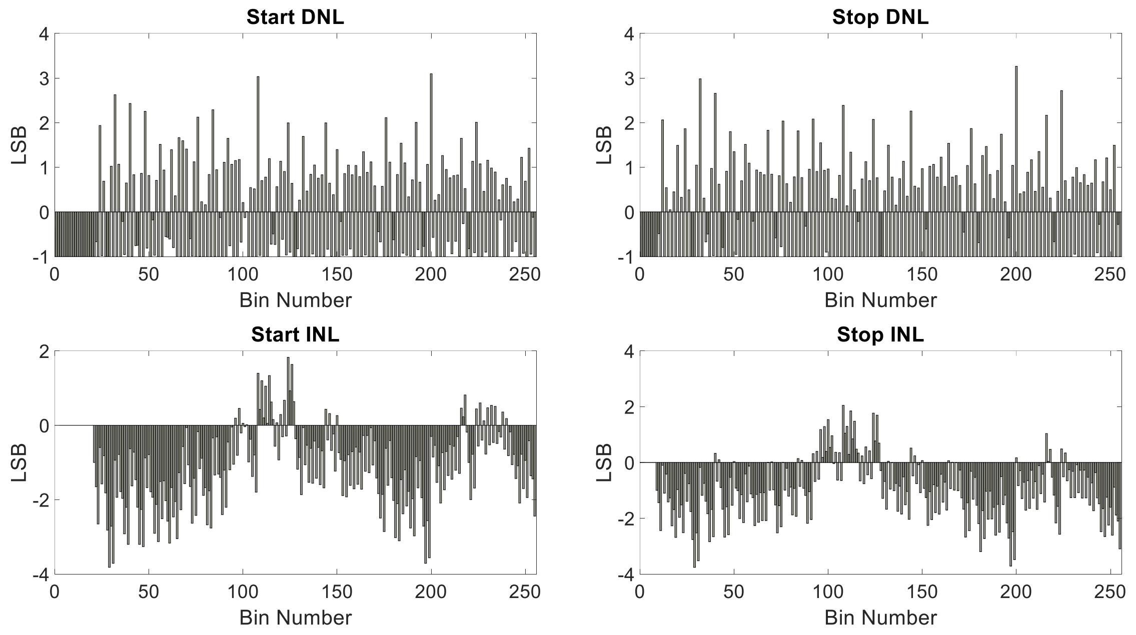


Figure 3.30- TDL TDC Linearity results for start (left) and stop (right) signals propagation

3.7.3. TDL TDC Precision

To assess the TDC precision, the waveform generator was configured to output a square-wave with 999.133 kHz frequency with 50% duty-cycle. The output frequency was verified using an oscilloscope to check the duration of the pulse between the rising and falling edge of the signal (portion of the signal measured by the TDC). A duration between 480.242 ns and 480.434 ns was observed.

From the code density test performed and the linearity results obtained, it is possible to conclude that the TDC precision will be considerably affected if equation (2.14) from Chapter 2 (which uses an average cell delay value) is used to calculate the time interval measurement. This is highlighted in Figure 3.31, which presents the results from the single-shot measurement without calibration (top) and with calibration (bottom). A precision improvement of 40.3 ps (2.4 LSB) was obtained when calibration is applied. The raw precision test shows a 481.007 ns average measurement (573 ps offset regarding the expected value) and a precision of 211 ps. After calibration, the precision is improved to 179 ps, with an average time interval measurement of 480.891 ns (457 ps offset regarding the expected value).

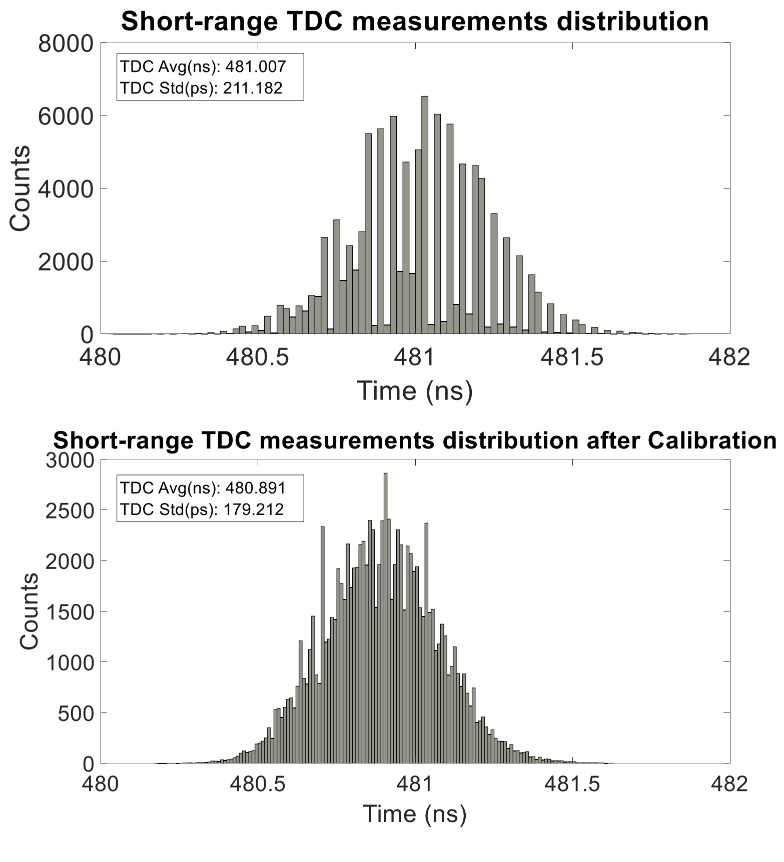


Figure 3.31- TDL TDC Single-Shot Precision before (top) and after (bottom) calibration

However, even with calibration, the obtained precision is still far from the ideal LSB size (approximately 17 ps). The reason for such results may be explained by two factors: first, the existence of multiple ultra-wide bins along the TDL deteriorates the TDC's precision, even when the real cell delays are used to calculate the time interval measured; second, the arbitrary waveform generator jitter and noise introduces errors to the time interval generated. To reduce the influence of the errors introduced by these factors, an average of 10 measurements was performed for both raw and calibrated data. The results are depicted in Figure 3.32 (being the non-calibrated measurement precision depicted in the top graph and the calibrated measurement precision on the bottom graph). A precision of 59 ps and 56.7 ps was attained for the raw and calibrated data, respectively.

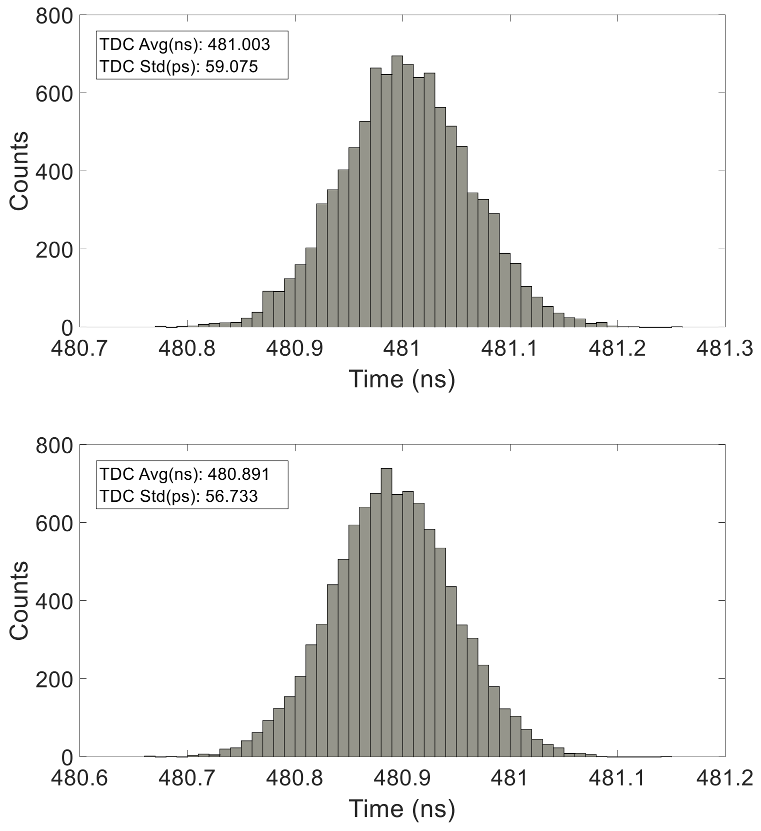


Figure 3.32- TDL TDC 10 Measurement Average Precision before and after Calibration

3.7.4. The synchronizer contribution

In order to understand the necessity of a synchronizer block, a set of measurements with the synchronizer disabled were performed. The results obtained for 1,000 measurements with and without the synchronizer implemented are presented in Figure 3.33, being the results without synchronizer displayed at the top of the figure while the results with the synchronizer enabled displayed at the bottom. As can be observed in Figure 3.33, multiple errors in the range of ±1 coarse counter LSB appear in the measurements. Moreover, there are even measurements where the error is equal to multiple clock cycles. This is because the coarse counter is binary, thus multiple bits can change simultaneously. When this happens, if only some bits are properly updated, errors equal to multiple clock periods can appear. With the synchronizer implemented, no measurement deviations equal or greater than the system clock period were recorded. Thus, it is possible to conclude that, in the FPGA implementation, the synchronizer module is working properly.

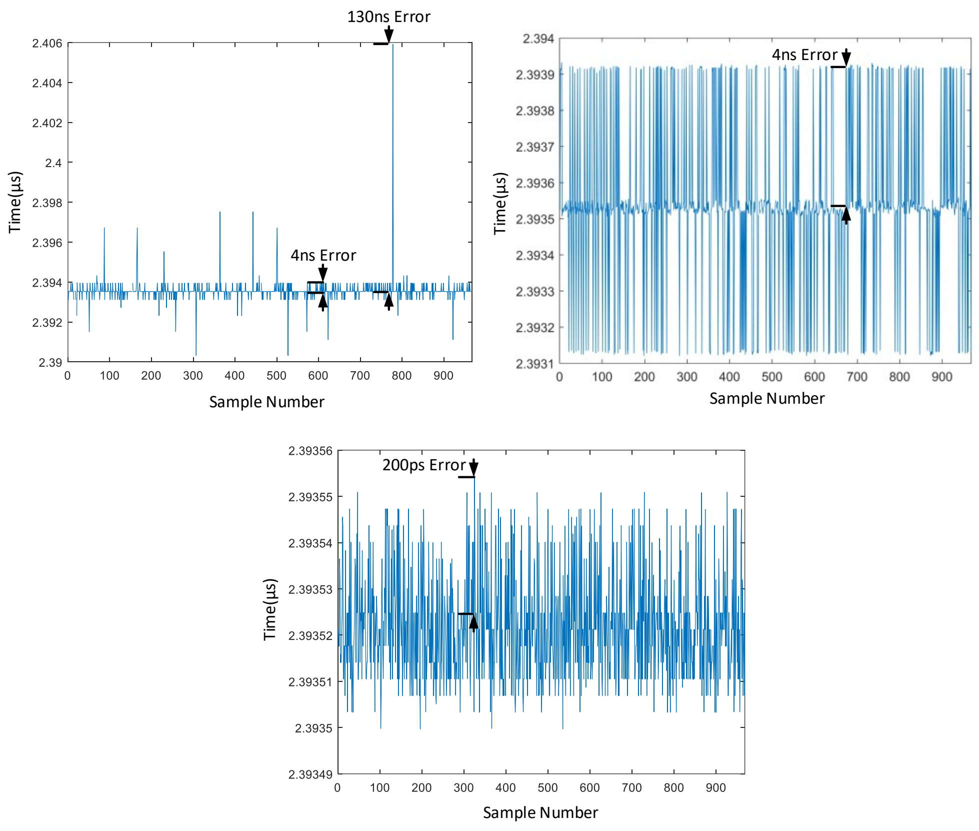


Figure 3.33- Synchronizer Effect on TDC Measurement Value Output.

3.7.5. Gray-code TDC Code Density Test

The same setup used to test the TDL TDC was used during the assessment of the Gray TDC. However, since the proposed Gray TDC architecture was targeting an improvement on the TDC linearity, two different implementations were deployed to FPGA and tested. The first implementation followed the strategy adopted by Wu and Xu [3.4], constraining just the placement of the TDC's channel LUTs and storing elements, while the routing was performed automatically, using the Vivado framework default implementation run.

The second implementation followed the design flow proposed in this Thesis, first placing the TDC's channel cells, then letting the tool perform automatic routing using a *low_net_latency* implementation run, and finally manually routing the nets identified to try to have equal parasitic capacitances. The TDC's start and stop channels' code density test results, for both implementations (default and manual routing), are presented in Figure 3.34 (being the default results are presented on the left while the manual routing implementation results are presented on the right). Since reducing the delays of the slowest nets is usually harder (most of the times impossible when using the *low_net_latency* option), the strategy adopted was to increase the delay of the fastest nets. This resulted in a higher average step delay. However, as can be seen in Figure 3.34, it reduces the delay differences across the TDC channel.

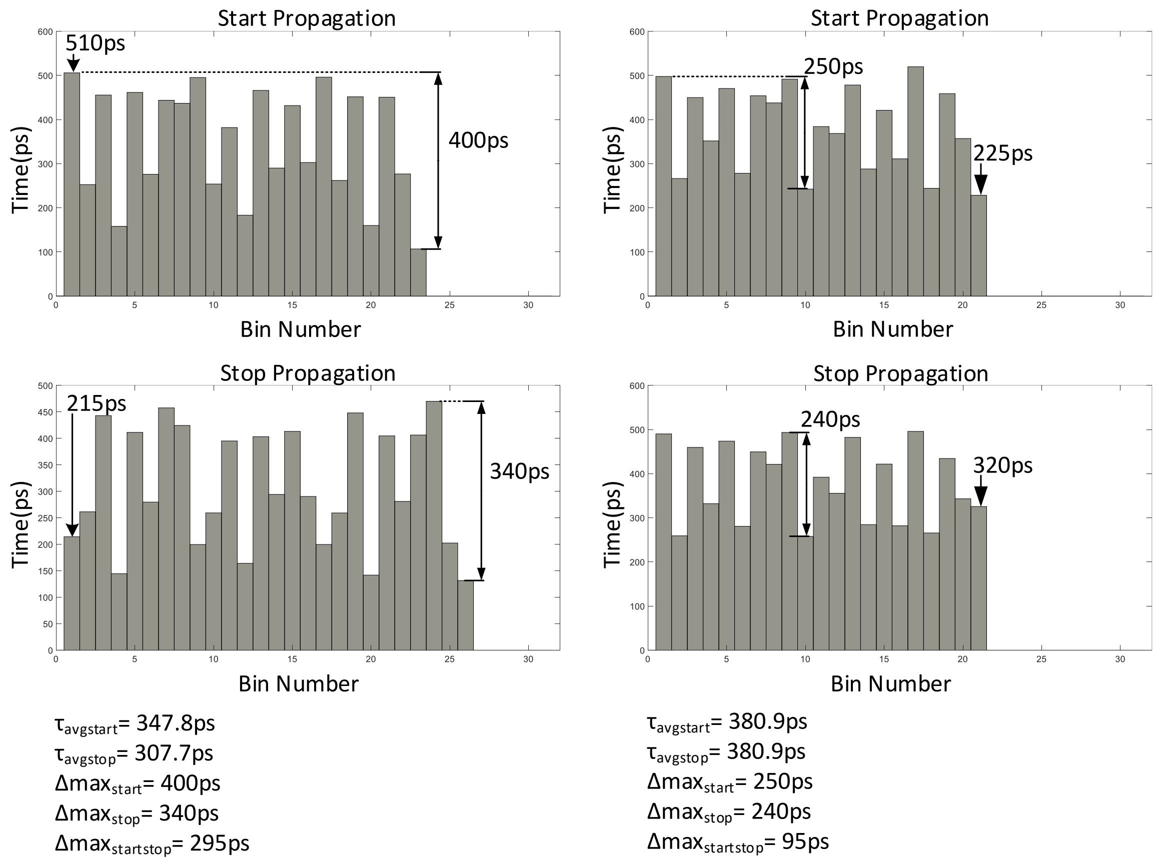


Figure 3.34- Gray-Code TDC Code Density Test for Default and Manual Routings

The previously presented Table 3.3 depicts the pre- and post-manual routing net delays for the stop channel. A 125 MHz reference clock was used in the Gray-code TDC. Thus, the average step delay is 380.9 ps, a 33.1 ps increase regarding the solution where only the placement is constrained. Another important factor to notice is the delay uniformity across channels. When manual routing is performed, the start and stop channels steps' delays are closely matched with a maximum difference of 95 ps for the same step (for the worst-case scenario), while the automatic routing implementation presented a maximum difference of 295 ps. When analyzing the work by Wu and Xu, a maximum difference of 230 ps can be observed. Thus, the proposed implementation improves the TDC scalability, since it will secure a uniform performance when multiple channels are implemented. Furthermore, the same calibration mechanism, for instance a single bin-by-bin calibration table, can be deployed to calibrate multiple TDC channels due to its similar delays' distribution, enabling resource and power savings.

3.7.6. Gray-code TDC Linearity

The DNL and INL calculated from the code density test results is presented in Figure 3.35 (being the default results are presented on the left while the manual routing implementation results are presented on the right). As expected, the manual routing scenario shows higher linearity with a maximum DNL of 0.38 LSB and an INL in the range of 0.01 and 0.7 LSB for the start channel (worst case) (see Figure 3.35).

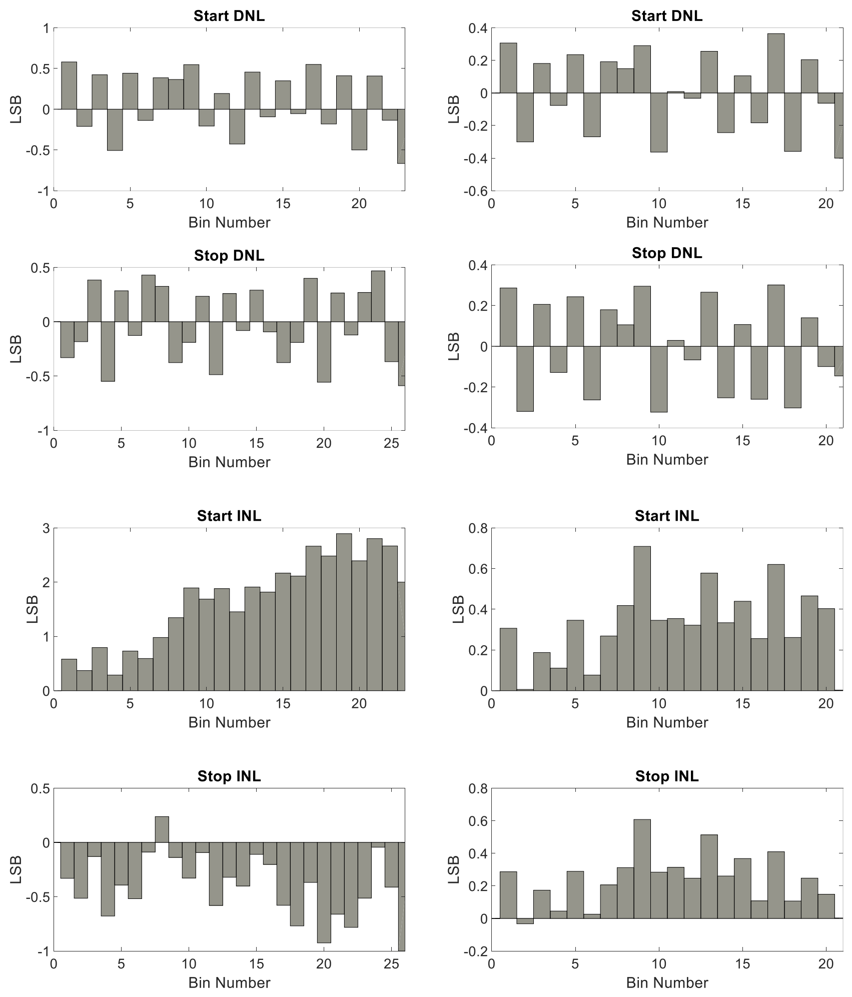


Figure 3.35- Gray-Code TDC Linearity Results for Default and Manual Routing

3.7.7. Gray-code TDC Precision

Although calibration can be applied to the TDC channel, given the obtained linearity, high performance without calibration is expected. The single-shot precision test results are presented in Figure 3.36 (being the default results are presented on the left while the manual routing implementation results are presented on the right). A difference of 2.2 ps precision can be seen when bin-by-bin calibration is applied to the manual routing implementation. On the automatic routing implementation, the precision difference is about 109 ps. Thus, the proposed design flow proved to be efficient in improving the performance of the TDC channel without extra resource costs. The trade-off is purely in the design stage which require an additional manual step.

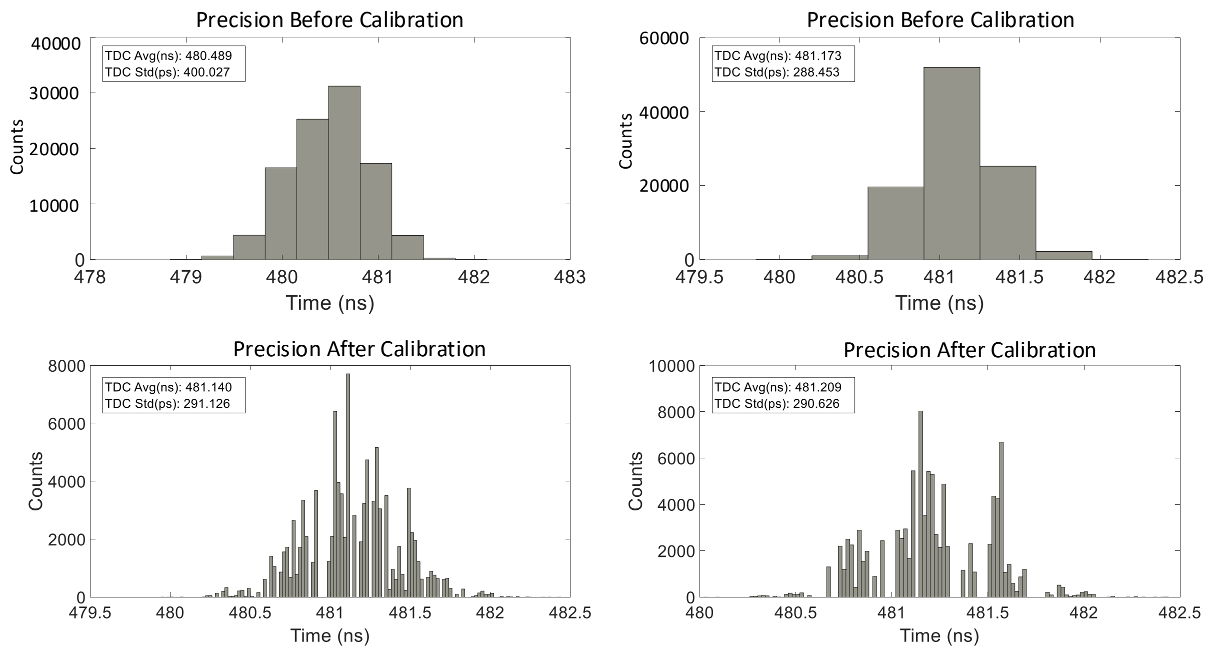


Figure 3.36- Gray-Code TDC Single-Shot Precision for Default and Manual Routing

Again, to reduce the influence of the errors introduced be the waveform generator, another test, in which each measure was a 10-measurement average was performed. The precision results are depicted in Figure 3.37 (being the default results are presented on the left while the manual routing implementation results are presented on the right). Here is important to highlight that the non-calibrated average precision of the manual routed TDC was able to surpass the average precision of the calibrated default routed TDC.

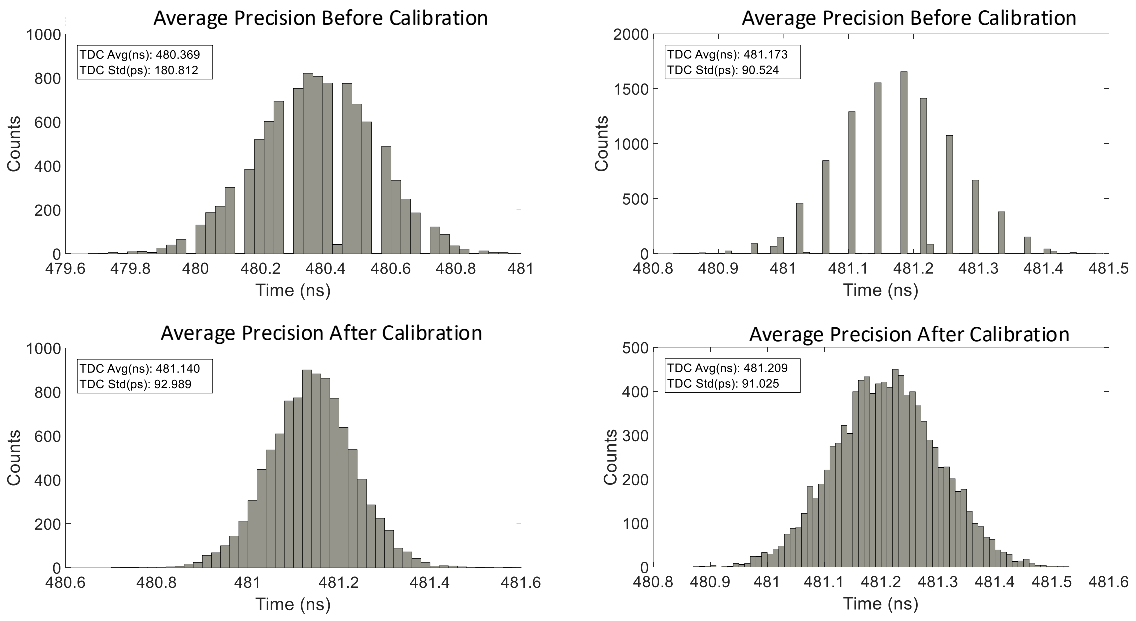


Figure 3.37 - Gray-Code TDC Average Precision for Default and Manual Routing

3.8. Discussion

As expected, the TDL implementation is able to achieve higher performance than the Gray-code architecture. However, the low resource utilization of the Gray-code architecture might be interesting for applications such as Flash LiDAR. With technology scaling down and new FPGA models arriving to the market with higher performances, like the recent Xilinx Ultrascale+, Gray-code TDC architectures might reach resolutions suitable for these sets of applications, at very low resource and power consumption. On these latter FPGA technologies, the routing impact will certainly be greater, thus manual routing when implementing a Gray-code TDC will be mandatory, not only to improve linearity, but also to avoid possible scenarios where some steps are not sampled.

It is also important to notice that, according to the waveform generator datasheet, the jitter when outputting a square-wave signal is typically bellow 1 ns, a value way above the resolution of the implemented TDCs. Thus, the measured TDCs' precision is certainly being deteriorated by the precision of the waveform generator.

Although both implemented architectures are purely digital and described using an HDL, the Gray-code architecture might not be suitable for ASIC migration. The homogeneous behavior displayed by the Gray-code TDC is only possible due to the existence of LUT cells which have a propagation delay unrelated to the function that is being implemented. Thus, every bit of the Gray-code will have a similar delay, no meter the complexity of the implemented expression (as long as the number of inputs does not exceed the available six inputs of the LUT). In ASIC, each bit will have a different implementation and thus, different combinatorial levels, which may lead to some bits having a much longer propagation than others. In Chapter 4, a preliminary study on the Gray-code TDC architecture migration is done to understand the viability of the port.

Regarding the TDL, a closer match can be done when implementing in ASIC by replacing a carry stage by a standard cell (like a buffer or a AND gate with the inputs shorted). Thus, this architecture seams more suitable for the migration. The main challenge will be to minimize the TDL non-linearity due to the lack of dedicated routing channels and the expected random placement done by the layout tools. Furthermore, in FPGA, a PLL was used to generate the reference clock phases, required by the synchronizer block. In ASIC implementation, in order to avoid the implementation of a PLL, the clock tree will have to be manipulated to generate the required clock phases. As in the case of the Gray-code TDC, in Chapter 4, a preliminary study was done to study the feasibility of the porting. According to the results obtained, the architecture to be ported was selected.

The implementation of FPGA-based TDC prototypes allowed to acquire the needed knowledge to test ToF measurement systems, crucial for properly testing the future ASIC prototype. Moreover, the Gray-code TDC, although not presenting highest precision, enabled the study of the influence that the routing and placement may have during the TDC implementation. These lessons were extremely important when porting a TDC architecture from FPGA to ASIC. In FPGA, the dedicated routing and mandatory vertical placement of carry blocks automatically minimize these issues, being process variations the main cause for the TDL non-linearity. In ASIC, these structured placement and routing must be mimic, otherwise the automatic placement and routing may lead to scenarios identical to the Gray-Code architecture (in which some steps might never be sampled depending on the manual routing performed).

3.9. Conclusion

High performance Time-to-Digital Converters are core structures on multiple time-of-flight systems. However, these systems are highly hardware dependent and required a custom-made process, which increases development time and costs. Recent FPGA technology developments enabled these platforms to reach performance values capable of competing with ASIC for some applications. Thus, FPGA-based TDC research interest has grew in recent years and many solutions are already available. However, the main problem persists: how to migrate an already validated TDC prototype to ASIC without requiring a custom-made cell design?

This chapter described and presented two different FPGA-based architectures, one targeting the maximum achievable resolution, and another focusing on resource saving and scalability, with the objective to study TDC systems and analyze how to seamlessly migrate the system from a prototype platform to a massive production ASIC. The major difference between the two architectures is the fine measurement module, while the remainder modules are the same, thus proving modularity and flexibility. All the modules were explained in detail, with higher emphasis on the fine measurement block. The details of implementation were discussed, together with all the systems' constraints and limitations. Functional timing simulations were presented to assess the systems' behavior, and the results explained.

The two TDC architectures were implemented, tested and characterized. The resolution, precision and non-linearity of each architecture was discussed. Based on the obtained results and the hardware structure of the implemented architectures, the ASIC migration feasibility was discussed at the end of this chapter. Finally, the main lessons learned from the FPGA-based TDC implementation process are presented, explaining how these can be important when migrating the proposed TDC architectures to ASIC.

References

[3.1] Digilent, "Zybo Z7 Board Reference Manual." Digilent, 2018. [Online]. Available: https://www.xilinx.com/support/documentation/university/XUP%20Boards/XUPZYBO/documentation/ZYBO_RM_B_V6.pdf.

[3.2] Xilinx, "Zynq-7000 SoC Data Sheet: Overview." Xilinx, 2018. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.

[3.3] Arm, "Cortex-A9: Technical Reference Manual." Arm, 2012. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0388f/DDI0388F_cortex_a9_r2p2_trm.pdf.

[3.4] J. Wu and J. Xu, "A Novel TDC Scheme: Combinatorial Gray Code Oscillator Based TDC for Low Power and Low Resource Usage Applications," in 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), 2019, pp. 1–7.

[3.5] D. Chaberski, R. Frankowski, M. Zieliński, and Ł. Zaworski, "Multiple-tapped-delay-line hardware-linearisation technique based on wire load regulation," Measurement, vol. 92, pp. 103–113, Oct. 2016.

[3.6] XIlinx, "Vivado Design Suite User Guide: Using Constraints," 2018. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug903-vivado-using-constraints.pdf.

[3.7] P. Dhaker, “Introduction to SPI Interface,” 2018. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html#. [Accessed: 13-Nov-2019].

4. ASIC-based TDC Development

Several conclusions were drawn from the implemented FPGA-based TDC prototypes, which are relevant when migrating or designing a synthesizable TDC architecture for ASIC. The implementation of the FPGA-based TDL architecture enabled the study of the impact of process variation on the TDC linearity. Also, the importance of the skew on the clock network distribution was also demonstrated with the appearance of ultra-wide bins on the clock region crossing steps. On the other hand, the Gray-code architecture implementation enabled the analysis of the routing impact on the TDC's performance metrics, showing that when resolutions as low as a few hundred of picoseconds are targeted, even the small parasitic loads introduced by routing can deteriorate the TDC linearity. Finally, the comparison of the two FPGA-based architectures implemented showed that a structured placement and dedicated routing enable higher performance to be achieved while fostering system scalability and performance homogeneity across multiple channels.

In this chapter, the migration and development process used to implement one of the developed FPGA TDC architectures to ASIC is described. Since the ASIC-based TDC architecture being implemented is the same as the one implemented in FPGA (described in Chapter 3), no further details about the architecture will be given. Only specific migration considerations will be addressed during this chapter. To perform a seamless migration between FPGA and ASIC, the ASIC development tools, and design flow are analyzed to understand the required changes. This analysis focus on the Register Transfer Level (RTL) generated by the ASIC tools, to understand the limits and compromises of each of the FPGA-based architectures prototyped. The architecture which achieved better results was selected for layout and fabrication. The entire design flow process of the selected architecture is described in detail, including all the developed scripts used to configure the ASIC tools.

The Chapter is structured as follows: Section 4.1 describes the development environment, the technology library used to implement the ASIC TDC and a comparison between the adopted digital design flow and typical one; Section 4.2 presents the analysis on the migration of the FPGA-based TDC architectures, discussing the obtained results and identifying the selected architecture for porting; the synthesis scripts and procedures are explained in Section 4.3; the layout considerations and scripts are described in Section 4.4; Section 4.5 presents the developed testbenches and the results from time simulation; the tape-out process is briefly addressed in Section 4.6; finally, Section 4.7 summarizes the Chapter.

4.1. ASIC development environment

There are multiple tools, from multiple vendors, that assist designers on digital, analog and mixed-signal integrated circuit development. These IC design tools usually offer two different interfaces: a TCL scripting interface, that allows the user to navigate throughout the multiple design phases using command line inputs (useful for automating the design flow); and a graphical interface, that guides the user throughout the design phases and allows for more precise manual changes and to visually analyze the design structure and statistics.

When the same vendor tools are used across all design phases, the border between phases is blurred and completely abstracted to the user. However, when tools from different vendors are used in different design stages, the user must understand the various outputs and inputs that are required for all stage's transitions. In the integrated circuit industry, Synopsys' Design Compiler is considered one of the best design synthesis tool, while Cadence's Innovus (former Encounter) and Virtuoso are the ones most frequently used for layout planning and tape-out. Thus, a multivendor design flow is usually adopted. In the following sections, a brief description regarding the digital IC tools used in this Thesis is presented.

Since the TDC performance metrics depend heavily on the hardware circuit, it is also important to understand the technology in which the TDC is being implemented. Thus, Section 4.1.2 describes the Taiwan Semiconductor Manufacturing Company (TSMC) 0.18 µm 6-metals process technology.

4.1.1. Development tools

A multivendor development environment was adopted, following the current industry trend. The Verilog analysis and RTL synthesis were done using Synopsys' Design Compiler. The Layout of the obtained Netlist was performed using Cadence's Innovus, and SimVision was used during behavioral and functional simulations. The pad ring definition and final DRC and LVS checks were done using Cadence's Virtuoso.

Design Compiler (DC)

According to Synopsys Design Compiler datasheet [4.1], DC Ultra is a RTL synthesis tool which performs concurrent timing, area and power optimization to guarantee better time Quality of Results (QoR). Apart from analysis, elaboration, compilation and static timing analysis of the developed design, DC Ultra offers a set of graphical interfaces and tools to help study the design's critical paths and possible congestion areas. On later versions of the software, cross-probing between the RTL source code and other design views (like the Netlist schematic view) was introduced, enabling designers to have better control and understanding regarding the optimizations and design changes that are being performed by the tool, and efficiently identify possible design issues in early stages of the development [4.1], [4.2].

The main optimization operations performed by the tool are based on arithmetic changes, logic duplication (to reduce the high fan-out loads on critical paths), design ungrouping (to reduce silicon area and achieve better timing performance), buffer insertion on high fan-out nets (to improve the total negative slack), and register retiming (which can also add pipeline registers on long combination paths) [4.1], [4.3]. The DC Ultra also enables the automatic addition of scan registers for test and debugging purposes.

Innovus

Genus is Cadence's framework for digital IC design. From the set of tools included, Innovus is used in the layout step (place and route). One of the main distinctive features of this tool, when compared to its competitors is the placement engine, GigaPlace. The GigaPlace follows a slack driven approach, as opposed to the traditional "time-aware" approach [4.4], [4.5]. According to Cadence [4.4], this enables a concurrent convergent optimization of both electrical and physical metrics. The new placement engine builds slack models considering the design floorplan, routings topologies, congestion and other electrical constraints, and optimizes the design placement.

Another distinctive feature is the clock tree synthesis (CTS) engine. In order to improve useful skew and merge physical optimization with clock tree synthesis, Innovus introduces a new CTS engine named Clock

Concurrent Optimization (CCOpt). The optimizations are based on true propagated clocks and consider on-chip variations (OCV) [4.4].

Apart from placing and routing the design, the tool also offers verification and validation options in order to test the implemented layout against the technology's DRCs. Similar to Synopsys' tools, Innovus has a TCL interface and a graphical interface, in which all the design characteristics can be inspected in detail, including, among others, power, electromigration, routing congestion analysis, clock tree debugging, design hierarchical view search.

SimVision

SimVision, also known as NCSim, is a Cadence software tool for debugging digital, analog, and mixed-signal designs [4.6]. It supports testbenches written in Verilog, SystemVerilog, VHDL and SystemC languages (or a combination of these). This tool offers multiple graphical functionalities that simplify the debugging process and can be used for behavioral and functional simulation. It includes support for functional timing simulations using standard delay format (SDF) files, ideal to validate a post-layout design.

Virtuoso

Virtuoso is an analog and mixed-signal design environment that offers multiple capabilities for electrical analysis and verification [4.7]. A graphical user interface and a TCL-based command line supports the development, when using this tool. In this Thesis, Virtuoso was used solely to create the pad ring for the developed chip, to perform the last DRC and LVS verifications and validations, and to tape-out the design.

4.1.2. Technology adopted

Depending on the technology adopted, some files required by the IC design tools to perform various verifications might not be available. This limits the depth of the analysis that can be performed. For instance, if only the worst-case capacitance tables (*captables*) are available on a technology pack, then, a best-case scenario analysis cannot be performed. Furthermore, if the layout of the digital cells is not available, it is not possible to perform a complete Design Rules Check (DRC) and Layout Versus Schematic (LVS) verification.

The set of available technologies for ASIC design was limited to the AMS 0.35 μm technology package and the TSMC 0.18 μm technology package (the ones available at the International Iberian

Nanotechnology Laboratory). The development package selected was the TSMC 0.18 µm, since it was expected that the lower node technology would result in higher performances to be achieved on synthesizable TDCs.

There are three different available libraries with different routing resources namely, 4-, 5-, and 6-metal layers, and multiple combinational and sequential logic cells. In the adopted TSMC package, the core library cells are always designed using metal layer 1. The 6-metal layers library was used to implement the proposed ASIC-based TDC. The technology package also includes Engineering Change Order (ECO) cells (designed using metal layers 1 and 2) to enable minor changes to the design after tape-out.

According to TSMC documentation, the digital standard cell library is compatible with a vast set of design tools. The development package is described in multiple formats, like the *.db* and *.lib* format required by Synopsys Design Compiler, and *LEF* files, required by layout tools. Moreover, these libraries are described in multiple timing models (like Non-Linear Delay Model - NLDM- and Composite Current Source Model - CCS), which allow the user to reach a compromise between the tool's run time and static timing analysis precision. Each timing model can be described based on the best, worst, or typical case scenarios.

A timing model is composed by tree models: the driver; the receiver; and the net. Driver and receiver models are characterized using a circuit simulation, like SPICE simulator LTSPICE. The wire model can be extracted from the layout using the various metal, via and contact parameters (among others), or it can be estimated. Only NLDM and CCS timing models are available in TSMC 0.18 µm technology package used. Thus, the other timing models will not be described in this Thesis. NLDM estimates cells' delays and transition times for the driver model based on six points (three for the input and three for the output). These points are: in/out slope lower threshold; in/out slope higher threshold; and in/out delay threshold. The receiver model is characterized using a single capacitor (load). For technology nodes above 65 nm, this model suffices for proper static timing analysis. However, when targeting 65 nm technology nodes or lower, the 3-point schema used by NLDM is not sufficient to properly reflect the circuits' non-linearity during static timing analysis. Furthermore, the miller effect on the receiver side, which in small impedance nets dominates the delay calculation, is not captured by NLDM. CCS models provide better accuracy when the net impedance is high, when compared to the driver resistance, since it models the driver as a current source. Regarding the receiver model, CCS is very similar to NLDM. However, the capacitance is divided in two, giving the model more granularity and enabling it to account for the miller's effect. Since 0.18 µm technology node is being used, the NDLM models were used in this Thesis.

During synthesis only the *.db* files are required for static timing analysis, since placement information is still not available, and routings are considered ideal. However, during layout, in order to perform proper time closure, there are additional files that must be provided. These files comprise: *.lef* (which contains the physical characteristics of the library being used); worst and best cases timing model libraries, for multi-corner multi-mode (MCMM) analysis; capacitance tables (*.captable*), used to model the interconnect parasitics of the design. Ideally, a worst- and best-case capacitance tables should be provided to the layout tool. However, the TSMC kit only provides the typical-case capacitance table file. Therefore, the same file had to be used when creating the time analysis corners, used during layout.

4.1.3. Design flow

When implementing digital systems, the traditional design flow can usually be automated by the design tools. As can be concluded by the aforementioned description of the design tools, the focus is on area, power and timing optimizations, which more often than not lead to changes on the generated Netlist when compared to the inputted RTL design. Although this might be advantageous to several designs, there are scenarios where it can lead to erroneous circuit behavior. Thus, when implementing a synthesizable TDC in ASIC, some changes must be performed to the typical design flow (see Figure 4.1). Typical HDL designs are technology independent, however this is not the case of a HDL TDC design. Thus, a previous study of the technology to be used is required to select the logic elements that will be selected to build the fine stage measurement. Afterwards, depending on the TDC architecture, a set of additional constraints must be loaded during synthesis to avoid the optimization of certain parts of the design (specifically if delay lines are used). These optimization constraints must also be included in the synthesis exported constraints file. As verified during the implementation of the Gray-code architecture in FPGA, routing has great influence on the TDC linearity. Therefore, the placement and routing of the design during layout phase must also be constrained, in contrast to the traditional time-driven placement and routing, implemented by the IC design tools. The remaining of the design flow is similar to the typical one. Figure 4.1 presents the digital design flow adopted during the migration of the FPGA-based TDC architectures. The additional synthesis' and layout's constraints used are explained in detail in Sections 4.3 and 4.4.

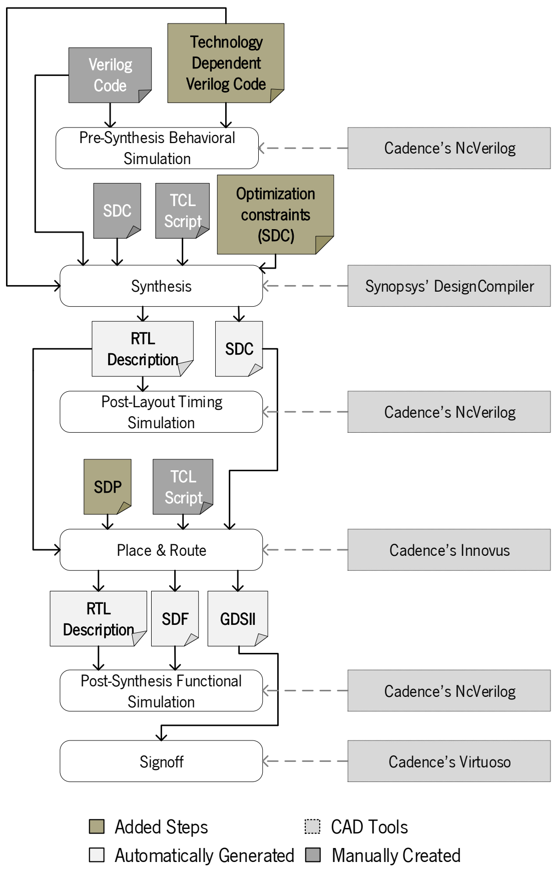


Figure 4.1 - Adopted Design Flow

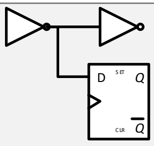
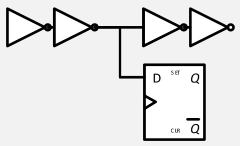
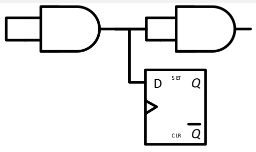
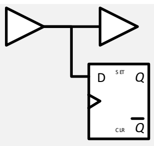
4.2. TDC architecture migration

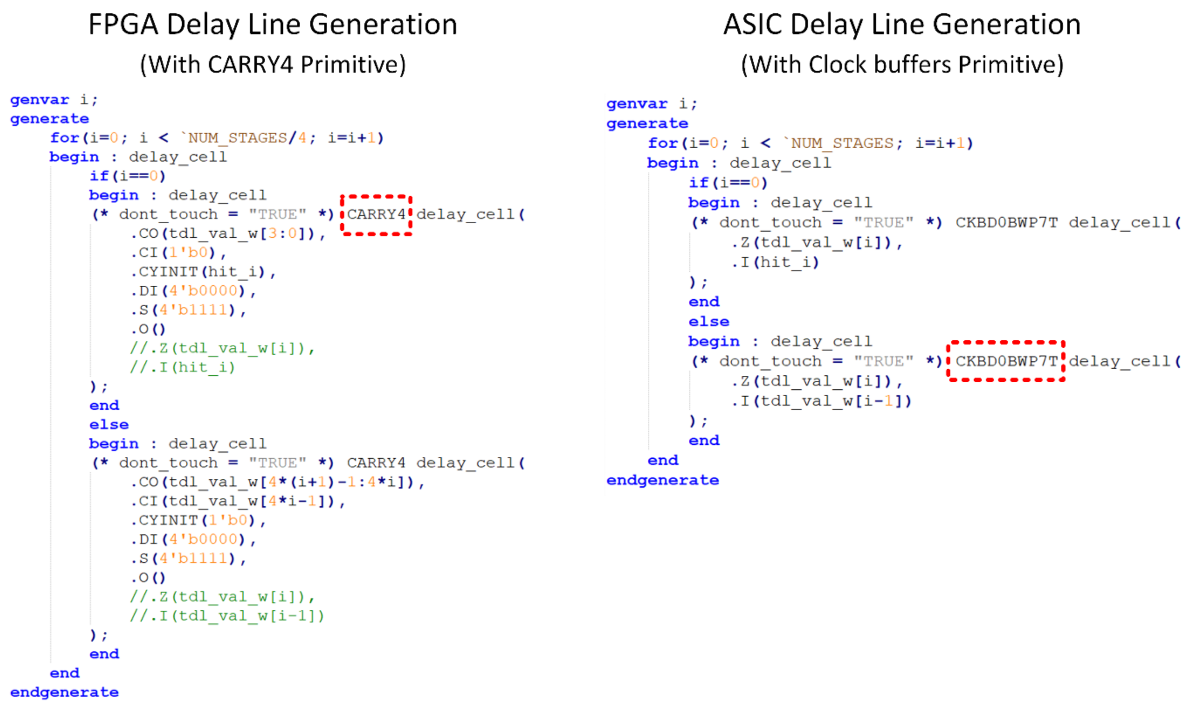
A preliminary study aiming to understand which one of the implemented FPGA-based TDC architectures would attain better performance when migrated to ASIC was made. This study was based on the obtained synthesized Netlists and the information available on TSMC 0.18 µm technology datasheet for the typical operation scenario. During this analysis, routing and placement was treated as ideal, i.e., not influencing the steps' propagation delays.

4.2.1. TDL preliminary results

The FPGA-based TDL TDC was implemented in a Hardware independent language, however it has a direct reference to a technology dependent cell responsible for creating the delay line. Thus, this HDL code must be changed to instantiate a cell available on the technology library being used. The TSMC digital standard cells library was analyzed to select which cell should be used in the TDL implementation. Since the same TDL is propagating the start and the stop signals, ideally the cell should present a similar low-to-high and high-to-low transition time from input to output. This type of characteristic is common on cells used to implement clock trees. Therefore, the set of clock buffers, inverters, and gates were analyzed. The cell that theoretically would offer better resolution would be a clock inverter. However, since the developed decoders expect a thermometer code with a sequence of 1s followed by a sequence of 0s (or vice-versa), if inverters were to be used, each TDL step would have to be comprised by two inverters, otherwise the decoder blocks would have to be changed. Another solution would be to use a clock AND gate with short-circuited inputs. Nevertheless, both solutions have higher propagation delay than the one obtained when a single clock buffer per step is used (see Table 4.1). So, the clock buffer with lower propagation delay and sufficient fan-out to supply the sampling flip-flop and the next step clock buffer was selected, since it was able to comply with the resolution requirements established for this Thesis application. According to the TSMC typical case datasheet, the CKBD0BWP7T has the lowest low-to-high and high-to-low propagation delays and fulfils the fan-out requirement. A comparison between the generate block used to implement the delay line in FPGA and the ASIC one is presented in Figure 4.2.

Table 4.1 - TSMC clock digital cells propagation delay analysis

Digital Cell	*Connection Schema*	*Total Fanout Capacitance (pF)*	*High-to-Low Propagation Time (ps)*	*Low-to-high Propagation Time (ps)*
Inverters *(CKND0BWP7T)*		0.007966 (one stage)	62.4	68.8
		0.003597 + 0.007966 (two stages)	110.3	121.9
AND gates *(CKAN2D0BWP7T)*		0.009842	141.8	126.8
Buffers *(CKBD0BWP7T)*		0.004137+0.004369 0.008506	107.1	105.1


FPGA Delay Line Generation
(With CARRY4 Primitive)
ASIC Delay Line Generation
(With Clock buffers Primitive)

```
genvar i;
generate
    for(i=0; i < `NUM_STAGES/4; i=i+1)
    begin : delay_cell
        if(i==0)
        begin : delay_cell
        (* dont_touch = "TRUE" *) CARRY4 delay_cell(
            .CO(tdl_val_w[3:0]),
            .CI(1'b0),
            .CYINIT(hit_i),
            .DI(4'b0000),
            .S(4'b1111),
            .O()
            //.Z(tdl_val_w[i]),
            //.I(hit_i)
        );
        end
        else
        begin : delay_cell
        (* dont_touch = "TRUE" *) CARRY4 delay_cell(
            .CO(tdl_val_w[4*(i+1)-1:4*i]),
            .CI(tdl_val_w[4*i-1]),
            .CYINIT(1'b0),
            .DI(4'b0000),
            .S(4'b1111),
            .O()
            //.Z(tdl_val_w[i]),
            //.I(tdl_val_w[i-1])
        );
        end
    end
endgenerate
```

```
genvar i;
generate
    for(i=0; i < `NUM_STAGES; i=i+1)
    begin : delay_cell
        if(i==0)
        begin : delay_cell
        (* dont_touch = "TRUE" *) CKBD0BWP7T delay_cell(
            .Z(tdl_val_w[i]),
            .I(hit_i)
        );
        end
        else
        begin : delay_cell
        (* dont_touch = "TRUE" *) CKBD0BWP7T delay_cell(
            .Z(tdl_val_w[i]),
            .I(tdl_val_w[i-1])
        );
        end
    end
endgenerate
```

Figure 4.2- FPGA vs ASIC HDL Comparison

The modified code was synthesized, avoiding the delay line optimization, according to the process described in Section 4.3. Analyzing the fine measurement module in detail, it is possible to verify the correct delay line generation (see Figure 4.3). Accordingly, a good estimation for the expected TDC performance can be obtained from the TSMC datasheet, using equation (4.1) and (4.2).

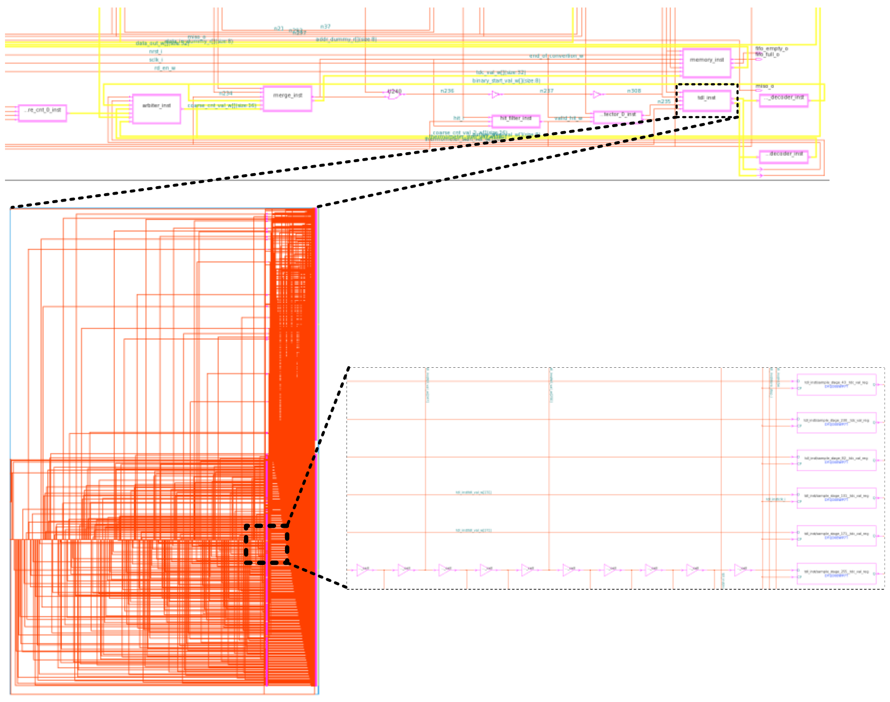


Figure 4.3- TDL Synthesized Netlist Overview

$$t_{PLH} = 0.0743_{(ns)} + 3.6185_{(ns/pF)} * C_{load_{(pF)}}, \qquad (4.1)$$

$$t_{PHL} = 0.0785_{(ns)} + 3.3704_{(ns/pF)} * C_{load_{(pF)}}, \qquad (4.2)$$

where t_{PLH} and t_{PHL} are the low-to-high and high-to-low propagation delays (in nanoseconds), respectively, and C_{load} is the total load being driven by the cell (in picofarad).

Considering that the clock buffer used has an input capacitance equal to 0.004137 pF and that the generated flip-flop D pin has a 0.004369 pF capacitance, the typical propagation delay of each step is approximately 105 ps and 107 ps for low-to-high and high-to-low transitions, respectively. These results comply with the defined LiDAR application's requirements, making the migration of this architecture feasible. The migration process can also be fully automated, requiring only a change in the instantiation of the cell used to build the delay line, if a different technology is desired. Regarding linearity, it is known that longer delay chains tend to have its performance degraded. This is mainly caused by process mismatch (something that cannot be controlled by design), temperature and voltage variations (which can be minimized by layout). Thus, if this architecture is to be ported to ASIC, the layout will have a significant impact on the system's linearity. The layout consideration will be further discussed in Section 4.4.

4.2.2. Gray-code TDC architecture preliminary results

The Gray-code TDC architecture does not need any Verilog change before being synthetized by the ASIC tools. It also does not require any special attention regarding optimization constraints when being synthesized. However, contrarily to what happens in FPGA-based implementations, where each Gray-code bit was generated by a single LUT, the resultant Netlist in ASIC is composed by multiple logic gates, arranged in a combinatorial loop, being the different Gray-code bits extracted at different points of the circuit (see Figure 4.4). Thus, while in FPGA every LUT has the same propagation delay independently of the truth table implemented, in ASIC development the various paths of the Gray-code combinatorial logic Netlist must be analyzed.

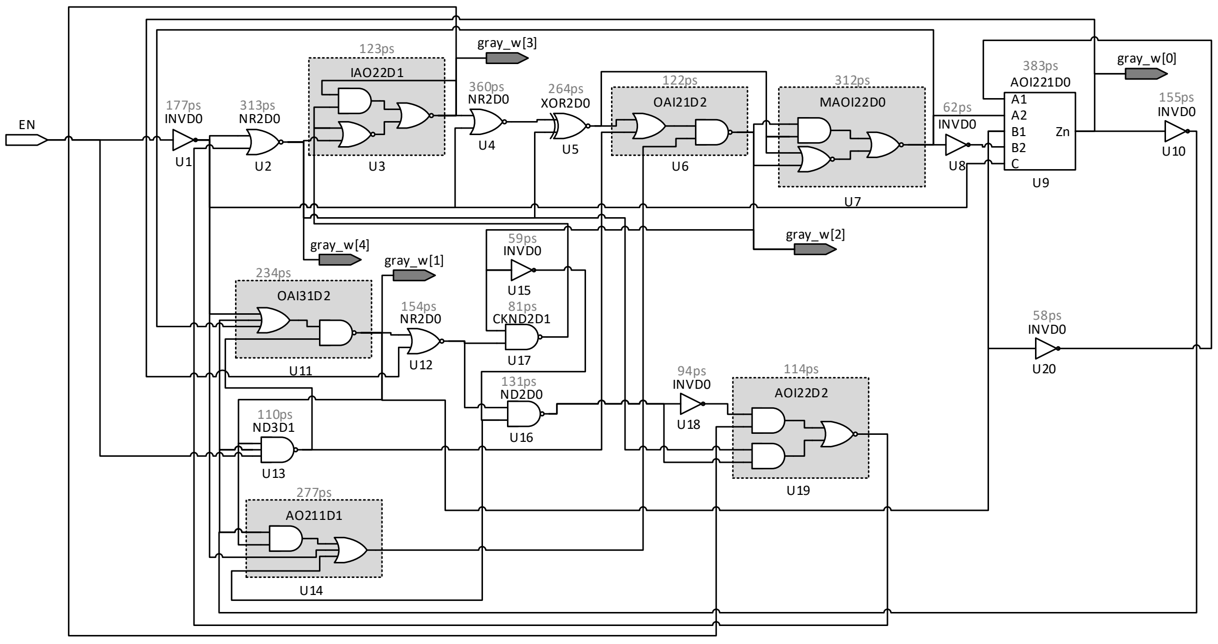


Figure 4.4- Gray-Code TDC Channel Synthesized Netlist Overview

When analyzing the Netlist to determine the estimated propagation delay of each step, it must be considered that, at any time, from one step to the next one, only paths that does not change multiple Gray-code bits must be considered as valid. Using the detailed Netlist schematic of the start channel fine measurement module, presented on Figure 4.4, and the transition from the code 00001 to 00011 as an example, only the paths "AOI221D0-INVD0-OAI31D2" and "AOI221D0-INVD0-ND3D1" are valid. The other ones like, for example, "AOI221D0-NR2D0-CKND2D1-IAO22D1-NR2D0-XOR2D0-MAOI22D0-OAI31D2", does not corresponds to a valid system behavior, since it would generate multiple bit changes on the outputted Gray-code. For brevity, the equations used to calculate the propagation delay of each logic cell (depicted in grey color at the top of each cell in Figure 4.4) are not presented. However, the process is analogous to the one used in the TDL architecture. First, the input and output capacitance for each cell pin is annotated. Then, for each cell, the output load is calculated and the value is used in the propagation delay equation of the cell (obtained from the TSMC standard digital library datasheet). With the propagation delay of each cell calculated (using the typical-case datasheet values), the Netlist paths are analyzed for each step, and the propagation delay of the cells contained in the path are added. It is important to notice that the time of a step is equal to the propagation delay of the bit that changes to the following bit to change, i.e., considering the sequence 00001-00011-00010, the step size of the code 00011 will be equal to the time required for the 4th bit (the one that changed from 00001 to 00011) to generate a change in the 5th bit (the one that changes from 00011 to 00010). Table 4.2 presents all the Gray-code steps delays and corresponding path according to the presented Netlist.

Since the goal is not to obtain a detailed characterization of the TDC channel, but rather an overview of the performance and architecture limitations, although some steps change at the high-to-low transition, the delay calculations were done considering the propagation delay of a low-to-high transition.

Table 4.2- Gray-Code Path Propagation Delay Analysis

Gray-code Step	*Path*	*Propagation Delay (ps)*
0	U1-U9	560
1/9/17/25/5/13/21/29	U10-U11	411
2/10/18/26	U9	383
3/19	U10-U14-U6	554
4/12/20/28	U7-U8-U9	757
6/14/22/30	U20-U9	441
7/23	U12-U17-U3-U4	718
8/24/16	U5-U7-U8-U9	1021
11/27	U10-U13-U6	387
15	U12-U16-U18-U19-U2	806
31	U12-U16-U19-U2	712

The analysis of the results presented on Table 4.2 demonstrate that eleven different paths are responsible for Gray-code changes (three more than in the FPGA case). Moreover, there is a considerable step delay variation, with a maximum variation equal to 638 ps. This value is even higher than the worst case scenario obtained in the FPGA simulation, which already included routings. Another important aspect to highlight is the complexity of the routing that will be required to be managed during the layout step. There are multiple insertion points per step and no routing pattern (when compared to the TDL case where the routing from one step to the next one followed a well-defined pattern).

In conclusion, although performance loss was expected due to the process technology in which the TDC is being fabricated (the same happened in the TDL architecture), the migration of the Gray-code TDC architecture to ASIC introduces multiple other issues that do not exist in FPGA-based designs. The following section compares the preliminary results from the TDL and Gray-code architectures.

4.2.3. Discussion

When analyzing the preliminary results of the implemented TDCs, it is clear that the performance of the Gray-code architecture is deteriorated when migrated to ASIC. This is mainly due to the dissimilarity of the combination logic for each Gray-code bit. While in FPGA-based implementation, a LUT cell with a fixed propagation delay (regardless of the implemented truth table) is available, in ASIC-based implementation logic cells must be used which, depending on the logic equation to implement, may have different levels of combination logic, resulting in different propagation delays. Furthermore, although in the best-case scenario (the one with less combination logic stages), the achieved resolution is circa to the one obtained in FPGA, the worst case path presents a propagation delay in the range of 1 ns. This propagation delay discrepancy between stages also has a negative impact on the overall TDCs linearity, forcing the implementation of an extra calibration mechanism, something that in FPGA-based design was not necessary. One can argue that additional combination logic could be added to the fastest paths in order to achieve a uniform delay in each Gray-code bit calculation. However, this solution would imply a resolution in the range of 1 ns. Please note that one of the requirements of this Thesis was to achieve a resolution and precision under 500 ps, in order to cope with the LiDAR requirements.

Regarding the ASIC-based TDL TDC implementation, due to its structure, a stable step propagation delay is expected across the delay chain. This stability may only be compromised by the parasitic loads introduced by the routing and PVT variations. Since the process technology in use to implement the TDC in ASIC is much bigger than the one used to fabricate the FPGA used during prototyping, the reduced resolution was expected. However, when considering the typical operation scenario for the selected clock buffer, the 105 ps (low-to-high) and 107 ps (high-to-low) transition time obtained are enough to comply with the LiDAR's requirements. The main drawback of the TDL architecture is its larger layout area and power requirements when compared to the Gray-code architecture. Nevertheless, the TDL higher linearity and superior resolution justify its implementation as opposed to the Gray-code. Regarding portability, the TDL forces the editing of the Verilog code block responsible for defining the delay line (which was not required in the Gray-code architecture since it was described in a technology independent fashion). Nevertheless, this change can be scripted using a define statement in the Verilog TDL description. Thus, a full automated migration was still feasible with minimal user intervention.

4.3. Synthesis

The first step in the design flow, is to run synthesis to obtain a technology dependent netlist from the technology independent code. Again, as in the case of the FPGA design flow, the tool must be configured to avoid unwanted optimizations. While in the case of Xilinx tools this could be done inside the Verilog code file, using the *dont_touch* keyword, when using Design Compiler, it is necessary to use a TCL command which can be included in a script that controls the flow of the synthesis process. The TCL command used is the *set_dont_touch* followed by the name of the cells that should not be considered for optimization (as depicted in Figure 4.5). In the ASIC-based design, both the TDL delay elements and the TDL sampling registers were constrained. Otherwise, since the input of the delay chain will always be equal to its output, the synthesis process would remove all the inserted buffers as well as its sampling registers (similarly, to what happens in the FPGA synthesis case). There is no need to constrain the start and stop storing elements since the tool will not optimize them (since the sampling stage registers are not optimized). The remainder of the design constraints, regarding clocks definition for static timing analysis, and input and output ports delays and driving capacitances, are loaded using a Synopsys Design Constraints (SDC) file.

```
set_dont_touch tdl_inst/delay_cell*
set_dont_touch tdl_inst/sample_*
```

Figure 4.5- TCL Command for TDL Optimization Constraint

Since different vendor tools are being used, the information regarding the optimization constraints must be registered and imported to the layout tool. This can be done using the SDC file. Since the synthesis process changes the name references of the design elements, a new SDC file, based on the one uploaded and including the optimization constraints must be generated. This is done using the *write_sdc* TCL command to create the base SDC file and then, the optimization constraints can be appended using the *echo* command with the text *set_dont_touch* followed by the new name of the TDL cells, obtained using the *get_cells* command option, as depicted in Figure 4.6.

```
echo {set_dont_touch [get_cells tdl_inst/delay_cell*]} >> ${SDC_DIR}/${DESIGN_NAME}.sdc
echo {set_dont_touch [get_cells tdl_inst/sample_*]} >> ${SDC_DIR}/${DESIGN_NAME}.sdc
```

Figure 4.6- TCL Command for Exporting the TDL Optimization Constraints

The created synthesis scripted flow goes as follows:

1. First, the target library to use during synthesis is defined (in a *.db* format). The worst-case NLDM timing library information was used during synthesis. Thus, the static timing analysis and circuit optimizations done by the tool ensure proper circuit operation, even in extreme, corner cases, scenarios.
2. After proper library configuration, the directory containing the source files of the design is searched and the design files are loaded and analyzed. After analysis, the design is elaborated, creating a first, still technology independent, Netlist. This view enables to do preliminary checks regarding the developed code. No optimization is performed until this point. Thus, one can verify the direct map between the developed code and the hardware generated.

3. Third, before compiling the design to obtain a technology dependent Netlist, the design constraints must be loaded to secure proper static timing analysis and give additional information to the tool (so that proper buffer insertion, logic duplication and other optimization tasks can be performed correctly). During this step, the mentioned optimization constraints, related with the TDL generation, should be added.
4. To eliminate scenarios where different modules have elements with the same name, the *uniquify* command should be used to solve the design ambiguities.
5. The design compilation is done using the *compile_ultra* command with the flag *-no_autoangroup*. This secures that the design hierarchy is maintained during compilation and that inter-modules optimizations are not performed. Instead, optimization is applied individually to each module as if the module was the only one in the design.
6. With the technology dependent Netlist created, a set of commands is used to change the names of the Netlist elements according to the Verilog naming convention. A second *compile_ultra* command invocation is done with the flags *-incremental* and *-area_high_effort_script* to try to obtain an improved Netlist with lower area consumption.
7. The *check_design* command is launched in order to the tool to validate the Netlist and, if there are no issues, the Netlist is written to a file in a Verilog format.
8. Timing, area and power reports are exported to further analyze the obtained Netlist performance.
9. Finally, the loaded constraints are merged with the TDL optimization constraints and a new standard delay format file (*.sdf*) is written and exported.

As could be seen in Figure 4.3, the Netlist schematic kept the hierarchy defined in the Verilog code, and the fine stage measurement module was not removed by the optimization algorithms, as intended.

4.4. Place and Route (Layout)

Placement and routing have great impact on system's performance, thus, before starting the Layout step, it is important to keep in mind the system's requirements and modules structure. In the proposed architecture, the synchronizer and fine measurement blocks demand for special attention during Layout. The remaining modules of the proposed architecture are typical digital blocks that do not required special consideration. The first aspect to address is the need for multiple reference clock phases, to assure a proper synchronizer block functioning. In FPGA-based development this was achieved using a PLL block.

In ASIC-based development however, since the phase differences that are needed are spaced, and the precision required for the phase generation is relaxed, it is possible to manipulate the clock tree generation to create these phases instead of implementing a PLL, which would increase system's complexity, power consumption and layout area. As discussed on Section 4.2.1, the layout phase will also have great impact on the TDC overall performance and linearity. Thus, mimicking the FPGA dedicated routing paths and placement could result in better TDC performance. However, manually placing and routing a delay line composed by many steps is a monotonous task that requires massive user intervention during the design flow (as in the case of custom cells design). To automate the process, Cadence's Innovus Structured Data Path (SDP) functionality was explored to constraint the fine measurement module placement.

The following sections will introduce the concepts related with Structured Data Path and Clock Tree Configuration, presenting the adopted approach and script files created that need to be loaded and integrated into Innovus design flow to enable the layout process automation.

4.4.1. Structured Data Path (SDP)

According to Cadence's Innovus user guide [4.5], the Structured Data Path (SDP) capabilities allow for better performance, power and layout rea to be attained. This can be done using an SDP TCL script, using the graphical SDP user interface, or by loading an SDP file into Innovus with information regarding the relative placement of the data path to constrain. Cadence advocates that SDP should be used when the design has standard cell columns and rows that need to be aligned, when a performance increase is required, or when time to market does not allow for a full custom design approach. Framing the proposed TDC architecture in this context, the application of the SDP functionality to perform the TDC layout allows the alignment of the multiple TDL steps automatically, without recurring to a custom manual flow.

Using the SDP functionality it is possible to concurrently place standard cells and SDP constrained cells, achieving optimal placement while providing a uniform environment for routing and timing analysis. Furthermore, the SDP ensures uniform routing among the constrained cells, which is a much-appreciated feature when designing TDL, since it guarantees a homogeneous routing parasitic effect along the delay chain, reducing non-linearities and contributing for a uniform step delay. Multiple SDP files can be created in order to experiment different placement topologies and study their effect on routing congestion and timing. Another important capability of SDP is the implementation of high-speed register columns, which help on reducing the clock skew and insertion delay (latency), since the routings between the clock drivers and the registers are shorter and more uniform. This also allows for a more precise clock driver selection because of the known routing and register load. Moreover, electromigration and IR-drop (current(I)*resistance(R) - voltage drop) on power rails can be reduce since the registers are grouped in columns.

However, the usage of the SDP requires detailed design knowledge in order to achieve the best possible results. Since Innovus does not automatically recognize SDP elements, these must be scripted based on naming conventions, respecting the design´s hierarchy and namespaces. Nonetheless, it is still less demanding than the full custom design process typically used to implement TDCs in ASIC.

Typically, the SDP information is added into the layout after importing the design and defining the floorplan (layout area, power planning and Macros placement). After defining the SDP constraints, SDP and standard cell placement can be performed. Optionally, after running the placement command, SDP columns and rows can be edited to explore alternative solutions. Upon conclusion, the typical digital layout flow can be used.

TDL SDP Placement Analysis

Based on the presented SDP capabilities, its application to migrate the TDL enables to address all the concerns identified in Section 4.2. Namely, dedicated and uniform routing and automated layout. Furthermore, this approach enables to mimic the Carry4 structure available in FPGA platforms, which is one of the reasons for the popularity of TDLs in FPGA platforms.

As stated in the previous section, the SDP demands for detailed knowledge of the design being implemented. This includes the dimensions and pin locations of the cells being used. In extreme cases, this may also include cells' blockages. Only with this information can the placement and routing be effectively predicted when structuring the SDP file. The TDL TDC architecture has three different cells instantiations, the clock buffer used as the delay element (CKBD0BWP7T), the sampling flip-flops (DFQD0BWP7T), and the storing flip-flops (EDFKCND0BWP7T). The detailed dimensions and pin locations of these cells are depicted in Figure 4.7. Figure 4.8 depicts a detailed view of two TDL steps.

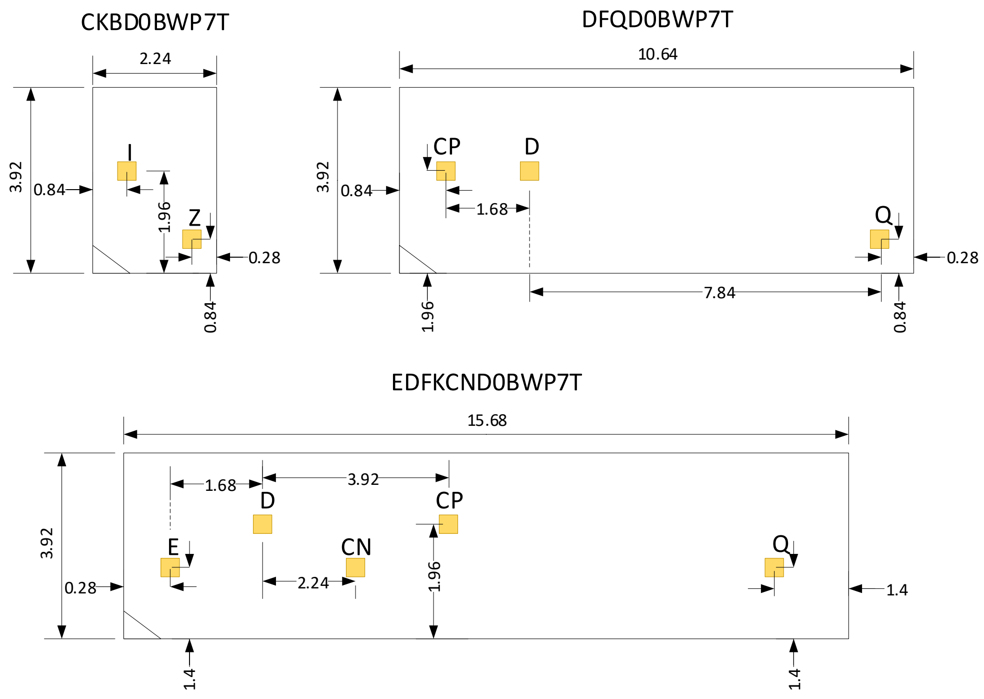


Figure 4.7- TDL's Cells Layout and Dimensions (units in μm)

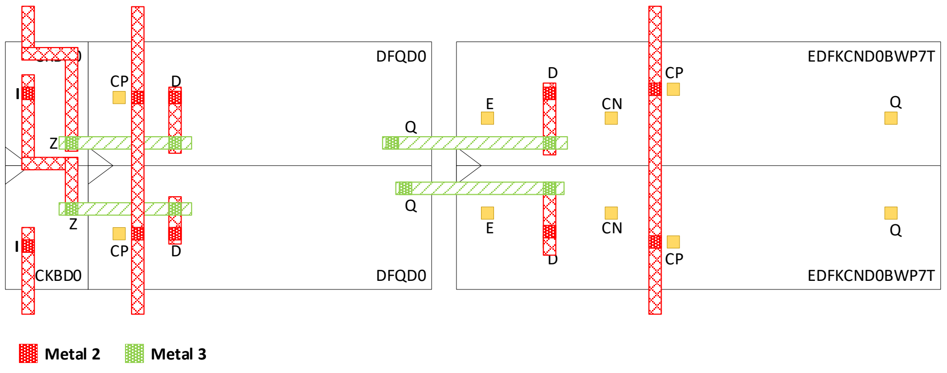


Figure 4.8- Detailed View of Two TDL Steps

Multiple SDP topologies were analyzed in order to find a balance between the TDC channel performance and the overall layout area, power and timing optimization. The first decision is to choose whether the TDL should follow a horizontal or vertical layout. Two drafts were made to study how the TDL disposition would affect the remaining system (see Figure 4.9).

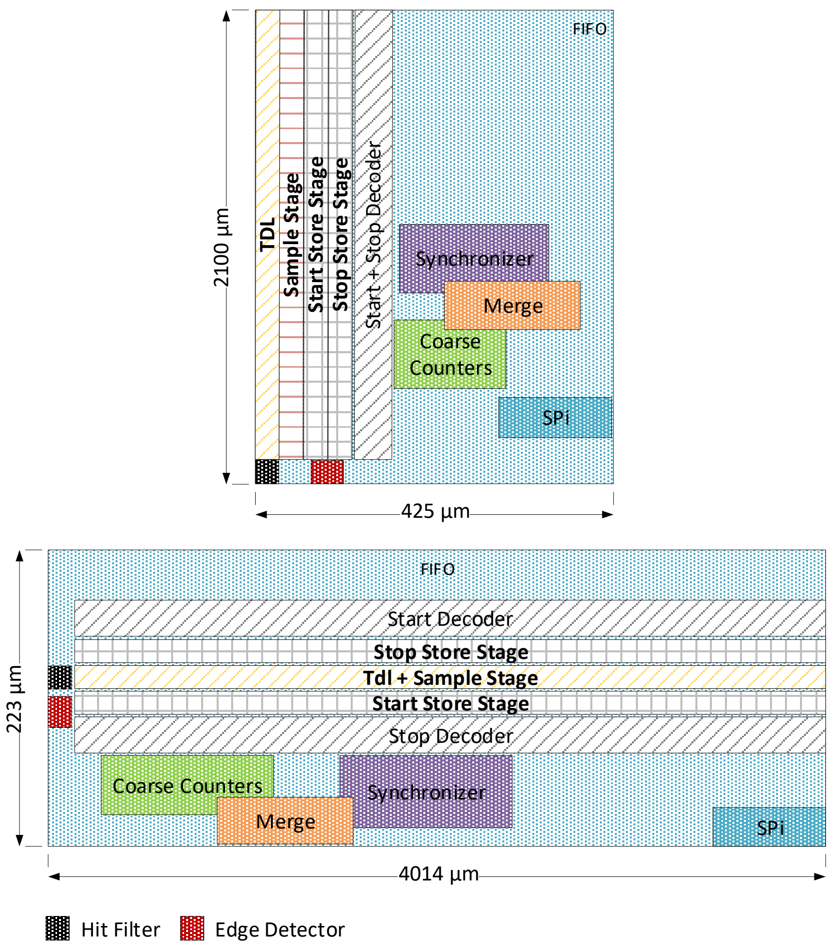


Figure 4.9- Vertical (top) and Horizontal (bottom) Study of the TDL Layout

A pure horizontal TDL would require a core width of 4014 µm, which would result in a poor layout ratio, making it difficult to properly place and route the remaining system and achieve time closure. Moreover, using the horizontal approach it is not possible to design the TDL to be routing and area optimized simultaneously. If uniform routing is targeted (the scenario depicted at the bottom of Figure 4.10), an approximately 72.45 µm^2 per TDL step would be wasted. On the other hand, if area optimization is targeted, the routing between stages would get compromised due to pins being harder to access (depicted at the top of Figure 4.10). Furthermore, both horizontal approaches would compromise the clock signal distribution, resulting in a complex clock tree design. In order for the horizontal approach to be feasible, the TDL would have to be folded. Although the approach seems feasible, considering the used cells, if the folded horizontal approach implementation is analyzed in detail, it is possible to conclude that this solution would increase the routing complexity between the TDL and the decoding stages while sharing the clock distribution issue identified for the non-folded horizontal approach.

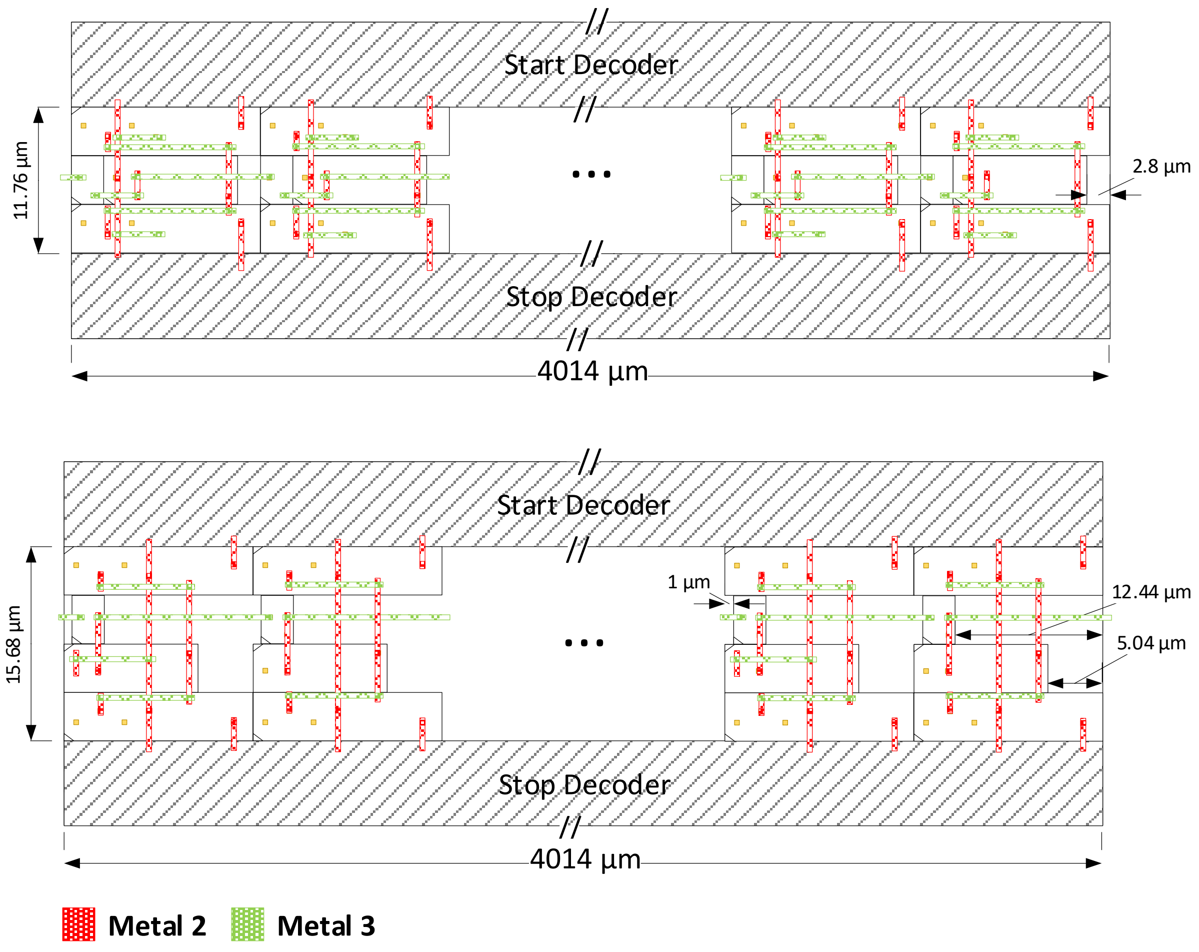


Figure 4.10- Horizontal Layouts Detailed View

The vertical layout offers a more advantageous solution for all scenarios. As in the horizontal approach, the TDL SDP topology can focus routing or area optimization. When targeting routing optimization (depicted on the left of Figure 4.11), the unused area corresponds to 32.928 µm^2 per step (55% less than in the horizontal topology). The vertical layout ratio is also better when targeting routing optimization. A maximum of 2010 µm height is required, if the TDL was not to be folded (50% less than the width required for the horizontal TDL). The other alternative, depicted on the right in Figure 4.11, focusing the area usage optimization, requires a 1050µm height to implement a non-folded version of the TDL, resulting on a 1050*850 µm layout (targeting the routing optimization would result in a 2100x425 µm layout). Both layouts are feasible, however, given the maximum chip size limitation of 2.5 mm for the ASIC fabrication run provided by TSMC, the routing optimized solution would constrain the chip pad ring design.

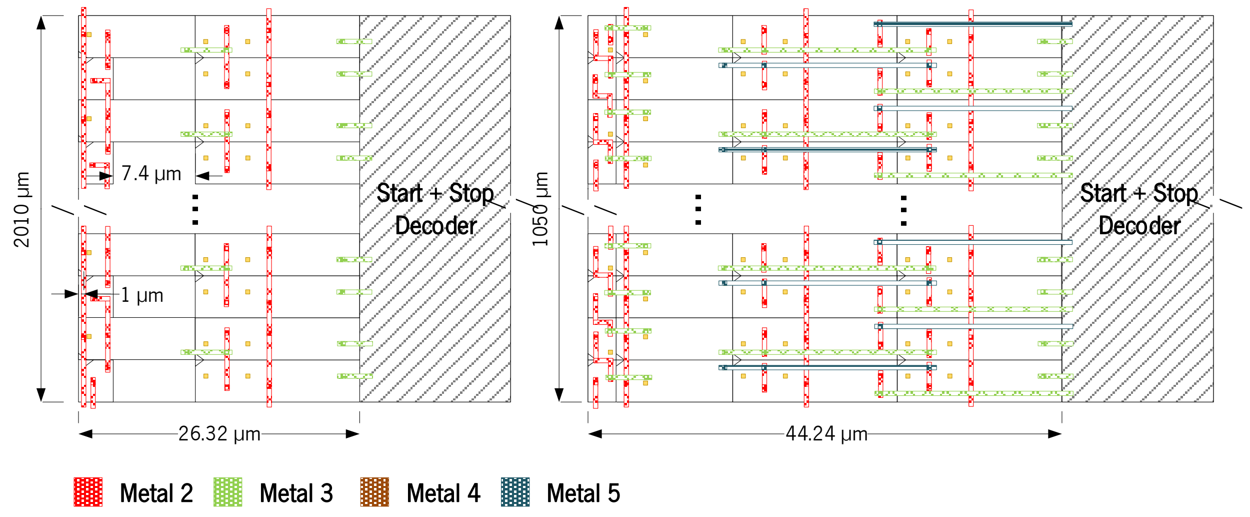


Figure 4.11- Vertical Layouts Detailed View. Optimized for Routing and for Area

Although it is possible to fold the TDL, the routing in the folding point would generate a non-uniformity in the TDL steps, similar to what happens with ultra-wide bins in FPGA platforms when crossing clock domains. Moreover, the folded TDL compromises the decoder blocks performance, which in turn deteriorates some TDL routings (see Figure 4.12). Analyzing the layout area of the optimized SDP solution, it is possible to conclude that the routing patterns that will be generated automatically will be very similar to the ones obtained when targeting routing optimization. Thus, the approach depicted on the right side of Figure 4.11 was selected for SDP implementation. Not only the routes follow a well-defined pattern, since layout ratio is balanced, but also the remaining system and clock tree generation can be easily placed and optimized, enabling power savings and easier time closure. Moreover, there is no need to fold the TDL, therefore the decoder blocks can be placed in a way that enables good performance while not interfering with the TDL routings.

SDP File Structure

In order to implement the defined structure presented on the previous section and to load the SDP placement constraints information to Innovus, a file following a standard structure must be created. This file contains the data paths' relative placement information, supports hierarchical constructs and wildcard for the design's instance names, allow pre-place location, orientation, flipping and alignment constraints to be defined, and supports numerical bus bit range and order sequence as part of an instance name. The final SDP file used to constraint the implemented TDL is depicted in Figure 4.13.

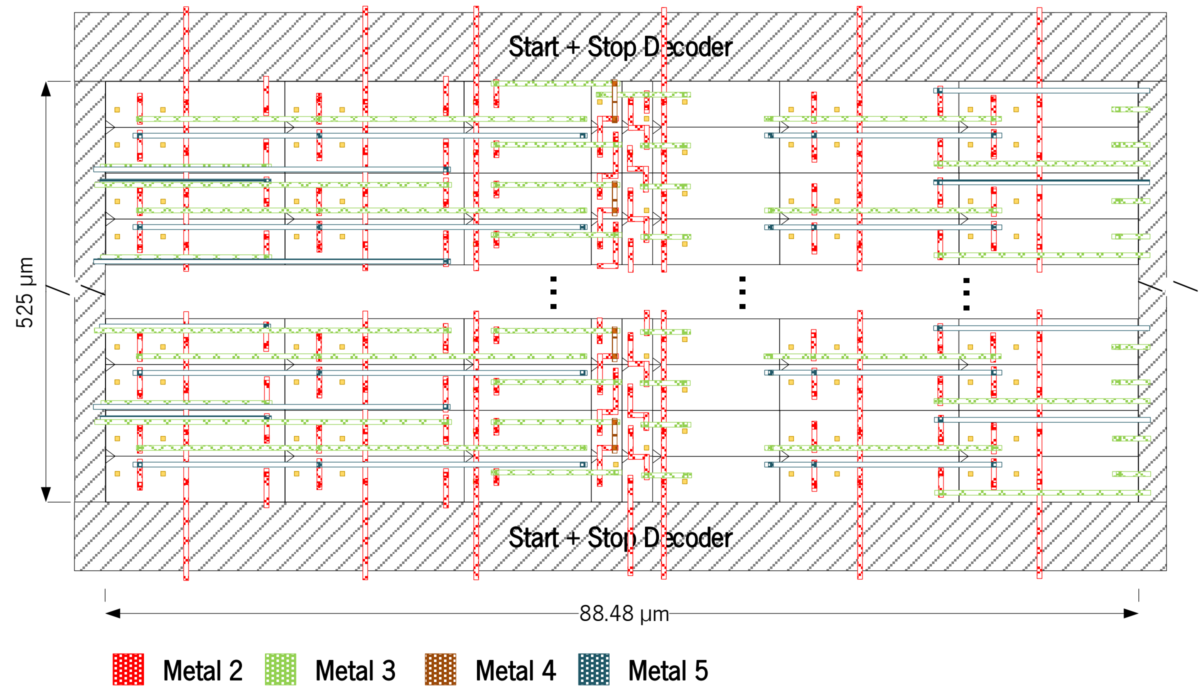
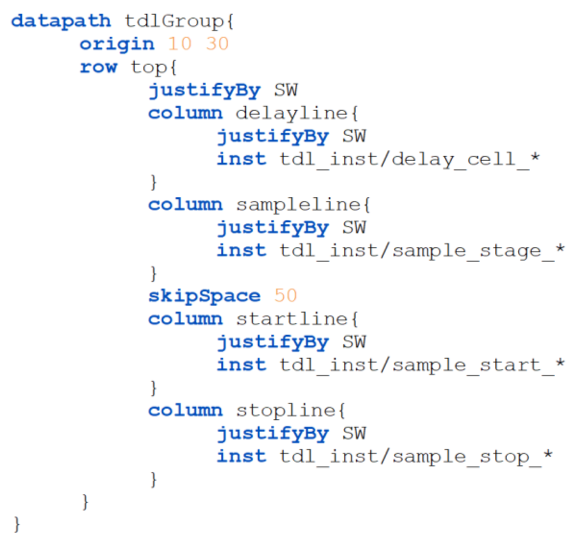


Figure 4.12- Influence of Folding the TDL on the Routing



```
datapath tdlGroup{
      origin 10 30
      row top{
            justifyBy SW
            column delayline{
                  justifyBy SW
                  inst tdl_inst/delay_cell_*
            }
            column sampleline{
                  justifyBy SW
                  inst tdl_inst/sample_stage_*
            }
            skipSpace 50
            column startline{
                  justifyBy SW
                  inst tdl_inst/sample_start_*
            }
            column stopline{
                  justifyBy SW
                  inst tdl_inst/sample_stop_*
            }
      }
}
```

Figure 4.13- Structured Data Path File used to Constrain the TDL Module

A total of sixteen keywords are available when defining an SDP file. In the presented example *datapath*, *row*, and *column* keywords specify the data path structure, row group and column group names respectively. The row and columns names defined must be unique within the same or across different datapath groups. The *origin* keyword is used to define the lower left floorplan coordinate of the data path structure. Design cells are declared using the keyword *inst*. Multiple cells can be specified with a single *inst* keyword using wildcards (as depicted in Figure 4.13, where all the cells with its name starting by *tdl_inst/delay_cell_* will be included in the SDP constrain). Note that the instance name must follow the hierarchy defined in the design to properly identify the targeted cell. The *justifyBy* defines the anchor point used when aligning the SDP group or cell. If the *justifyBy* property is not defined in a group or cell, the value used is the one defined in the hierarchical group immediately above (parent group). Finally, the *skipSpace* keyword defines a space that should be empty. When defined in a column, this value represents the number of rows to skip. When specified in a row, the skipped value is equal to the specified number times the pitch of the first vertical layer, i.e. the number of Metal 2 tracks to skip. Thus, if a Metal 2 track has 1 µm width, and a *skipSpace 10* is specified inside a row, then a 10 µm space between the previously defined row element and the next row element, will be left empty. To specify that the value used in the *skipSpace* definition should be interpreted in microns instead of the number of Metal 2 tracks, the keyword *micron* can be used. If a group of instances needs to be placed with space between them, it is possible to use the *spreadGroup* keyword followed by the value to skip and the cells to instantiate.

It is also possible to specify the orientation of the cells using the *orient* keyword and to align the placement of the cells by pin name using *alignByPinName*. This last keyword allows the router to generate a straight routing path connecting all defined cell instances. Thus, this keyword is typically used when defining high speed register columns, to align their clock pins.

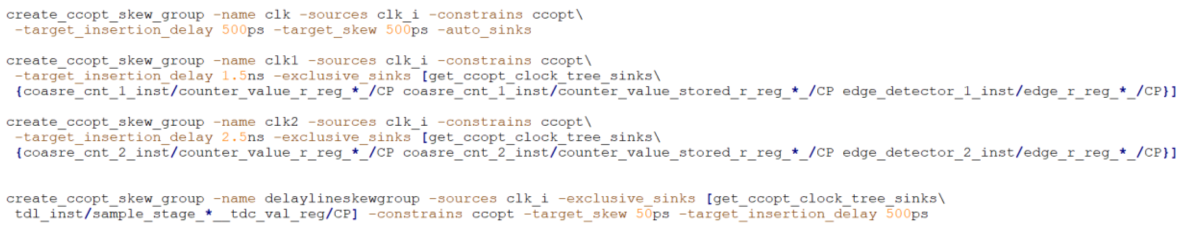
4.4.2. Clock Tree Configuration

Since version 14.2, the clock tree synthesis engine used in Innovus is the Clock Concurrent Optimization (CCOpt) which performs Clock Tree Synthesis (CTS) using timing driven useful skew and Data path optimization simultaneously, as opposed to the CK engine, used in older versions of the software, which performed the traditional global skew balancing based on a FE-CTS specification [4.5].

The new engine enables for smaller and highly efficient clock tree generation based on the timing constraints files. It is also possible to include additional constraints to the clock tree generation using the Innovus' command line. The set of constraints created can then be merged in a clock tree specification file. This file enables the user to analyze an overview of the clock tree prior to its implementation. Moreover, CCOpt flow merges post CTS optimization as part of its final internal optimization process. According to Cadence's Innovus user guide [4.5], to properly operate CCOpt requires high quality multi-mode timing constraints, with clocks configured in ideal clocking mode, i.e. estimated clock network latency from the SDC constraints. The main CCOpt configuration steps are:

- Load post-CTS timing constraints, since the CCOpt flow performs post-CTS optimization these must be defined prior to the call of the TCL command that runs the CTS (typically this step was done only after the CTS was generated and was used to analyze the system with propagated clock timing models instead of the ideal scenario).
- Configure CCOpt mode settings, like the active analysis views.
- Define clock routing types (optional). This step is used to specify routing rules, usually referred as Non-Default Routing rules (NDRs), like different widths and shielding for the clock nets. These rules can be added using a LEF file or the *add_ndr* TCL command.
- Configure the library cells to use on the CTS generation.
- Configure maximum transition and skew target.

With the structure of the TDL defined and the global clock configurations done, before creating a clock tree specification (used by the CCOpt to generate the design's clock tree), further configurations must be done to guarantee the minimum skew between the TDL's sampling flip-flops. The clock tree configuration must also comprise constraints that enable the generation of two different reference clock phases for the synchronizer. This can be achieved by configuring different skew groups and respective insertion delay parameter. First, a global skew group was created, to target all the cells of the design, with a 500 ps insertion delay. In order to create the phased clocks, two skew groups were created with an insertion delay 1.5 ns and 2.5 ns, targeting only the coarse counter 1 and 2, respectively (see Figure 4.14). Thus, a 1 ns phase between clocks was created (equivalent to a 5° and 10° phase regarding the reference clock). For these clocks, no maximum skew was defined, since each group was composed by 16 flip-flops. Therefore, it was predictable that a single clock buffer would be used by the clock tree generator, resulting in a reduced skew value induced by the routing mismatch. These new skew groups constraints override the previously defined global skew group just for the targeted elements. Finally, a fourth skew group, with the same insertion delay of the global skew group is created for the TDL registers. This groups constraints the targeted skew to 50 ps, in order to secure a skew lower than the step delay, eliminating the risk of bubble occurrence on the sampled thermometer code.



```
create_ccopt_skew_group -name clk -sources clk_i -constrains ccopt\
 -target_insertion_delay 500ps -target_skew 500ps -auto_sinks

create_ccopt_skew_group -name clk1 -sources clk_i -constrains ccopt\
 -target_insertion_delay 1.5ns -exclusive_sinks [get_ccopt_clock_tree_sinks\
 {coasre_cnt_1_inst/counter_value_r_reg_*_/CP coasre_cnt_1_inst/counter_value_stored_r_reg_*_/CP edge_detector_1_inst/edge_r_reg_*_/CP}]

create_ccopt_skew_group -name clk2 -sources clk_i -constrains ccopt\
 -target_insertion_delay 2.5ns -exclusive_sinks [get_ccopt_clock_tree_sinks\
 {coasre_cnt_2_inst/counter_value_r_reg_*_/CP coasre_cnt_2_inst/counter_value_stored_r_reg_*_/CP edge_detector_2_inst/edge_r_reg_*_/CP}]


create_ccopt_skew_group -name delaylineskewgroup -sources clk_i -exclusive_sinks [get_ccopt_clock_tree_sinks\
 tdl_inst/sample_stage_*__tdc_val_reg/CP] -constrains ccopt -target_skew 50ps -target_insertion_delay 500ps
```

Figure 4.14- Clock Tree Skew Groups for Phase Generation

The SPI slave clock tree was also constrained using the typical SDC *create_clock* constraints. False paths were also defined in the SDC file to prevent the system from considering different clock domain paths during optimization and time closure. Since the FIFO block was implemented with double register synchronizers for the read and write addresses, a proper data clock domain crossing is always secure. A representation of the Innovus' clock tree debug view for the created clock tree specification file is presented in Figure 4.15.

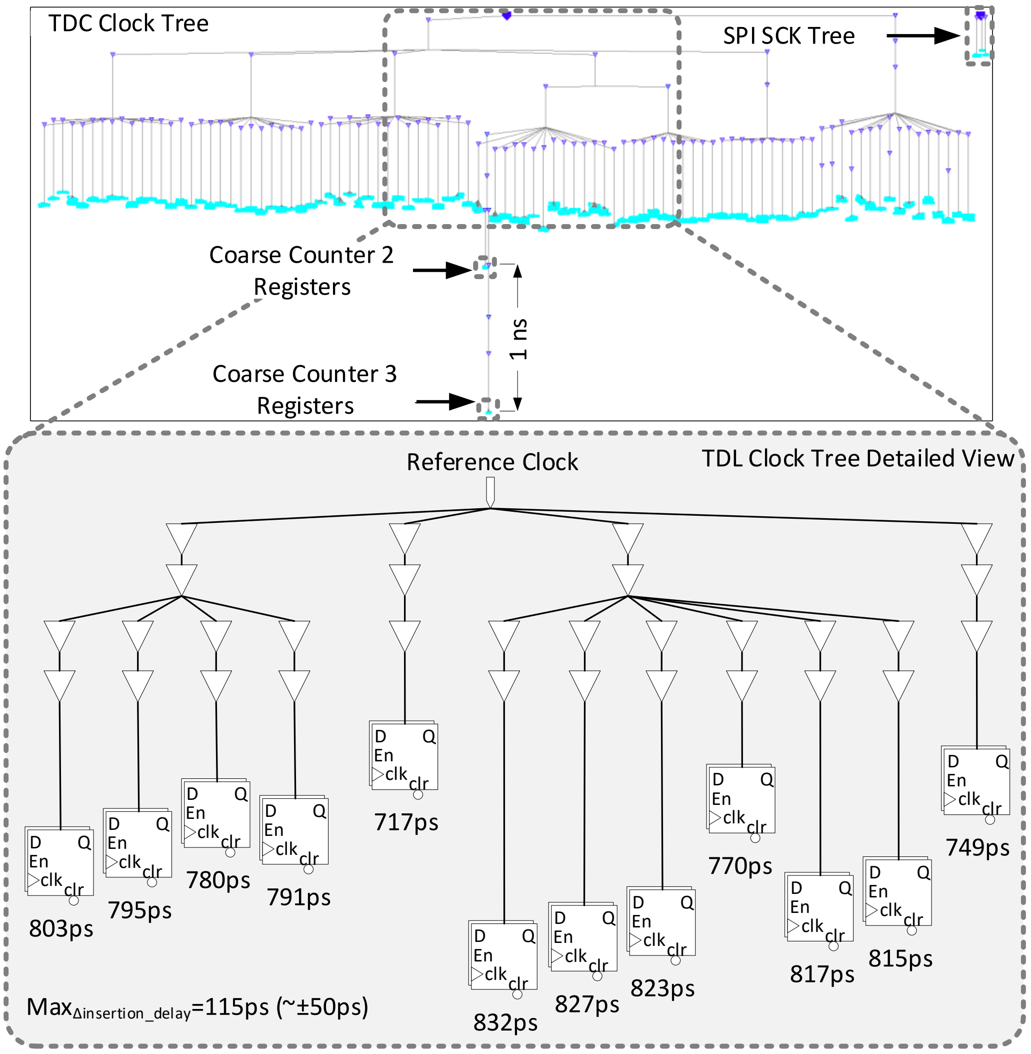

TDC Clock Tree
SPI SCK Tree
Coarse Counter 2 Registers
1 ns
Coarse Counter 3 Registers
Reference Clock
TDL Clock Tree Detailed View
D Q En clk clr
803ps
795ps
780ps
791ps
717ps
832ps
827ps
823ps
770ps
817ps
815ps
749ps
Max$_{\Delta insertion\_delay}$=115ps (~±50ps)

Figure 4.15- Clock Tree Overview

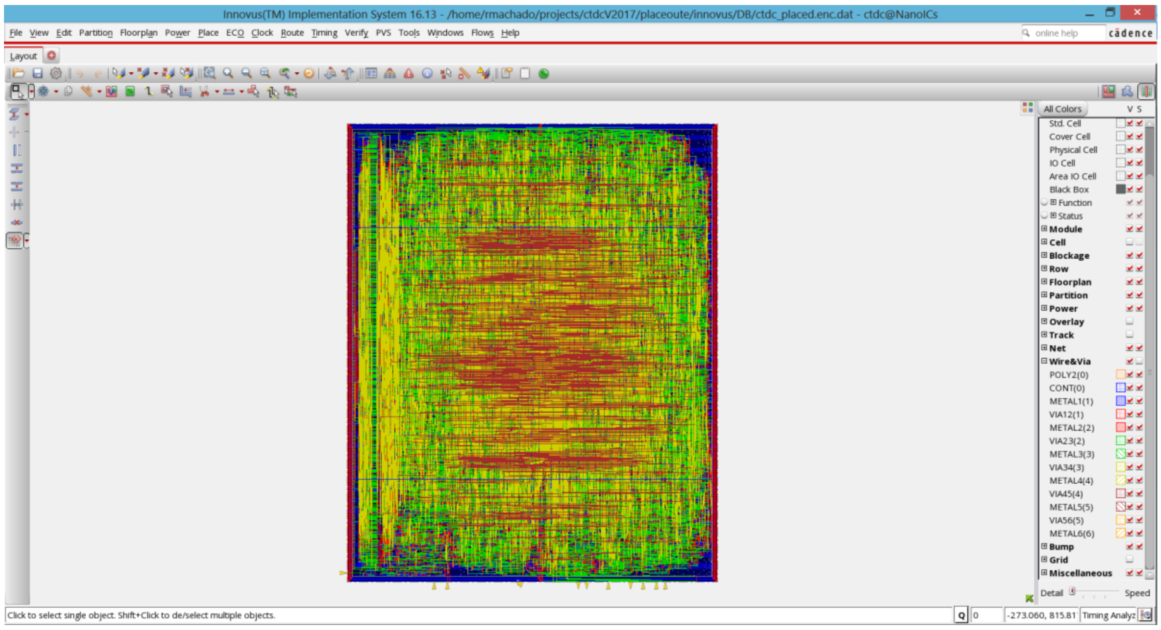
4.4.3. Layout Scripted Flow

Apart from the SDP flow and additional CTS constraints for phase clocks generation, the layout flow used is similar to the typical digital ASIC flow. Moreover, the needed changes were fully scripted, thus an automated process was maintained. The final layout is depicted in Figure 4.16. The created layout scripted flow goes as follows:

1. Since Synopsys Design Compiler was used to get the synthesized Netlist, a configuration step must be done. This comprises the import of the extracted Netlist and SDC from Design Compiler, the specification of the maximum and minimum timing libraries to be used (the same used during synthesis), the specification of the technology LEF file (having the standard digital cells' layout information), the creation of RC corners and the loading of the capacitance tables specification for timing analysis, and the definition of the design analysis views, which aggregate all the specified timing information. Other parameters like the maximum allowed routing layers and the preferred layer for clock routings are also defined during the initial configuration. For this design, only 5 out of the 6 layers available were used, since the last metal layer was reserved for post-layout pad-ring routing. The process technology being targeted should also be defined. Although not necessary, this last configuration automatically sets some tool's parameters specific for that process technology.
2. After successfully importing the design and configuring the tool, floorplan and Macro placement can be performed. The initial dimensions for the floorplan were defined based on the area report results exported during synthesis.
3. Third, the power ring and VDD and GND nets are connected (power planning).
4. With the floorplan and power plan concluded, the SDP constraints are loaded to Innovus using the *readSdpFile* command. The placement mode is also configured using the *-sdpPlace* and *-sdpAlignment* flags set to true, as suggested in the Innovus user guide SDP flow. The placement was also configured to be time driven and to not place I/O pins automatically. Finally, the I/O pin file is loaded (with the floorplan location of eac I/O pin. The placement of the SDP and standard cells was done concurrently by executing the *placeDesign* command.
5. In recent versions of Innovus, global routing was performed simultaneously with the placement activity. This allows for an early analysis on design's routing congestion points. It also allows for early power analysis to check for IR drops. If the results are satisfactory (for routing a <0.05% vertical and <0.1% horizontal routing congestion are desired), the pre-CTS optimization phase can be performed. Otherwise, new floorplan and placement iterations should be done until the obtained results are satisfactory.
6. Afterwards, the clock tree synthesis was performed following the procedure defined in Section 4.4.2. Although CCOpt has already post-CTS optimization integrated, which reclaims design area

and fixes DRC and setup violations, an additional call to optimize the design while correcting hold violations is required. This is done running the *optDesign* command with the *-postCTS* and *-hold* flags.

7. Finished the CTS phase, fillers were added to the design in order to meet metal density rules defined by the foundry and to secure a continuity on the power rails.
8. Finally, the layout is ready to be routed. The detailed explanation of the routing process is out of the scope of this thesis. It is only important to mention that, after the first route iteration (executed using the *routDesign* command), the *optDesign* command with the *-postRoute* option is executed to fix DRC, setup and hold violations. A final routing iteration was executed after the optimization completion.
9. The final layout was checked against connectivity, geometry, and process antenna design rules to verify that there are no DRCs violations.
10. Finally, the layout timing, area, and power reports are generated. The RC parasitic are extracted and annotated in a *.spef* and *.sdf* file formats, and the layout Netlist for post layout simulations was exported. The GDSII layout stream file generation is the last step before completing the layout phase. This stream file exported will be used as an input to Virtuoso, where the chip pad-ring is created, the layout integrated and the final tape-out made.


Innovus(TM) Implementation System 16.13 - /home/rmachado/projects/ctdcV2017/placeoute/innovus/DB/ctdc_placed.enc.dat - ctdc@NanoICs

Click to select single object. Shift+Click to de/select multiple objects.

Figure 4.16 - Post Place & Route TDC layout

4.4.4. Standard Delay Format Results Analysis

The SDF file exported has detailed information on the propagation delays for every connection and cell included in the resultant Layout Netlist. This information can be used to predict and study the delay line linearity. At the beginning of this Chapter, it was stated that, in order to improve the TDL linearity, placement constraints would have to be defined. This was addressed through SDP capabilities. To validate the adopted approach and understand its impact on the TDC channel, a second layout without the SDP flow was created and the results compared. A comparison between the two final layouts is depicted in Figure 4.17. A detailed analysis at the non-SDP layout shows a great discrepancy on the cells positioning as well as non-uniform distance between cells implementing subsequent steps.

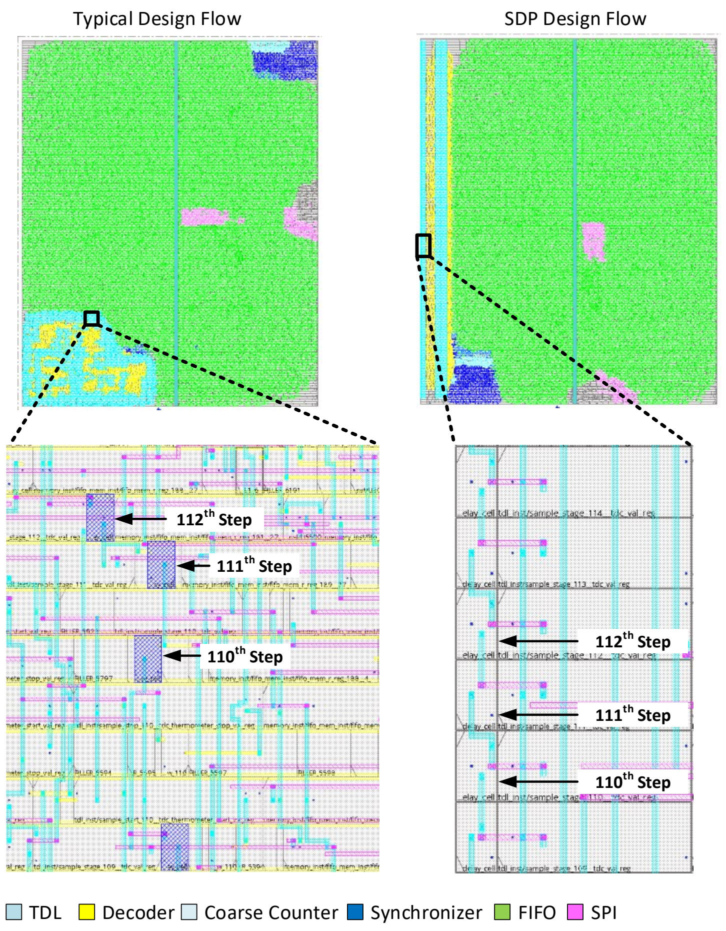
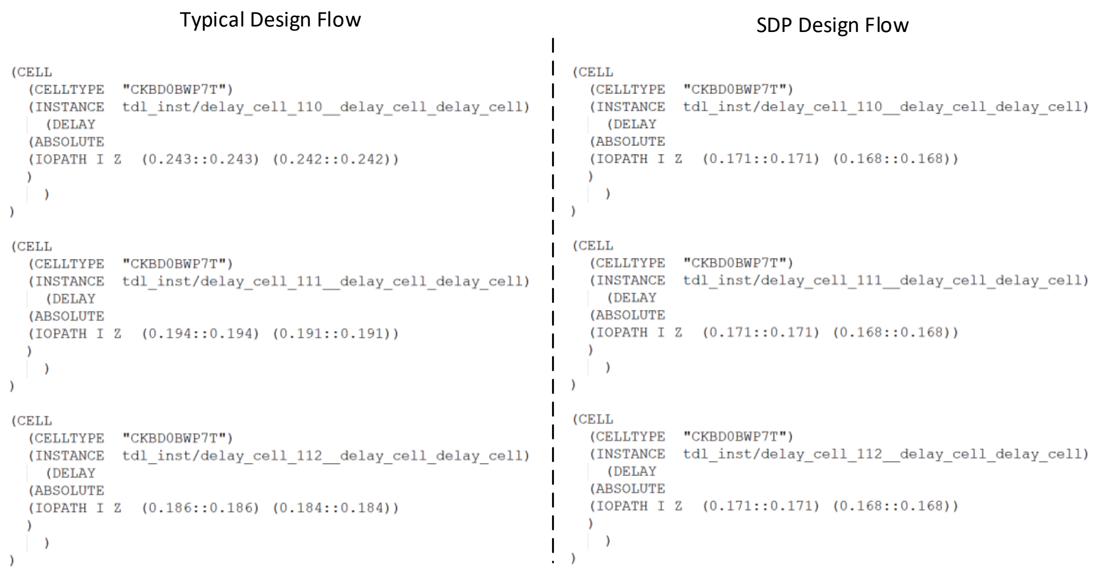


Figure 4.17- Typical and SDP Layout Comparison

The exported SDF files are also a good indicative regarding the effectiveness of the SDP approach. The first thing to notice when analyzing both SDF files is that, although the routings from the two implementations are completely different, the delay introduced in the data path is identical in both scenarios (zero in the SDP case and a maximum of 1 ps in the non-SDP). However, if the cells' propagation delays are compared, it is possible to see that the non-uniform routing generates large variations on the cells' propagation delays (see Figure 4.18). This is mainly due to the discrepancy of the capacitive load introduced by different routings. A comparison between the achieved linearity considering the worst-case scenario for the created layouts is presented in Figure 4.19 (based on the data extracted from the SDF file). The SDP layout presents a maximum variation of 1 ps between steps while the non-SDP approach has a maximum on 77 ps. This discrepancy between steps result in a maximum DNL and INL of 0.38 LSB and 8.58 LSB for the non-SDP implementation respectively. In the case of the SDP layout, the values of the DNL and INL achieve a maximum of 0.038 LSB and 1.1 LSB respectively.



Typical Design Flow

```
(CELL
  (CELLTYPE  "CKBD0BWP7T")
  (INSTANCE  tdl_inst/delay_cell_110__delay_cell_delay_cell)
    (DELAY
  (ABSOLUTE
  (IOPATH I Z  (0.243::0.243) (0.242::0.242))
  )
    )
)

(CELL
  (CELLTYPE  "CKBD0BWP7T")
  (INSTANCE  tdl_inst/delay_cell_111__delay_cell_delay_cell)
    (DELAY
  (ABSOLUTE
  (IOPATH I Z  (0.194::0.194) (0.191::0.191))
  )
    )
)

(CELL
  (CELLTYPE  "CKBD0BWP7T")
  (INSTANCE  tdl_inst/delay_cell_112__delay_cell_delay_cell)
    (DELAY
  (ABSOLUTE
  (IOPATH I Z  (0.186::0.186) (0.184::0.184))
  )
    )
)
```

SDP Design Flow

```
(CELL
  (CELLTYPE  "CKBD0BWP7T")
  (INSTANCE  tdl_inst/delay_cell_110__delay_cell_delay_cell)
    (DELAY
  (ABSOLUTE
  (IOPATH I Z  (0.171::0.171) (0.168::0.168))
  )
    )
)

(CELL
  (CELLTYPE  "CKBD0BWP7T")
  (INSTANCE  tdl_inst/delay_cell_111__delay_cell_delay_cell)
    (DELAY
  (ABSOLUTE
  (IOPATH I Z  (0.171::0.171) (0.168::0.168))
  )
    )
)

(CELL
  (CELLTYPE  "CKBD0BWP7T")
  (INSTANCE  tdl_inst/delay_cell_112__delay_cell_delay_cell)
    (DELAY
  (ABSOLUTE
  (IOPATH I Z  (0.171::0.171) (0.168::0.168))
  )
    )
)
```

Figure 4.18- TDL Typical and SDP Design Flow steps' Propagation Delays comparison

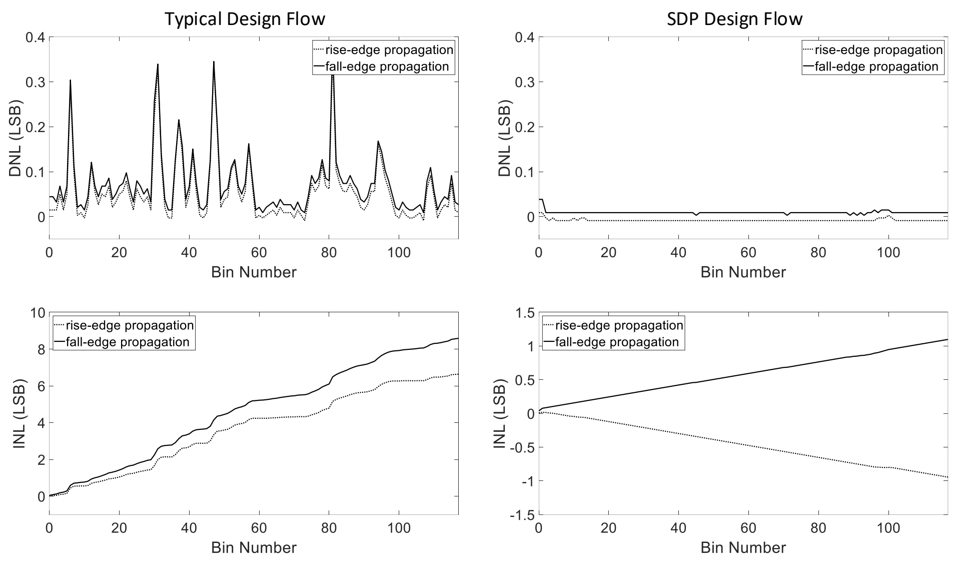
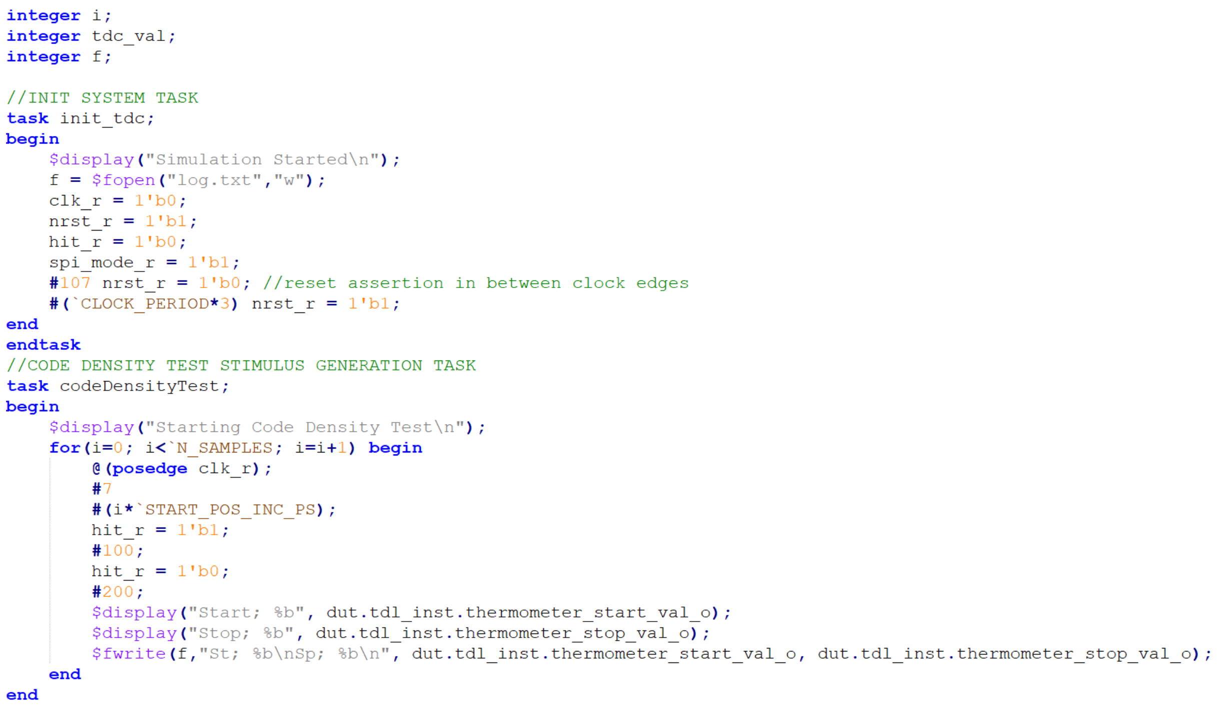


Figure 4.19- TDL Typical and SDP Approach Non-Linearity Comparison

4.5. Simulation Results

According to the metrics presented on Chapter 2, the code density test is of particular interest to determine a good approximation to the steps' real delay and, subsequently, the TDL linearity behavior. Thus, in order to validate and test the implemented layout, a testbench simulating a code density test was implemented (see Figure 4.20).



```
integer i;
integer tdc_val;
integer f;

//INIT SYSTEM TASK
task init_tdc;
begin
    $display("Simulation Started\n");
    f = $fopen("log.txt","w");
    clk_r = 1'b0;
    nrst_r = 1'b1;
    hit_r = 1'b0;
    spi_mode_r = 1'b1;
    #107 nrst_r = 1'b0; //reset assertion in between clock edges
    #(`CLOCK_PERIOD*3) nrst_r = 1'b1;
end
endtask
//CODE DENSITY TEST STIMULUS GENERATION TASK
task codeDensityTest;
begin
    $display("Starting Code Density Test\n");
    for(i=0; i<`N_SAMPLES; i=i+1) begin
        @(posedge clk_r);
        #7
        #(i*`START_POS_INC_PS);
        hit_r = 1'b1;
        #100;
        hit_r = 1'b0;
        #200;
        $display("Start; %b", dut.tdl_inst.thermometer_start_val_o);
        $display("Stop; %b", dut.tdl_inst.thermometer_stop_val_o);
        $fwrite(f,"St; %b\nSp; %b\n", dut.tdl_inst.thermometer_start_val_o, dut.tdl_inst.thermometer_stop_val_o);
    end
end
```

Figure 4.20- Code Density Test Testbench

The testbench generates a fixed length pulse with 100 ns. To ensure a sliding window effect, the stimulus generation is delayed by *i*START_POS_INC_PS* after the reference clock positive edge has been detected. *START_POS_INC_PS* is a constant used to control the time shift increment (in picoseconds) that is made from one iteration to the next, while *i* represents the iteration number. A 40 ps time increment was used on the simulation results presented in this Thesis. The number of pulses to generate was defined by *N_SAMPLES*. Since a total of 100 thousand samples is planned for the experimental tests to reduce probabilistic errors, the same value was used in the testbench for coherency. As the objective of the testbench is to validate the system behavior and study the linearity of the implemented TDL, after each pulse, the value on the second stage start and stop sampling registers were written to a file for further analysis using a MATLAB script. A SPI master read request was also simulated using the task *spim_rd* to validate the SPI interface operation (not depicted in Figure 4.20). The post-layout timing simulation waveforms are similar to the ones presented on Chapter 3 on Figure 3.24. The MATLAB analysis of the obtained simulated code density test are depicted in Figure 4.21.

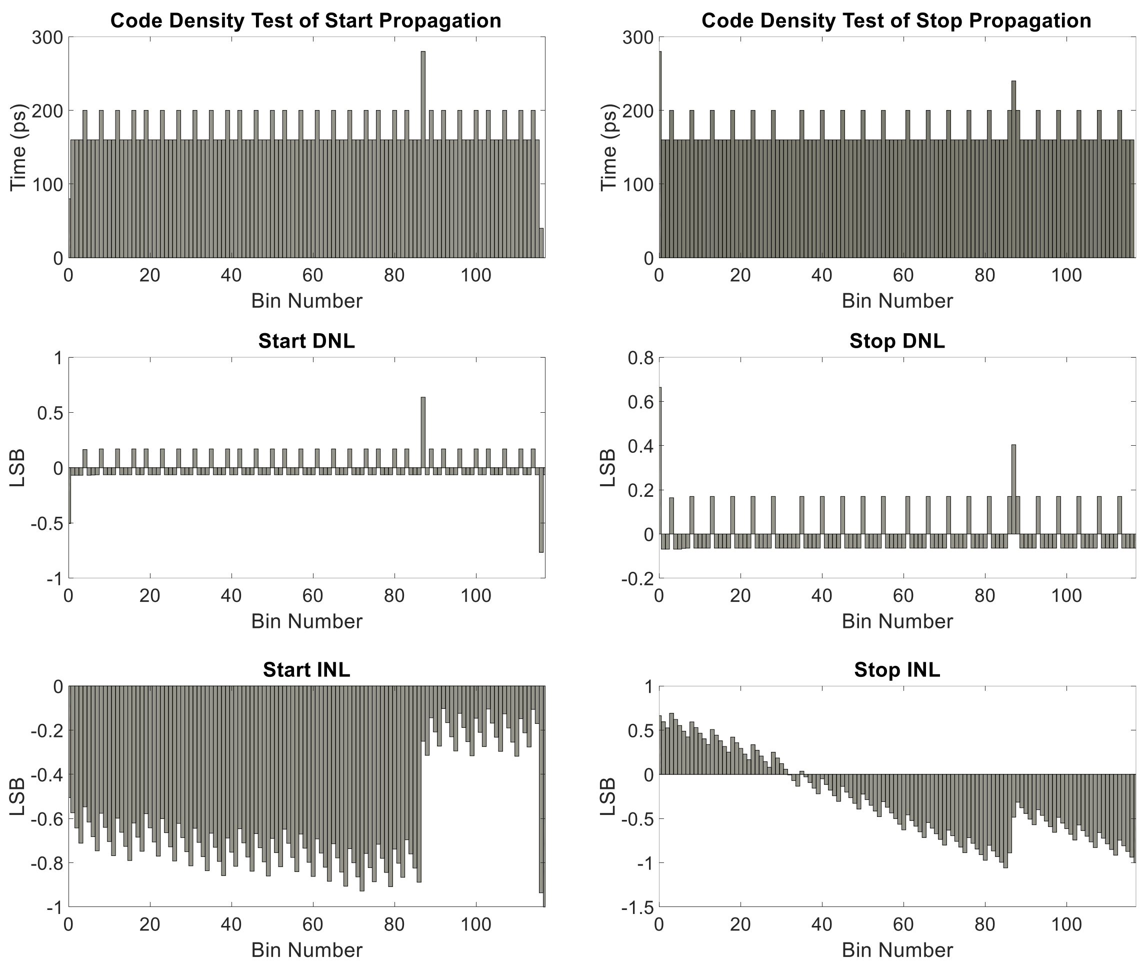


Figure 4.21- Worst-Case Post-Layout Timing Simulation Linearity Results

As expected, the start and stop step delays present similar behavior, with most of the steps presenting a 160 ps propagation delay. However, there are some cells with a 200 ps delay. Considering the results obtained from the SDF file and CTS, this discrepancy is probably due to the ±50 ps skew between the sampling registers. Furthermore, since the hit signal is completely asynchronous to the TDL sampling process, metastability scenarios are frequently detected in simulation, resulting on sampled values with multiple 'X's. This makes it difficult to precisely determine the last step to be sampled and could also justify this discrepancy. It was decided that the 'X' values on the thermometer code would be considered as '1's for the analysis. The simulated results were obtained from the worst-case timing model, thus only 117 steps are considered. In the ASIC experimental tests, a typical behavior is expected, thus, an average delay of 105-107 ps per step should be verified. This represents a total of approximately 188 steps to cover the entire reference clock period. Therefore, although these simulations are helpful to predict some delay discrepancies on the implemented ASIC, it must be stated that the experimental result are expected to be considerably different. Nevertheless, from the analysis of the code density test, a group of slower steps is expected around the 87th step (the peak existing on both start and stop code density tests).

Regarding Linearity, the DNL values are in the range of [-0.1:0.7] LSB, and the INL, for the worst-case scenario verified when propagating the start signal, in the range of [-0.7:0.9] LSB (considering a 170.9 ps LSB). Again, these results are only approximations. Nonetheless, a DNL in the range of ±0.2 LSB is expected for most steps in the implemented TDC, as showed by the simulated results.

4.6. Tape-out

Since the obtained performance results from post-layout simulations were satisfactory, the last step before tape-out is to build the pad-ring, integrate it with the TDC core layout (depicted on the right side of Figure 4.17), and perform final verification, namely LVS and DRC. For the digital core input and output pins the PDUW0408CDG pads were used. The schematic of the pad is presented in Figure 4.22, and the pad signals for input and output configuration is displayed in Table 4.3. The final layout checks were done using Assura.

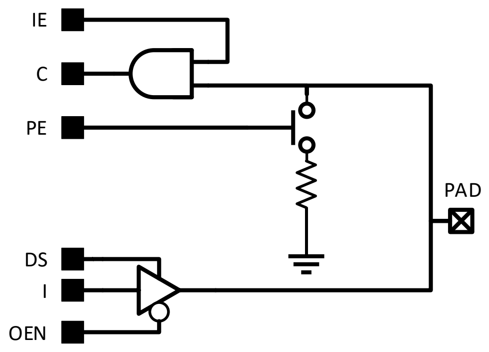


Figure 4.22- I/O PAD Schematic

Table 4.3- I/O PAD Configuration

Configuration	*DS*	*OEN*	*I*	*PE*	*IE*	
Input	X	1	X	X	1	C = PAD
Output	X	0	0/1	X	0	PAD = I

The final layout, ready for tape-out, and the fabricated TDC chip are depicted in Figure 4.23. In Figure 4.23, the final TDC layout used during tape-out is depicted in a), while b) depicts the fabricated ASIC. The numbers 1, 2 and 3 stands for the TDC IP, an independent slave I2C IP and a slave SPI IP (also implemented), respectively. In c) a microscopy photography of the ASIC is presented. The last metal layer was used to cover the chip rendering it impossible to see the other metal layers' details. Finally, the ASIC bounded to its carrier socket is depicted in d).

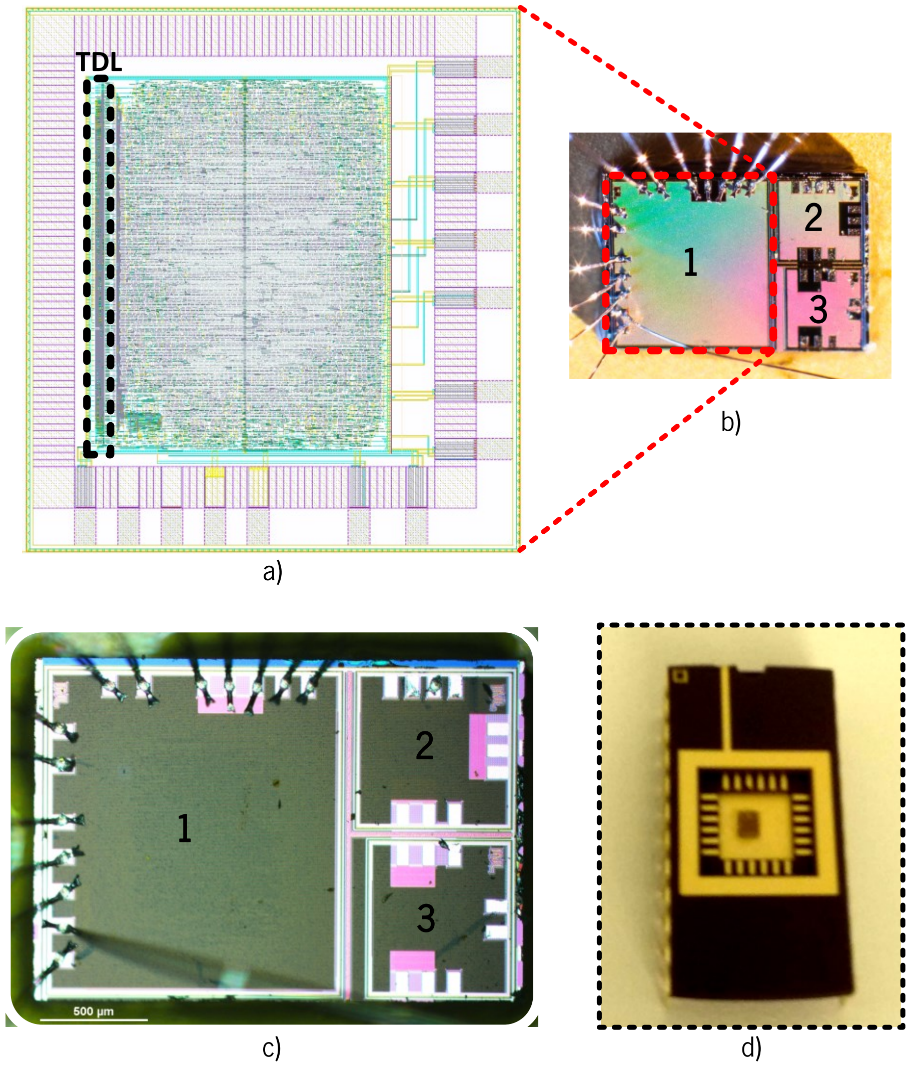


Figure 4.23 - Different views of the fabricated ASIC

4.7. Conclusion

The digital ASIC design, although being capable of a full automated flow, usually requires multiple iterations of the same step to improve the system's performance and optimize its layout area, power consumption and timing results. Multiple vendors offer a complete design environment covering all the digital design phases, with multiple automation options accessible through graphical interfaces or TCL console commands. However, the traditional approach is to adopt a multivendor flow, taking advantage of the best tools from each vendor. Generally, for synthesis, the Synopsys' Design Compiler offers the best results, while Cadence's Innovus is often the option for layout implementation. For time analysis, Synopsys' Primetime typically is selected, while mixed-signal integration and tape-out is usually done using Cadence's Virtuoso.

When adopting a multivendor development flow, multiple files need to be exchanged between tools. Moreover, tool configuration must be done multiple times with different files. In order to improve the design flow and secure proper tool configuration in every digital design, a script was implemented to automatically create the digital project hierarchy, the scripts defining the design flow, and the configuration files for the different tools. Thus, the user must only provide the digital design top module name and the needed design files will automatically be created. Then, only project specific parameters need to be edited on the generated scripts, like the size of the layout, and if SDP flow is to be included or not.

During this chapter, a multivendor design flow, used to migrate and implement the developed FPGA-based TDL prototype to ASIC was presented. The main digital ASIC design flow was analyzed, and the required adaptations discussed. Emphasis was given to the layout implementation, since this phase has great impact on the final TDC channel linearity and performance. The SDP flow was explained and multiple TDL layouts discussed. The CTS process using the new Innovus CCOpt engine was presented, detailing the creation of skew groups to generate different clock phases for proper synchronizer block functioning. The SDP effect on the design was studied and the extracted SDF timing information used to predict the implemented TDC performance. Post-layout timing simulations were performed and presented to validate the system functionality. According to the obtained results, the worst-case scenario resolution is approximately 169 ps. However, since the calculation performed for the typical operation showed a cell propagation delay of 105-107 ps, a resolution within ±10% of this value is expected. Regarding non-linearity, a maximum DNL of 0.7 LSB was verified, while the peak-to-peak INL was 1.6 LSB. A similar linearity performance is expected in the typical operation scenario. Finally, the chapter ends by presenting the result of the integration between the developed TDC core and the pad-ring, including an explanation on the pad-ring configuration.

References

[4.1] Synopsys, "DC Ultra." Synopsys, 2018. [Online]. Available: https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/dc-ultra-ds.pdf.

[4.2] Synopsys, "Design Compiler Graphical." Synopsys, 2014. [Online]. Available: https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/dc-graphical-ds.pdf.

[4.3] Synopsys, “Design Compiler NXT.” Synopsys, 2018. [Online]. Available: https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/design-compiler-nxt-ds.pdf.

[4.4] Cadence, “Innovus Implementation System.” Cadence, 2019. [Online]. Available: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/innovus-implementation-system-ds.pdf.

[4.5] Cadence, “Innovus User Guide,” 2016.

[4.6] Cadence, “SimVision Debug,” 2019. [Online]. Available: https://www.cadence.com/en_US/home/tools/system-design-and-verification/debug-analysis/simvision-debug.html. [Accessed: 26-Nov-2019].

[4.7] Cadence, “Virtuoso Analog Design Environment Family.” Cadence, 2014. [Online]. Available: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/custom-ic-analog-rf-design/virtuoso-analog-design-fam-ds.pdf.

5. Experimental Results

A promising approach for a FPGA-based TDC architecture migration to ASIC platforms was proposed in this Thesis. According to the simulated results, linearity deviations under 1LSB are expected across the implemented delay line. The proposed SDP approach enabled a structured cell placement, which enables an uniform routing between steps and consequently parasitic load homogenization. This resulted in a buffer delay variation of 1 ps across the entire delay line. However, the sampling flip-flops clock skew also influences the step delay. Since the clock tree is automatically generated by the CTS engine, it can only be partially controlled through clock skew group constraints. Thus, CTS will be the major contributor for the implemented TDC non-linearities.

In this chapter the fabricated readout was experimentally tested. In Figure 5.1, the readout carrier board developed is presented in a), while the detailed view of the bounded ASIC TDC on the chip carrier is presented in b). The final readout system with the ASIC TDC, a microprocessor, a temperature sensor and a reference TDC integrated is depicted in c).

The main focus of the research work described in this chapter was to experimentally validate the proposed approach and identify its shortcomings. Nonetheless, a performance evaluation of the designed ASIC, in terms of resolution, linearity, precision and thermal stability is also presented. The main experimental results include short- and long-range measurement precision. The thermal stability of the TDC was investigated in a range between 0 and 50 degrees Celsius. Some layout considerations and architectural changes are discussed, to better understand the obtained results and how to improve the fabricated TDC performance. To reduce the effect of non-linearities and increase the TDC's performance, a software calibration mechanism is investigated. The software calibrated TDC thermal stability was tested in the same temperature range mentioned previously. The chapter ends with some reflections summarizing how to improve the proposed architecture and correct its current issues.

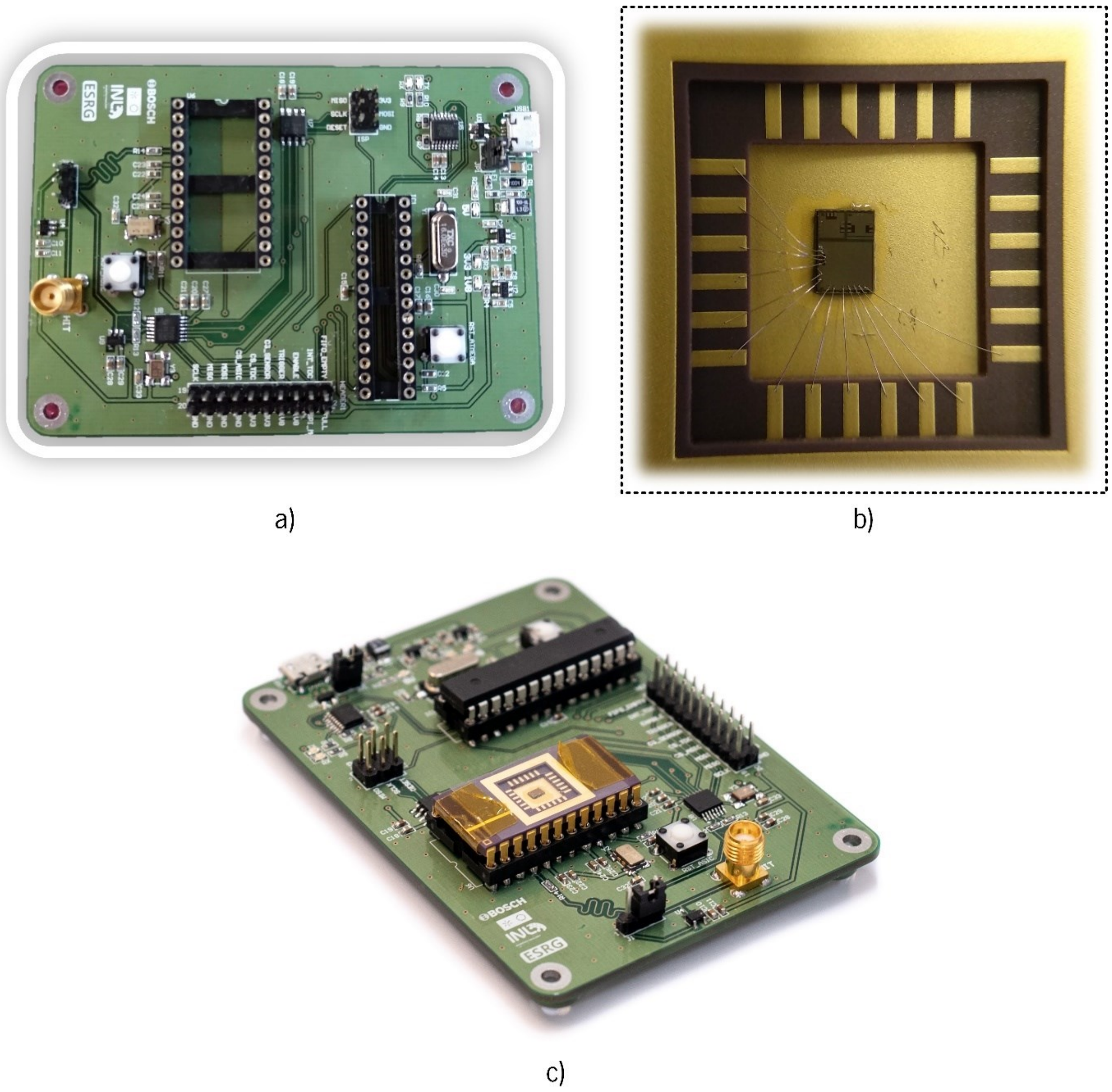

a)
b)
c)

Figure 5.1- Fabricated Readout System

5.1. Measurement Setup

The measurement setup used to assess the TDC performance is depicted in Figure 5.2 (on the left side). The setup is composed by a Tektronix AFG1022 arbitrary function generator, a PCB developed to integrate the fabricated ASIC TDC with a microprocessor, and a host computer running a MATLAB script for data analysis and display. During the temperature tests, the PCB was placed inside a DY60T hoven from ACS (see Figure 5.2 on the right), with the USB and input signal cables being passed through a hole at the side of the oven.

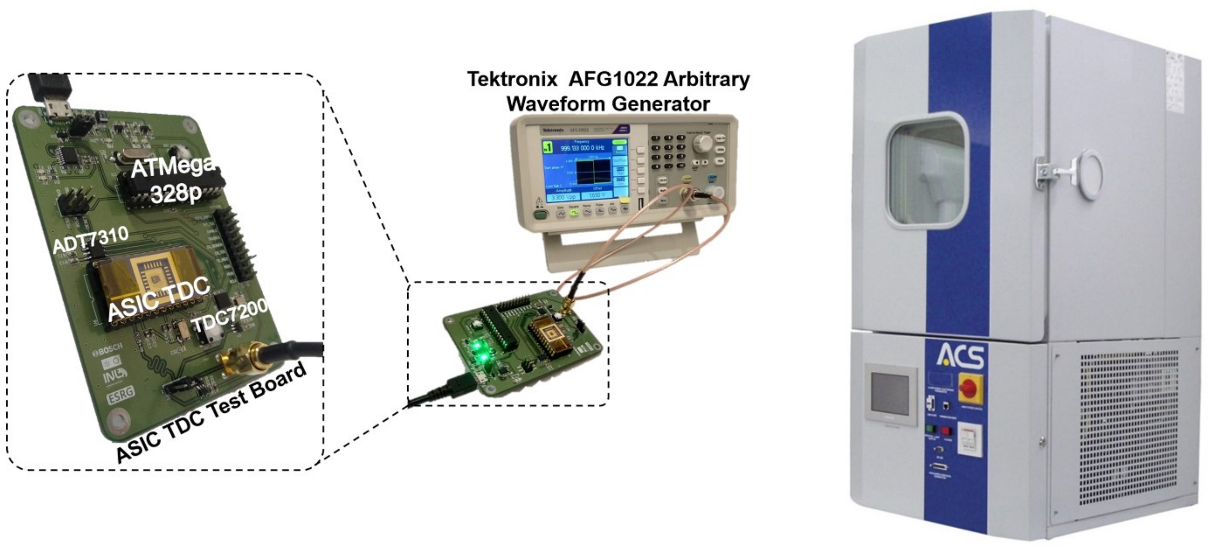


Figure 5.2- Measurement Setup and Hoven used during Temperature tests

The ASIC TDC Test Board (PCB) integrates a Microchip (former Atmel) microcontroller, configured to perform the bridge between the TDC and the host computer. The SPI master interface from the microcontroller was configured to read from the ASIC TDC, rearrange the data to the format expected by the MATLAB script, and send it through serial port to a host PC. The microcontroller is also used in latter tests to implement a software calibration algorithm for the TDC measured values. In order to validate the measures performed by the TDC, a reference TDC from Texas Instruments, the TDC7200, was used. This TDC as a resolution of 55 ps, 35 ps precision, and in mode 1 can measure time intervals in the range of 12 to 500 ns [5.1]. This reference TDC was used mainly to validate synchronization errors that might have not been corrected by the implemented synchronizer block. The temperature sensor, an ADT7310 [5.2], was used during the thermal tests to monitor the PCB temperature. Two external crystals, a 50 MHz one for the TDC and a 16 MHz for the microcontroller are also included along with voltage regulators, to secure a stable power supply for the ASIC TDC. A jumper was included at the input of the TDC to select between the original signal, connected to the board using the SMA connector, and the output of a Schmitt-trigger, used as a filter, to guarantee proper transitions when measuring a noisy or low slew rate signal.

At the time the tests were performed, only the Tektronix AFG1022 arbitrary function generator was available. The function generator enables the generation of a pulse wave with a duration in the range of 40 ns to 999 s and a frequency in the range of 1 mHz to 12.5 MHz [5.3]. However, the <12 ns edge transition time and <1 ns (rms) of typical jitter are far from the ideal values when testing a device capable of measuring picosecond differences. Thus, although the code density test and linearity analysis performed might not get compromised, due to its probabilistic nature (in which case the 100 thousand samples measured help on minimizing these issues), the obtained single-shot precision values were negatively affected by function generator's non-ideal characteristics.

5.2. TDC Characterization

The TDC was characterized according to the metrics defined in Chapter 2. Thus, a code density test was performed, and the values recorded used to study the real steps' delays across the implemented TDL. The average step delay was calculated and the linearity of the TDC analyzed considering the calculated average step delay as the LSB. The precision tests were performed for two different time intervals, a short-range interval of approximately 480 ns, and a long-range time interval of approximately 1.101.321 ns. Single-shot measurement values are presented (no average was applied). As explained in Chapter 3, the TDC conversion time is dependent on the time interval being measured, however, after the occurrence of a stop event, a fixed number of extra clock cycles are required to secure a stable value on the output of the decoder blocks. A three cycle clock delay was defined on the ASIC implementation. After getting a stable value on the decoder blocks, an extra clock cycle is required to merge an write the values to the FIFO memory. Thus, a total of four clock cycles are required until the measurement value is ready. Since a 50 MHz reference clock is being used, the TDC has an 80 ns deadtime in-between measures, resulting in a maximum of 12.5 MHz sampling rate.

5.2.1. Code Density Test

The code density test was performed using the same configuration as in the FPGA performance analysis, i.e. the function generator was configured to output a square wave with 999,133 kHz, and 100 thousand samples were captured. The results representing the TDL's steps delays when propagating a start and stop event are depicted in Figure 5.3.

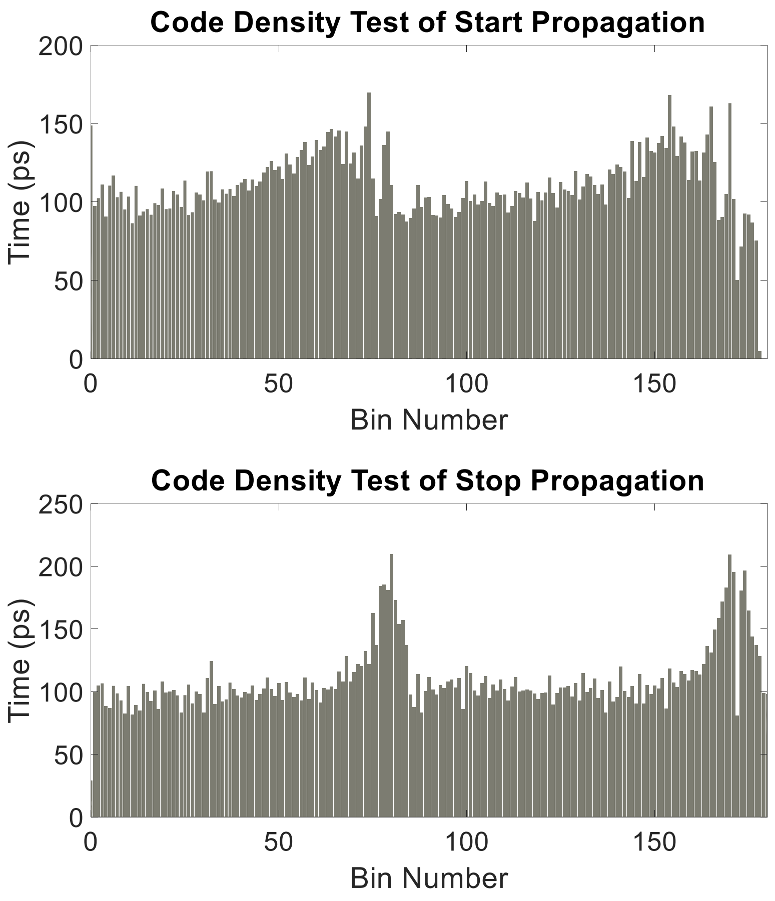


Figure 5.3- ASIC TDC Code Density Test Results

From the analysis of the code density test, if the reference clock period is divided by the maximum number of steps traversed, an average step delay of 112.3 ps and 111.1 ps, for start and stop events propagation respectively, can be obtained. These values are close to the theoretical 105 ps and 107 ps values obtained from the TSMC datasheet equations. Also, a peak on the steps propagation delay can be observed around the 80th-90th step, which was expected due to the results of the worst-case scenario post-layout timing simulation. This is even more noticeable on the case of the stop event propagation. Since the propagation delay when considering the worst-case scenario gives an average propagation delay of approximately 170.9 ps, the experimental values can only be compared to the simulated ones until the 117th step. However, another peak on the steps' propagation delay can be observed around the 160th step for both start and stop events propagation. These variations are related with the TDL sampling clock skew. A detailed analysis on these linearity deviations is presented in Section 5.4. Moreover, no zero delay steps exist, validating the minimal skew approach adopted during CTS.

5.2.2. Linearity

The DNL and INL results for start and stop propagation events are presented in Figure 5.4. Overall, and considering a 111 ps LSB size, the TDC step variation is within ±0.2 LSB for both start and stop propagation, with some outliers reaching 0.5 LSB and a peak non-linearity of 0.9 LSB. Regarding INL, a variation in the range of -2.7 LSB to 3.9 LSB was verified on the propagation of the start event, while in the case of the stop event, a -8 LSB peak was obtained. Based on these results, a considerable increase in the TDC performance is expected if bin-by-bin calibration is applied. A study on the influence of this calibration method on the TDC performance is presented in Section 5.4.2.

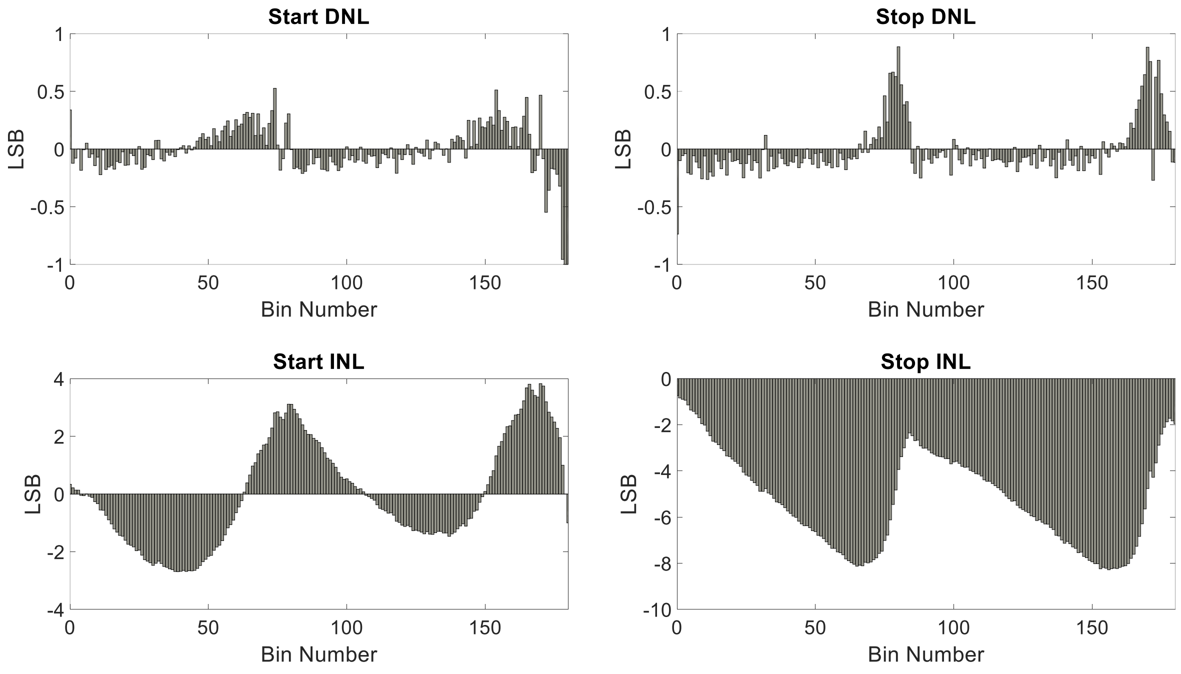


Figure 5.4- ASIC TDC Linearity Results for Start and Stop Propagation

5.2.3. Precision

Apart from the TDL linearity, when combining a fine measurement method with a coarse measurement to increase the system's dynamic range, the reference clock non-ideal characteristics also interfere with the measurement precision. Thus, the TDC's single-shot precision was studied for both short- and long-range measurement scenarios, with the test results being presented in Figure 5.5.

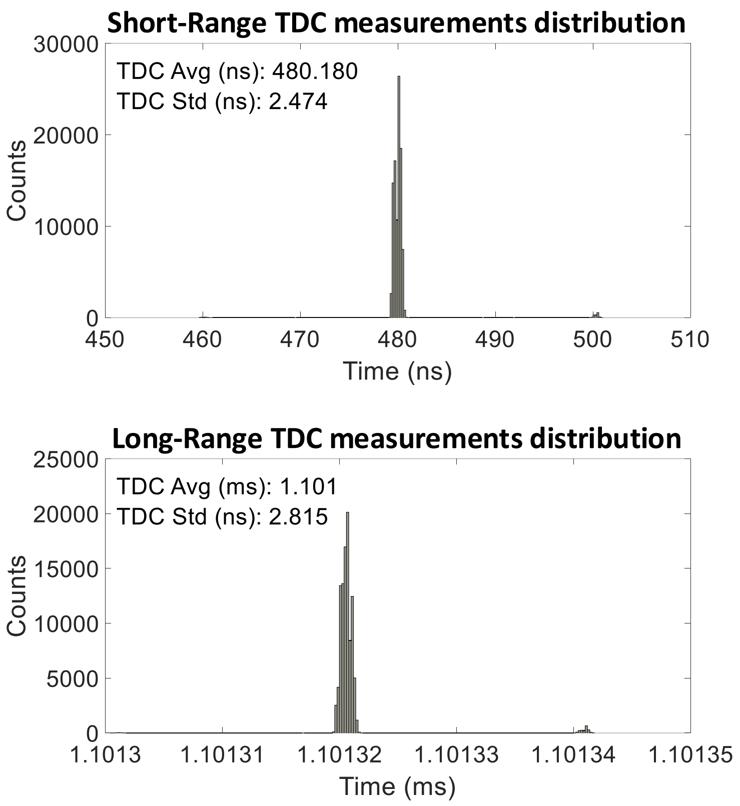


Figure 5.5- ASIC TDC Raw Single-Shot Precision Results

The first thing to notice when analyzing the obtained results is the existence of some outliers with a measurement error equal to one reference clock cycle. Although representing less than 1% of the total measurements done, even with an LSB resolution of 111 ps, the TDC could only achieve a standard deviation of approximately 2.47 ns and 2.8 ns for the short- and long-range measurement, respectively. These results, according to the equations presented in Chapter 2, in a single-shot resolution of 1.75 ns and 1.98 ns respectively. These outliers represent corner cases that the implemented synchronizer is not able to correct. In Section 5.3, a study of the TDC's performance when no outliers are measured is presented. An analysis on the current architecture and a proposal for simplifying the synchronizer and guarantee its correct operation in every scenario is also presented on Section 5.3.

5.3. Synchronization Issues

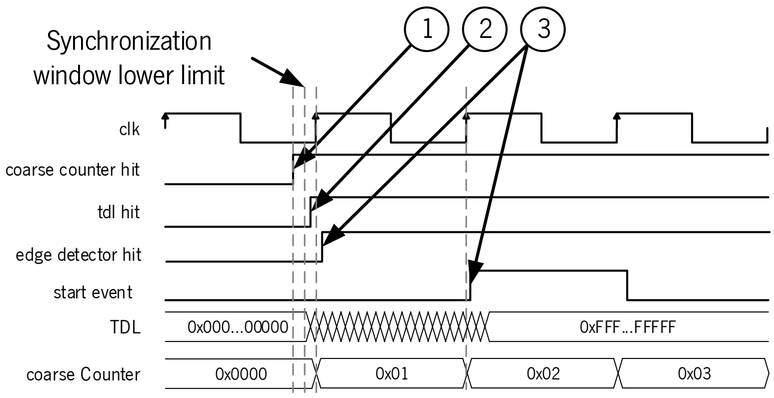
Although the synchronizer implemented on the FPGA-based TDC was able to effectively correct coarse measurement errors, when migrated to ASIC, some corner cases were not being correctly compensated. These scenarios were not detected during post-layout timing simulations due to the synchronizer block functioning timing dependency. Furthermore, since some start and stop values were sampled with metastability (which is normal due to the TDL asynchronous principle of operation), some of the outputted TDC values in the simulation could not be validated, contributing to this phenomenon to go unnoticed.

5.3.1. Architectural Changes

Multiple parameters must be controlled during the synchronizer implementation. The first, and most important, is the reference clock phases generation. Depending on the clock phases generated, the comparison values used on the synchronizer must be changed accordingly, to adjust the synchronization window. The hit signal skew between the first stage of the TDL, the enable pin of the coarse counter registers and the start and stop event detectors is also important and should be kept as small as possible, otherwise, scenarios like the one depicted in Figure 5.6 might occur, resulting on an incorrect synchronization error detection. Finally, the clock skew between the first sampling registers of the TDL, the edge detector modules and the coarse counter, must also be controlled and within a limited range of values defined by the implemented reference clock phases. This hardware dependency implementation makes the synchronizer block susceptible to PVT variations and demands manual tuning based on simulated results (in the case of ASIC platforms) or experimental results (in the case of FPGA). Since the simulated results are based on worst- or best-case scenarios, it is hard to correctly tune this solution for all scenarios, resulting on some corner cases not being covered in the ASIC implementation.

These concerns were addressed in detail in [5.4], where a methodology for designing synchronizers for Nutt-TDCs is presented, based on the experience acquired and the analysis of the ASIC TDC performance results. However, the proposed methodology demands for a different coarse counting schema and is still hardware dependent.

In order to overcome the above limitations, a different solution was explored and later tested using an FPGA platform. The alternative solution is based on the sampled values of the first and last step of the TDL. Since the TDL is designed to encompass a time period greater than the reference clock cycle, the exclusive OR operation between the first and last step of the TDL can be used to identify a start or stop event. Thus, since the TDL is sampled using the reference clock, the generated start and stop events are already synchronous. The proposed solution schematic is presented in Figure 5.7.



1 Hit signal arrives earlier to enable pins of coarse counter (coarse will update correctly because timing respects setup constraints).

2 Hit signal arrives the tdl inside synchronization window, thus the synchronizer will alter the value of the main coarse counter or select a different coarse counter as the correct value.

3 Hit signal arrives the edge detector after the clock rising edge delaying the TDL sampling for the duration of one clock cycle, resulting in an incorrect synchronization error detection.

! Shifting the given example to the right, the hit signal could arrive to the enable pins of the coarse counter inside the metastability window but due to the hit signal skew the value sampled in the TDL would be outside of this window, resulting in a not detected synchronization error

Figure 5.6- Effect of the Hit Signal Skew on the Synchronizer

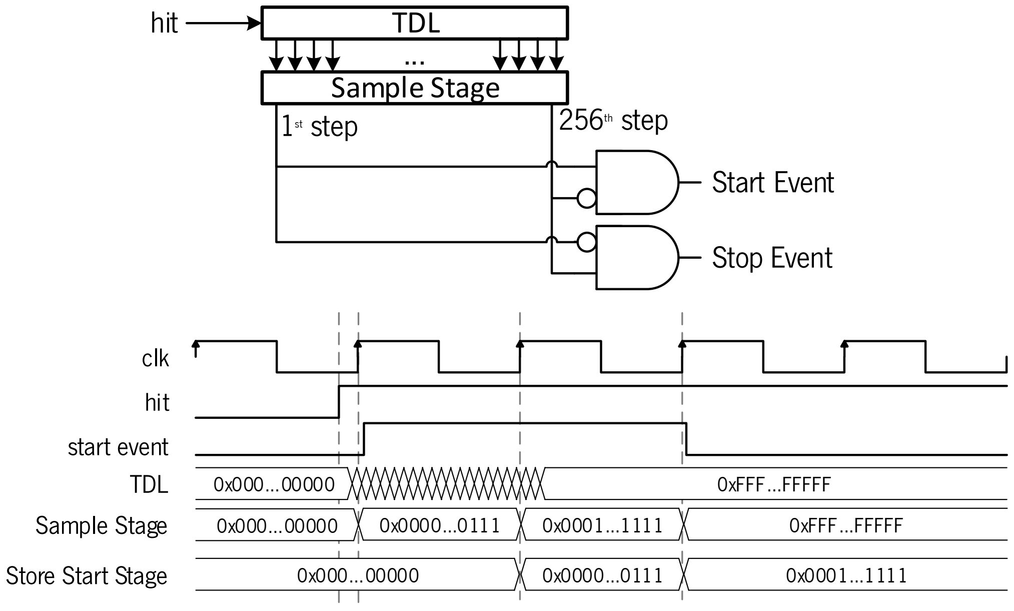


Figure 5.7- Hardware Independent Synchronization Circuit

The validation of the proposed solution was done by changing the synchronizer block on the FPGA-based TDC architecture. The obtained results were similar to the ones presented on Chapter 3, with the difference of an offset in the code density test, equal to the propagation delay of the extra TDL steps minus the setup time of the sampling flipflops (as can be seen in the waveform diagram in Figure 5.7, where the TDL is sampled two consecutive clock cycles by the Store Start Stage). Due to the adopted synchronization method, any hit event arriving inside the window captured by the first kth-1 steps of the delay line (with k being the number of extra steps implemented to cover more than one reference clock period), will be captured again during the second clock cycle. Thus, and considering *j* as the number of steps required to cover a full reference clock cycle, the TDC output for a hit signal in this scenario will always be *j+(k-1)*. Since both start and stop event are sampled using the same TDL, this has no effect on the final measurement precision. Regarding the coarse counter, although it is sampled two times, meaning one less count, since the TDL is also sampled twice, it will have a timestamp equal to one coarse count plus the fine time to be measured. This secures that the system always outputs a correct measurement value with no errors greater or equal to one coarse counter.

5.3.2. Corrected Precision

In order to evaluate the true precision of the TDC (without synchronization errors), the MATLAB script was changed to analyze the obtained results and correct any measurement error with a deviation from the mean measurement value equal or greater than one reference clock cycle. The correction simply added or subtracted the value of one reference clock cycle period, keeping the fine measurement information intact. The corrected short- and long- measurement range precision are presented in Figure 5.8.

As can be observer, once the synchronization issues are corrected, the TDC standard deviation drastically improves to 318.6 ps and 409.8 ps, for short- and long-range measurements, respectively (an improve of approximately 85%). This corresponds to a single-shot resolution of 225.3 ps and 289.7 ps, respectively. Although the achieved performance already complies with the defined requirements targeted by this Thesis application, the non-linearity results strongly suggest that the performance of the TDC can be further improved if a calibration mechanism is applied.

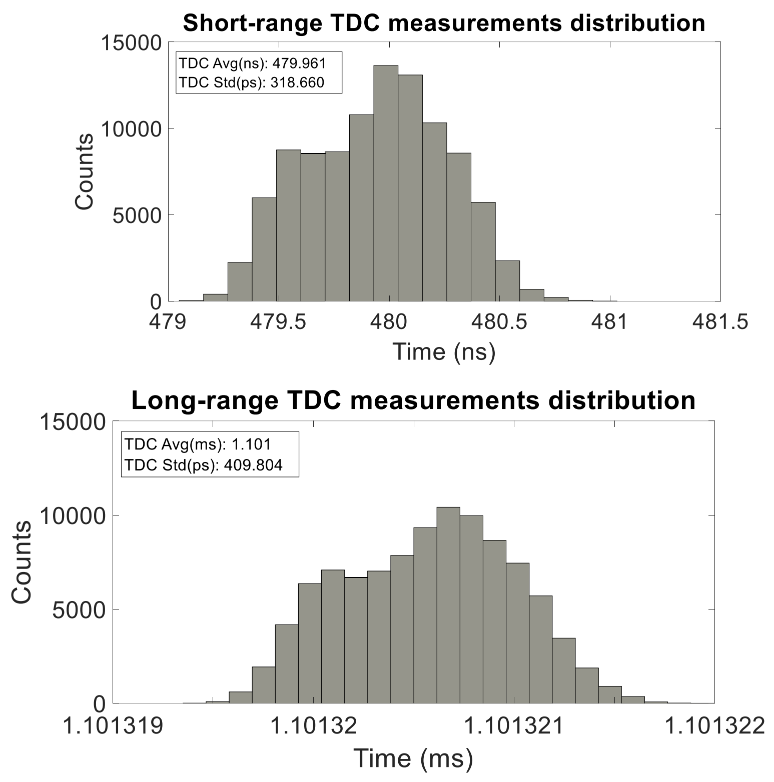


Figure 5.8- ASIC TDC Precision after synchronization errors correction

5.4. Linearization Issues

Although the implemented methodology enabled DNL values lower than 1 LSB (see Figure 5.4), when compared to the simulated values, a higher linearity was expected. In order to understand the reasons behind the loss of performance, the implemented TDL was analyzed with further detail.

The first aspect to consider when comparing the experimental results and the simulated ones are process variation and differences in the sampling flip-flops' setup and hold times, which lead to different step timings, and cannot be controlled by design (unless a full custom design approach is adopted to try to minimize these effects). However, in 0.18 µm process technology (which is already a stable and well categorized process), the effect of process variation on standard digital cells, although present, should not generate such linearity discrepancies in the same chip.

The second aspect to consider was the parasitic capacitance introduced by routing. These were covered with the Structured Data Path approach, which, according to the extracted SDF layout file, proved to reduce variations of tens of picoseconds (when no SDP was applied) to a maximum of one picosecond step's delay variation in the worst-case scenario. Thus, the observed experimental variations cannot be explained by the delay line interconnects.

Therefore, the reason for such linearity variations must be closely coupled to the clock tree distribution along the delay line, since this was the only TDL layout process that was only partially controlled through clock skew constraints. Following, a detailed analysis on the generated clock tree and skew timings is presented. To improve the TDL's linearity, future design considerations to control the TDL clock tree distribution are also discussed.

5.4.1. Layout Considerations

When designing a TDL, the analysis is usually done considering the clock tree distribution as ideal, which results in a delay step equal to the propagation delay of the element used to build the TDL. However, the routings' skew in the clock tree distribution can be consider as another delay line, leading to a Vernier delay line structure, where both start and stop signals are delayed. This means that the effective step delay will vary according to its clock signal insertion delay. Moreover, due to the parallel structure of the clock tree distribution (as opposed to the serial distribution of the hit signal), when analyzing a scenario where a given step *i* has a higher insertion delay than the previous one *i-1*, the effective result is an increase of the ith step's delay and a decrease on the ith-1 step's delay, equal to the amount of skew between the two sampling registers (see Figure 5.9 and Table 5.1).

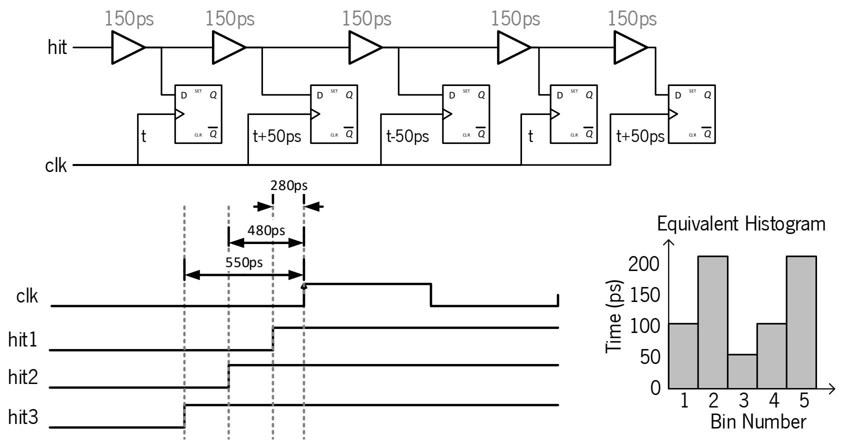


Figure 5.9- Clock Skew Effect on the Propagation Delay of the TDL's Steps

Table 5.1- Example of the Sampled Values of Figure 5.9

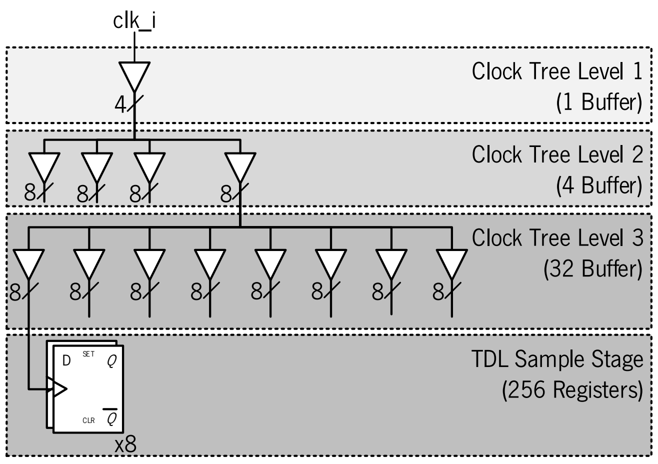
Hit Signal	*Expected TDL Output*	*Real TDL Output*
hit1 (280 ps)	00001	00011
hit2 (480 ps)	00111	00011
hit3 (550 ps)	00111	00111

Thus, controlling the clock tree generation for the TDL is another crucial part of the design process that must be constrained. A representative view of the fabricated ASIC clock tree was presented in Figure 4.15. As depicted, a maximum skew deviation of 115 ps exist in the TDL clock tree distribution. Thus, if half the value of the maximum skew deviation is considered as the 0 point (meaning that the steps could have a ±57.5 ps skew relative to the middle point), and two consecutive steps exist with this skew difference, the resultant effective delay would be, in the worst-case scenario, equal to 172.5 ps (see the histogram presented on the example of Figure 5.9). This justifies some of the biggest discrepancies observed on the code density test histograms presented. Thus, it is possible to conclude that controlling the clock tree distribution when implementing a TDL TDC is as critical as controlling its steps' interconnects. According to Cadence's documentation [5.5], SDP can be used to implement very fast register columns. This functionality, along with the structured and uniform routing provided by SDP, can enable low skew on the TDL clock distribution, improving the proposed TDC architecture linearity and performance, at the cost of a slightest increase on the complexity of TDC design, due to extra SDP constraints and manually inclusion of clock buffer on the TDL Verilog description file. The TDL was analyzed and the designed clock tree, along with the Verilog code changes needed, are presented in Figure 5.10.

5.4.2. Software Calibrated Precision

The non-uniformity of the TDC steps' size can be compensated by post-measurement calibration techniques. Since an INL on the range of 8 LSB was obtained, it is expected that the implementation of a calibration mechanism will have great impact on the TDC final precision, thus justifying its extra processing cost. One of the most commonly used methods to calibrate TDLs is the bin-by-bin calibration.

Bin-by-bin calibration consists on the use of the data extracted from the code density test to build a lookup table. The lookup table can be built based on two main strategies [5.6] (see equation (5.1) and (5.2)). With the lookup table created, instead of multiplying the binary value outputted by the decoder (identifying the last propagated step) by the average step's propagation delay, the decoded value is used as an index



```
//TDL CLOCK TREE
//1st LEVEL
(* dont_touch = "TRUE" *) CKBD0BWP7T clk_tree_buffer_lvl1(
    .Z(clk_lvl_1),
    .I(clk_i)
);

//2nd AND 3rd LEVELS
genvar m;
genvar n;
generate
    for(m=0; m<256/64;m=m+1) begin
        //2nd LEVEL BUFFERS
        (* dont_touch = "TRUE" *) CKBD0BWP7T clk_tree_buffer_lvl2(
            .Z(clk_lvl_2[m]),
            .I(clk_lvl_1)
        );
        for(n=0;n<8;n=n+1) begin
            //3rd LEVEL BUFFERS
            (* dont_touch = "TRUE" *) CKBD0BWP7T clk_tree_buffer_lvl3(
                .Z(clk_lvl_3[m*8+n]),
                .I(clk_lvl_2[m])
            );
        end
    end
endgenerate
//FIRST STAGE D FLIP FLOPS TO SAMPLE DELAY CHAIN
genvar j;
genvar k;
generate
    for(j=0;j<256/8;j=j+1) begin
        for(k=0;k<8;k=k+1)
        begin : sample_stage
            (* dont_touch = "TRUE" *) DFQD0BWP7T tdc_val_reg(
                .Q(tdl_val_r[j*8+k]),
                .CP(clk_lvl_3[j]),
                .D(tdl_val_w[j*8+k])
            );
        end
    end
endgenerate
```

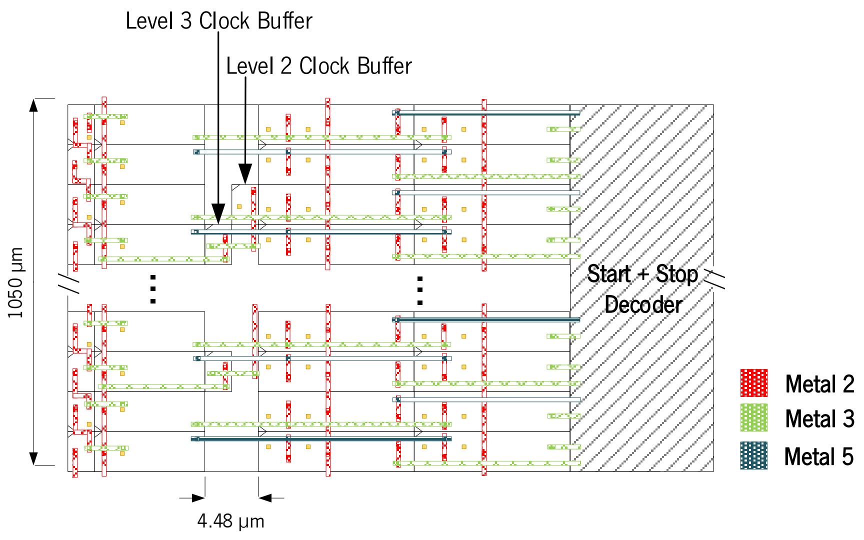


Figure 5.10- TDL Clock Tree Design and Layout

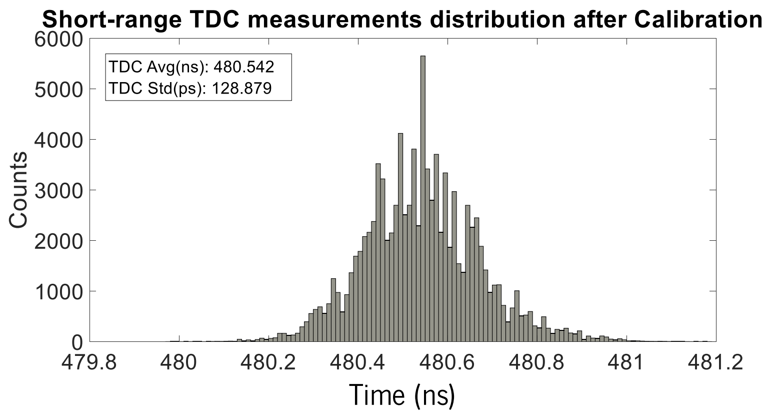
to the calibration table, i.e. the TDC calibrated value will be the one stored on the ith position of the lookup table (being *i* the value outputted by the decoder).

$$t_i = \sum_{j=0}^{i} d_j, \quad (5.1)$$

$$t_i = \frac{d_i}{2} + \sum_{j=0}^{i-1} d_j, \qquad (5.2)$$

In the above presented equations, t_i is the calibrated delay value of the lookup table to the ith step, d_j is the true delay of the jth step (obtained from the code density test), and d_i is the true delay value of the ith step. The first approach considers the sum of the delay values obtained with the code density test as the calibrated values, while the second one calibrates the values to the center of each step (the first term is always half of the current step delay). According to [5.6], the RMS measurement errors are reduced when the calibration is done to the center of the step. However, the first approach (the one considering only the sum of the terms), is easier to implement, especially if the calibration is to be implemented in hardware. In this Thesis, since the objective of calibration was to study the potential of the developed TDC if linearization errors are minimized, the first approach was adopted to implement a software calibration table for the TDC.

A calibration table was built according to equation (5.1) with the data obtained from the code density test presented in Section 5.2.1. The calibrated single-shot precision results for short- and long-range measurements are presented in Figure 5.11. The TDC's precision is enhanced to 128.9 ps and 189.3 ps (for short- and long-range measurements respectively) when calibration is applied, representing a 40% and 46% improvement.



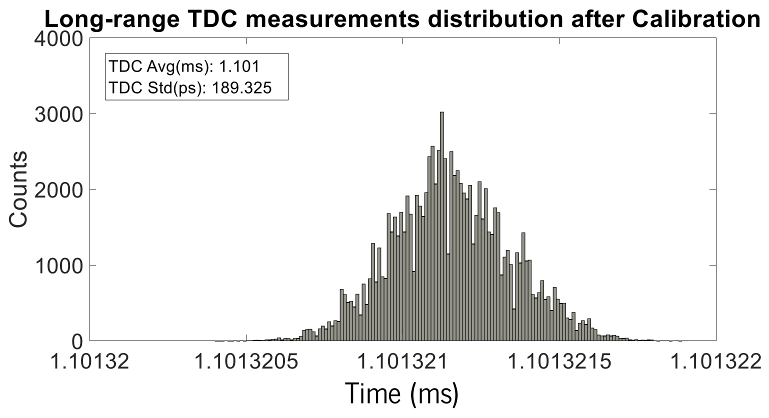


Figure 5.11- ASIC TDC Calibrated Single-Shot Precision

5.5. Thermal Stability

Temperature conditions influence electronic circuits working characteristics. Consequently, the performance of the developed TDC will vary according to temperature variations, due to changes on the cells' propagation delay. Thus, a study on the effect of temperature on the TDC performance was conducted. A temperature range between 0°C and 50°C was analyzed. The code density tests for the corner cases (0°C and 50°C) are presented in Figure 5.12. As can be seen, in the 0°C scenario, the temperature variation has a reduced effect on the size of the TDL steps (when compared to the typical case, 20°C, presented in Figure 5.3). However, at 50°C, only 168 steps are sampled, meaning an average step delay of 119 ps, 7.9 ps more than the typical operation scenario.

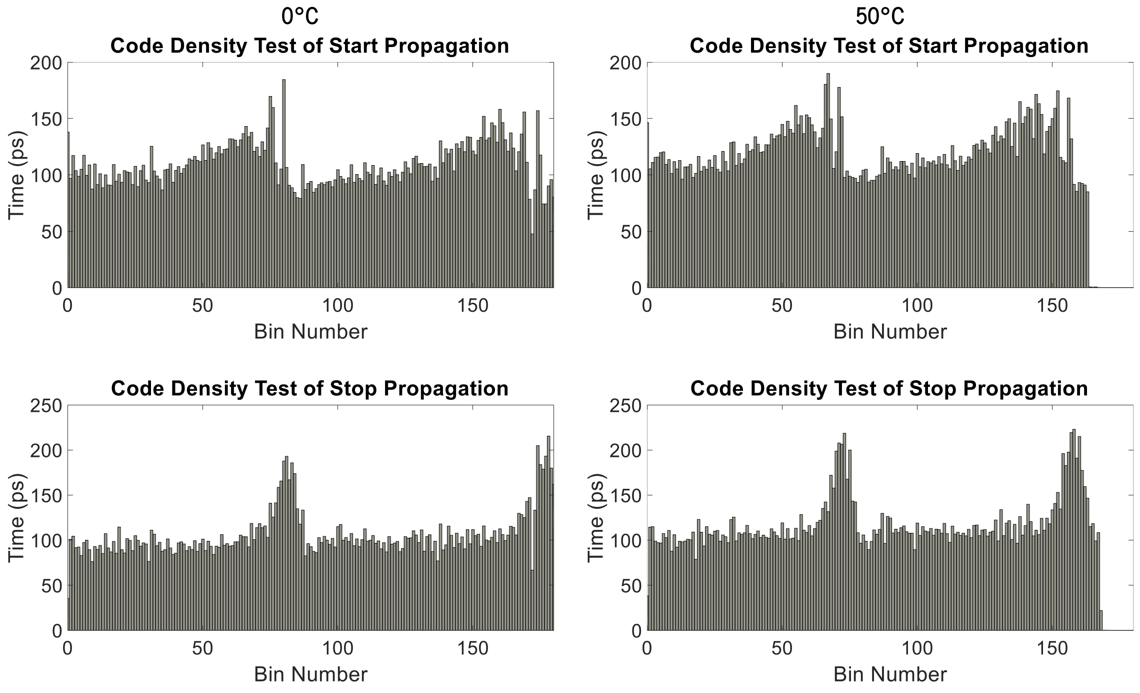


Figure 5.12- Code Density Test Results for Temperature Corner Cases

The precision of the TDC and its variation with temperature was studied in three different configurations. First, the average step size was maintained at 111.1 ps (the size of the step at ambient temperature of 20°C) and the precision was calculated for the different temperatures without applying any type of calibration. In this scenario a maximum precision variation of 94 ps was observed (see Table 5.2 and Figure 5.13). Then, a calibration table built according to the process described in Section 5.4.2 was employed to calibrate the TDC across all ranges of temperatures. This solution proved to be effective when the TDC is operating at temperatures in the range of 20°C or below. However, for temperatures above 30°C, the calibrated precision converges to values similar to the ones obtained when no calibration is applied. Finally, a calibration table for each 10°C temperature range interval was created and applied to the TDC measurements. With this solution, a maximum of 2 ps precision deviation was obtained. This method, although proving to be very effective, demands for extra memory usage since different calibration tables must be stored. Moreover, it also requires constant monitorization of the operational temperature of the TDC in order to properly select the calibration table to be applied.

With the performed temperature test, to reduce the need for different calibration tables per temperature range, an equation representing the step's propagation delay variation with temperature was developed. Considering the average step propagation delay variation between 20°C-50°C, a 0.243 ps/°C was obtained.

Table 5.2- Single-Shot Precision Values along the Studied Temperature Range

Temperature (°C)	*Precision (ps)*		
	Without Calibration	With Single Calibration Table	With Multiple Calibration Tables
0	348	189	128
10	337	136	127
20	318	129	129
30	360	233	129
40	385	323	129
50	412	394	129

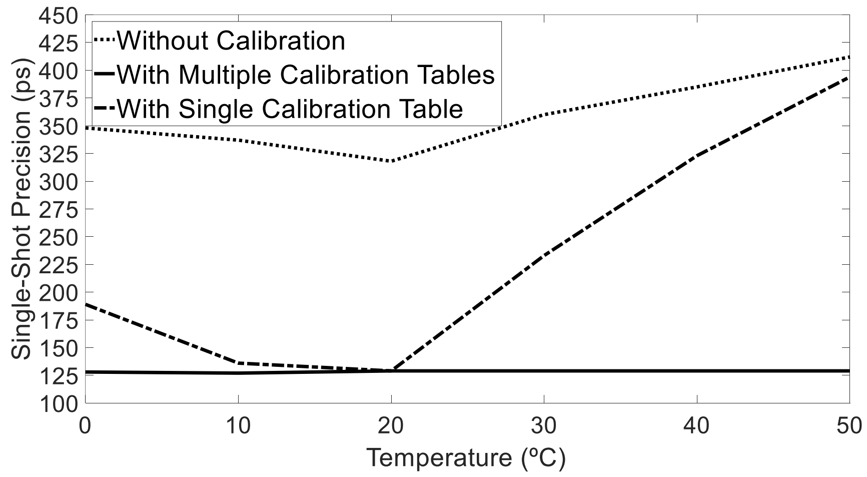


Figure 5.13- Single-Shot Precision variation with Temperature

Therefore, using the normal operation temperature (20°C) calibration table as the base correction value, the calibrated TDC measurement for different temperatures can be obtained using equation (5.3).

$$s_i = c_i + (\Delta t * 0.243), \qquad (5.3)$$

Where s_i is the calibrated value with temperature compensation, c_i is the value obtained from the base calibration table, and Δt is calculated by subtracting 20 to the current operation temperature. The results of using this methodology are presented in Figure 5.14. This solution still requires the monitoring of the TDC's operation temperature but, although not offering the most optimal results, allow for a favorable trade-off between TDC's precision and memory requirements.

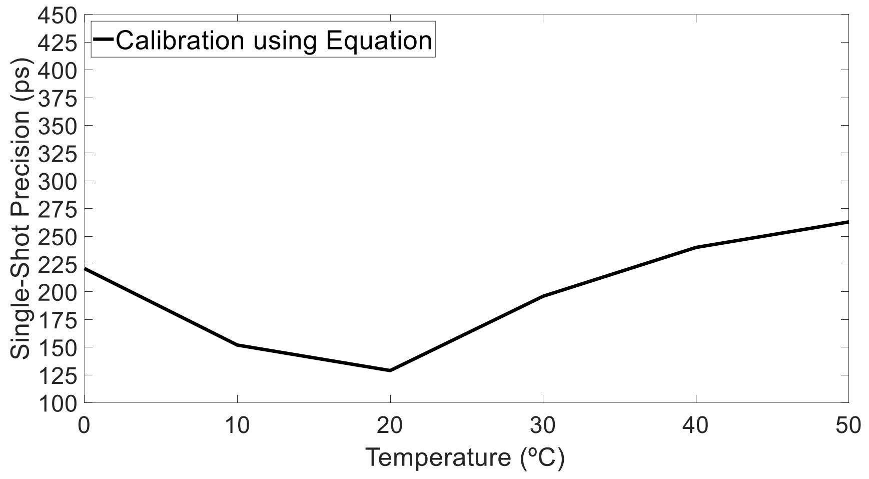


Figure 5.14- Corrected Single-Shot Precision Variation with Temperature using Equation 5.3

Finally, it is important to state that, although effective on increasing the TDC precision across a different range of temperatures, the obtained precision variation is still higher than when a dedicated temperature calibration table is used. This is because each step is differently influenced by the temperature, and the TDL's non-uniform clock distribution tree is probably one of the major contributors. Therefore, it is expected that the clock tree changes proposed in Section 5.4.1, not only will reduce the need for calibration due to higher TDL linearity, but also contribute to a more uniform temperature influence on each TDL's step, resulting on a more effective temperature compensation when using the proposed equation.

5.6. Conclusion

The achieved results comply with the requirements for modern LiDAR applications with a resolution and precision higher than 400 ps and a measurement range of more than 1 ms. When the calibration is applied, depending on the strategy adopted, a resolution of 128 ps can be attained, resulting in a depth measurement resolution of 1.9 cm. Further performance improvements can be achieved if statistical methods (like multiple measurements average) are applied instead of using single-shot measurements.

Although effective on reducing the synchronization errors, due to its intrinsic hardware and timing dependence, the migrated synchronizer block was not capable of completely eliminate the issue. A simpler and hardware independent solution was proposed and later validated using an FPGA platform. The proposed solution does not require any hardware placement or timing concern, thus making it ideal for porting. Moreover, it greatly reduces the hardware utilization and the need for generating phased clocks.

Regarding linearity, although DNL was always in the range of ±1 LSB, the INL of the TDL oscillated in the range of 8 LSB. Therefore, calibration mechanisms must be implemented to minimize measurement errors and improve precision. These results contrasted with the delay line timing information obtained from the SDF file. The origin of the non-linearities of the TDL were investigated and a solution to improve it was proposed based on controlling the clock tree generation for the TDL.

Temperature compensation methods were proposed to secure a stable TDC operation across a 0°C to 50°C range. It is expected that, once the linearity of the TDL is improved by the homogeneous clock tree distribution, correcting the TDC temperature drift using an equation describing its thermal behavior may become more effective.

Further experimental results must be obtained using different ASIC TDC samples to investigate the repeatability of the results, and check if the ASIC sample tested was not in one of the corner cases. Moreover, the available function generator was not ideal for sub-nanosecond measurements due to its typical jitter of 1 ns. This is certainly introducing time interval generation errors that are deteriorating the TDC performance measurement, especially its precision. Nevertheless, the obtained preliminary results are promising, and the suggested architectural changes have the potential to further improve the TDC's performance.

References

[5.1] Texas Instruments, "TDC7200 Time-to-Digital Converter for Time-of-Flight Applications in LIDAR, Magnetostrictive and Flow Meters.", TDC7200, Rev. March 2016, Texas Instruments, pp. 1–50, 2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/tdc7200.pdf

[5.2] Analog Devices, "±0.5°C Accurate, 16-Bit Digital SPI Temperature Sensor.", ADT7310, Rev. A, Analog Devices, 2011. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/ADT7310.pdf.

[5.3] Tektronix, “Arbitrary/Function Generator AFG1000 Series Datasheet.” Tektronix, 2016. [Online]. Available: https://www.tek.com/datasheet/arbitrary-function-generator.

[5.4] R. Machado, J. Cabral, and F. Alves, “Designing Synchronizers for Nutt-TDCs,” in 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), 2019, pp. 1–6.

[5.5] Cadence, “Innovus User Guide,” 2016.

[5.6] J. Wu, “Uneven bin width digitization and a timing calibration method using cascaded PLL,” in 2014 19th IEEE-NPSS Real Time Conference, 2014, pp. 1–4.

6. Conclusion

The theoretical and experimental verification and validation of a high resolution synthesizable TDC was presented in this Thesis. Two different TDC architectures were studied, analyzed and implemented using fast prototyping platforms (FPGA prototypes). After a preliminary study, the Tapped Delay Line (TDL) TDC architecture was selected since it showed better linearity and precision performance. The architecture was migrated to ASIC. The migration process and scripts were presented and discussed in Chapter 4. A complete TDC characterization of the FPGA prototypes and of the final TDC ASIC was made, and the results were presented in Chapter 3 and Chapter 5. The discussion, in Chapter 5, includes the limitations and shortcomings of the proposed TDC's solution, and proposals for future improvements were also identified.

In this Chapter a summary of the developed work, main contributions and conclusions of this Thesis is presented. Future research work proposals are also identified and discussed.

6.1. Conclusions

Based on the results obtained during this Thesis research work, some general conclusion can be drawn regarding TDCs and their applications. The fast evolution of FPGA devices has made possible the implementation of TDCs with performances comparable to those of ASIC based TDC. The technical evolution of FPGAs was mainly triggered by lower development costs, fast development cycles and system's reconfigurability functionalities. These features have made FPGA's attractive during the last decade as a prototyping platform and, in some cases, for early product deployment [6.1]. Nonetheless, TDC's performance is closely coupled to the technology being used. The typical ASIC development approach is based on a full-custom process, in which the TDC is designed at the transistor level, taking advantage of the full design flexibility that this technology offers. However, this flexibility hinders design portability to other manufacturing technology. Even though some research works targeting synthesizable TDCs for ASIC are presented in literature ([6.2], [6.3]), it is always necessary to select which standard cell should be used when coding in HDL. The same happens during FPGA design, where the structure of the Configurable Logic Block (CLB) must be known to properly implement a TDC. Thus, it is difficult to envision a TDC architecture that can completely abstract the designer from the details of the hardware underneath. Nevertheless, synthesizable architectures offer better portability, reducing the required effort when technology migration is required. Regardless of the adopted technology (ASIC or FPGA), a TDC will always be highly dependent on the ASIC fabrication technology or FPGA architecture selected. For instance, although the Gray-code architecture does not present any technology dependent instantiation when analyzing the HDL code, the architecture is carefully tuned for Xilinx 7-series FPGA that has LUTs with six inputs. When the proposed architecture was migrated to ASIC, its technology dependence became clear, with the generation of different combinatorial paths that result in poor linearity and low resolution. Therefore, research on TDCs design automation should be pursued if faster development cycles and portability are desired. A complete hardware and platform independent TDC IP is yet to be achieved.

As mentioned in Chapter 1, the main motivation for this Thesis was to develop a time interval readout system capable of complying with the automotive LiDAR requirements. Since LiDAR is a technology that is still under research, the readout system to develop should promote portability and integration on a full automated design process, in order to promote a seamless technology migration.

A fully digital, synthesizable, TDC architecture implemented using a technology independent HDL and a fully automated design flow has been achieved in this Thesis. A novel Tapped Delay Line (TDL) TDC architecture was described, which uses Structured Data Path (SDP) to constraint placement and routing, in order to achieve superior linearity and performance independently of the process technology adopted.

A thorough literature review on FPGA-based TDCs was conducted (to the extent of the author's knowledge, the only one focused on FPGA TDC architectures to the date) following the described methodology in RM1, resulting in publication J1 [6.4]. Apart from summarizing the most relevant works on the field of FPGA-based TDCs, the work also proposed a taxonomy to classify the existing FPGA TDC architectures and provides paths for future research on this field. This artifact enabled the achievement of O1 while answering the research questions RQ1 and RQ2, introduced on Chapter 1.

After conducting RM2, two prototypes based in different architectures (TDL TDC and Gray-code TDC) were developed in FPGA, according to RM3. These architectures served as base for the portability methodology development and enabled the exploration of software and hardware calibration mechanisms. The test procedures and TDC characterization process were also explored using the FPGA prototypes implemented, according to what was defined in RM4.

These prototypes resulted in two publications, C1[6.5] and C2 [6.6], and addressed research question RQ2. Regarding the TDL TDC prototype, a resolution of 17.2 ps was achieved, with a maximum differential non-linearity of 3.3 LSB and integral non-linearity of 5.6 LSB peak-to-peak. The prototype achieved a raw single-shot precision of 211 ps and a calibrated single-shot precision of 179 ps (please refer to Chapter 2 where a description of resolution, DNL, INL and single-shot precision are presented, according to the definitions used in the literature on TDC's research field). Using the Gray-code TDC prototype, the effect of routing on the TDC resolution, linearity and precision was studied. It was demonstrated in this Thesis that, when targeting sub-nanosecond resolutions, controlling the routing between TDC interpolation steps is mandatory to achieve superior performance and scalability. It was also proved that, while maintaining the routing patterns, higher performance similarity between TDC channels could be achieved. This opens the possibility to use a single calibration mechanism to multiple channels, allowing savings in terms of area and power consumption. These conclusions resulted in artifact P1 that, by the time of this Thesis submission, was under revision. The Gray-code TDC prototype achieves a resolution of 380.9 ps with a differential non-linearity of 0.38 LSB and integral non-linearity of 0.71 LSB peak-to-peak. A single-shot precision of 290 ps was measured with or without calibration for the gray-code prototype, proving the efficiency of the proposed manual routing. The obtained results were limited by the error introduced by the waveform generator used, which had an average jitter superior to any of the implemented TDC prototypes' resolution. With the implementation and characterization of these prototypes O2 and O3 were achieved (please refer to Chapter 1 for the definition of RQx, Ox and RMx).

Based on the conclusion drawn from the implemented FPGA prototypes and the study of the ASIC CAD tools conducted according to the described methodology in RM5, a design methodology for a technology independent TDC design was developed. The methodology support is the use of SDP constraints to secure a stable TDL generation across different technologies, which ensures improved linearity. This culminate in the production of artifact J2 [6.7], answering RQ3 and achieving O4 and O5. A set of TCL scripts were developed during this Thesis to implement the proposed design flow and all the auxiliary files required, including project hierarchy structure and CAD tools' configuration files.

Following the proposed design flow methodology, the FPGA-based TDL TDC prototype was migrated to TSMC 0.18 μm CMOS technology, in order to validate and study the effectiveness of the proposed approach. The migration process followed the research methodology described in RM6 and RM7. According to the post-layout timing simulated results, for the worst-case scenario, a tenfold improvement on the TDC's linearity was achieved when using SDP to constraint the layout. As technology advances, reaching lower process nodes, the effect of routing parasitics in TDCs become a major concern. The study performed using Structured Data Path (SDP) to constraint the TDC's placement and obtain a uniform routing proved to be effective on homogenizing the delay line steps' delays. As presented in Chapter 4, the extracted post-layout timing information showed a reduction greater than 90% on the cells' delay variation, when SDP was used. Thus, the use of SDP on synthesizable TDCs is advantageous to improve linearity and performance. Moreover, SDP secures a technology independent structured placement and patterned routing, improving design's portability. The resultant ASIC layout was fabricated (O6), tested and characterized (according to the description in RM8). The achieved performance proved to be capable of complying with LiDAR requirements, validating the proposed methodology. The fabricated ASIC was able to achieve a resolution of 115 ps and a single-shot precision value better than 400 ps, without calibration, as described in Chapter 5. When calibration was applied, a single-shot precision value of 150 ps was achieved, proving that the TDC is suited for modern automotive LiDAR applications, one of the main goals of this Thesis, answering the final research question (RQ4) and completing O7.

Finally, for TDL TDCs, the clock tree generation must also be constraint using SDP. Clock skew group constraints, although very useful to eliminate "bubble" issues (please see Chapter 4), are not adequate to ensure acceptable values of DNL and INL. Furthermore, temperature variation on asymmetrical TDLs' clock tree has different impact per TDL step, reducing the effectiveness of the TDC calibration. This was noticeable in the description presented in Chapter 5, when an equation characterizing the TDL's steps propagation delay variation with temperature was used to calibrate the TDC.

The fabricated TDC performance is summarized in Table 6.1, along with a comparison with some current state-of-the-art devices. As can be seen, although not capable of competing with the most sophisticated TDC devices in the literature, the TDC described in this Thesis work can still achieve better resolution and linearity than some recently proposed TDCs (implemented using the traditional custom cell design process), while operating at a lower clock frequency. However, the use of standard cells instead of the typical custom cell design is process, results in lower power and area performance. Nevertheless, the major contribution of this Thesis are the scripts which enable the implementation of a TDC using a full automated design flow, that were created during this Thesis research. These scripts only receive the name of the delay element to be used in the TDL construction as input and automatically generate the layout of the TDC. According to the results obtained, the fabricated ASIC-based TDC offers standard cell level resolution and precision in the range of ±3 LSBs with no calibration mechanism (1.15 LSB when calibration is applied for short-range measurements).

Table 6.1- State-of-the-art Comparison

	This Work	*[6.8]-12*	*[6.9]-19***	*[6.10]-18*	*[6.11]-18*	*[6.12]-14*	*[6.13]-17*
Technology	180nm	350nm	180nm	130nm	180nm	130nm	350nm
Architecture	TDL*	Two-Step (DLL)	Two-Step (Pulse Shrinking)	Phased Clocks	Two-Step (Cyclic Vernier)	Two-Step (DLL)	DLL
Resolution (ps)	111	8.878	2	780	377	5	320
Precision (LSB)	2.8 1.15*	1.1	0.7	0.05	0.82	0.6	0.73
DNL (LSB)	±0.8	-8.3:5.8	1.5	±0.05	1.41*	±0.9	±0.68
INL (LSB)	-2.6:3.8-	-22:7.5	4.2	±0.05	2.31*	±1.3	±1.21
Range	1310.72 µs	4.5 ns	130 ns	102.4 µs	355 ns	-	2.5 µs
Power (mW)	36 (@1.8 V)	85 (@3.3 V)	18 (@1.8 V)	6.5 (@1.5 V)	0.65	43 (@1.2V)	10.9 (@3.3 V)
Area (mm^2)	0.89	8.88	0.08	-	0.028	-	0.152
Operating F. (MHz)	50	220	-	320	-	781	100

*After calibration **Simulation

6.1.1. Contributions

This Thesis contributes to the advance of the current state-of-the-art by exploring the effect of routing on the TDCs linearity. The number of works exploring TDC's routing is scarce, being the work by Zhang et al. [6.14] one of the few exploring routing as an interpolation step. The work by Chaberski et al. [6.15] explored the effect of the output load of the TDC's interpolation steps by implementing "dummy" buffers on the output of each interpolation step. Nevertheless, to the best of the author's knowledge, there are no works exploring the routing to reduce the TDC non-linearities, being the traditional approach the implementation of calibration tables, in the FPGA case, and the design of the TDC in a locked-loop configuration, in the case of ASIC. The implementation of the gray-code architecture with controlled routing in this Thesis had proven that the proper control of the routing resources in FPGA could lead to a high linearity TDC that can reach high performance without calibration. Furthermore, the proposed method proved to be efficient on securing a scalable and uniform TDC channel, independently of its positioning in the FPGA. The results obtained were further explored in ASIC with promising results.

Another contribution to the state-of-the-art is the proposal of SDP as a methodology to achieve a technology agnostic, linearity improved TDC. While traditional ASIC-based TDC architectures require interpolation stages designed at the transistor level, to improve resolution and steps mismatch, which increase the architecture technology dependency, this Thesis proposed SDP as a way to implement a synthesizable TDC with a fixed placement and routing, independently of the technology being used, to improve the TDC's linearity. The improvement achieved by controlling the routing of the TDC thorough SDP, proved that, when targeting sub-nanosecond resolution, this aspect of the design cannot be overlooked, whether in ASIC no FPGA. Moreover, the proposed method can be applied to other synthesizable architectures, like the one presented in [6.2], in order to further improve the TDC's performance.

The set of scripts created to support the proposed methodology presents another contribution. ASIC design is an interactive process where, more often than not, some phases must be revisited to improve performance and/or correct system's functionality. The proposed methodology, as well as the design of a synthesizable TDC, requires small changes on a typical ASIC design flow. Thus, the set of scripts developed will be helpful to anyone designing synthesizable TDCs.

This Thesis final contribution is the new readout system for time interval measurement with LiDAR compliant performance. While there are multiple time interval systems on the literature, this Thesis offers a solution with a tradeoff between resolution, flexibility and development time.

6.1.2. Limitations

Even though several contributions were made to the state-of-the-art, the present work has some limitations that must be identified and addressed to improve the reported results. Although improved flexibility and technology independence have been achieved, the proposed TDC resolution is limited to the propagation delay of the cells available on the standard digital library being used. This is a well-known issue of TDL architectures. If a resolution below the propagation delay is required, other synthesizable TDC architectures must be explored. This would require a new SDP file, tailored to the designed architecture, however, the remaining of the design flow and scripts would be kept unchanged.

SDP guarantees a fixed, technology independent TDC layout, and had proven to be efficient on achieving a uniform routing, however, a deep knowledge of the design being implemented is required to properly configure the SDP constraints and achieve the best possible results. Thus, although useful for simplifying the migration of the TDC architecture to different technologies, the first SDP iteration demands high effort and technology knowhow.

The adopted SDP approach had the objective of reducing the TDC's non-linearities and to improve its precision. Although a DNL under 1 LSB and a single-shot precision of 1.15 LSB had been achieved, in order to further improve precision, the TDC's INL must be reduced. It was proved that the major TDC's non-linearities were due to the clock tree distribution asymmetry. Thus, SDP constraints must be implemented addressing the clock tree to reduce the need for post-measurement calibration.

The major issue of the current version of the TDC presented in this Thesis is related to the synchronization mechanism. Although this mechanism revealed to be important to reduce the number of measurement errors, there are some corner cases in which the synchronization errors are not being correctly identified and corrected. In Chapter 5, a possible solution to address this issue was presented and discussed.

The fabricated ASIC TDC has a slave SPI module which enables its interface with a microcontroller and helps on reducing the ASIC pin count, which in turns contributes for a reduction on the ASIC area since in most digital designs, the pad-ring area is usually superior than the core area. Moreover, the SPI implemented is completely generic and can be integrated in other designs, since it is already validated by this Thesis' implementation. Nevertheless, in applications where higher integration level is desired, the SPI must be removed from the design and the FIFO memory implemented directly mapped into the processor's peripherals address space. Due to time constraints, it was not possible to fabricate a microcontroller with the developed TDC already integrated. The author expects to address this integration on future research, upon correction of the aforementioned synchronization and clock tree distribution issues.

6.2. Future Work

As always, there are several aspects that need to be further researched to enhance the results achieved. Moreover, the conclusions and artifacts of the present work opened a set of possibilities that should be further investigated.

Several hypotheses were identified to address the current TDC prototype limitations. These limitations and the strategies to tackle them, have already been discussed in Chapter 5. Future work will focus on studying the effectiveness of the proposed methods, especially the homogenization of the clock tree distribution to the TDL. This will require a new SDP file creation, in order to add constraints to the placement of the clock buffers included in the TDL design that guarantee minimum clock insertion delay skew between TDL's steps. After, new layout and post-layout timing simulations must be performed to validate if the changes made are able to reduce the simulated non-linearities. Due to the timespan of this Thesis, it was not possible to have this new version of the TDC developed, fabricated and tested. The author is willing to participate in future research.

As stated before, TDCs are highly dependent on the hardware being used. Thus, new technologies leverage TDC performance. Xilinx UltraScale+ platforms have been showing promising results in this field of research, being used by multiple works for TDC implementation (see Chapter 2). Therefore, the migration and performance exploration of the proposed FPGA-based architectures (TDL TDC and gray-code TDC) to Xilinx UltraScale+ platforms are part of the author's plans for future research. Since UltraScale+'s CLB structure and routing resources are different from the ones presented in the Xilinx 7-Series FPGAs (the one used in this Thesis), the exploration of this technology may generate higher performance TDCs, as well as, new architectural possibilities, using the proposed architectures as base.

This Thesis focused the development of TDCs for LiDAR applications. Another popular application for TDCs are All-Digital Phase-Locked-Loops (ADPLL). Although the presented architectures lack the performance typically required by ADPLL (<20 ps resolution), the proposed design flow methodology based on SDP could be applied to other synthesizable TDCs to improve their performance. Consequently, the author intends to explore new and existing synthesizable TDC architectures, like the one presented in [6.2], and implement them using the proposed design flow methodology, in order to explore ADPLL applications.

Apart from the current version optimizations and new applications exploration, future research directions should focus on full system integration (TDC and processing unit), since it may improve peripheral access. For instance, in typical LiDAR applications, frame rates between 10-20 fps are desired. Considering a 20 fps case, the measurement and data processing of an entire frame must be done inside a 50 ms timing window. For a 50° Horizontal x 15° Vertical field-of-view, with 0.15° step resolution, a total of 33400 points per frame must be measured and processed. Considering 200 m as the maximum measurement distance, each point measurement would take 667 ns to complete. Thus, a full frame, at maximum distance, would take 22.2778 ms to be constructed. This leaves approximately 27.5 ms left for frame processing (not considering the transmission time of every measurement when using a serial interface that can be made in parallel to the frame acquisition process for the most part). This rather low timing margin, along with the typically high image processing timing costs, highlights for the need for a faster data access mechanism.

A direct TDC peripheral mapping into the processing system's memory would allow extra slack for the processing image task to be performed. Furthermore, this solution would give the highest possible integration, along with application flexibility and reconfigurability. Regarding the FPGA implementations, an AXI slave interface is already part of the TDC IPs developed. This allows integration with any Arm processors. On the ASIC implementation, a slave SPI is currently being used to give external systems access to the FIFO that stores the TDC's measurements. Therefore, the development of a processor with a TDC IP peripheral integrated in ASIC technology, to increase system integration, is also a research paths of interest.

Regarding the processing system selection, considering the LiDAR embedded application requirements, both Arm Cortex-M0 and the recently popularized RISC-V based cores (like the Rocket Chip [6.16], [6.17]) are possible solutions. In the case of the Arm based architectures, the TDC interface would have to include an AXI interface in order to map the peripheral into the processing unit memory. Some RISC-V processors cores allow the direct connection of the TDC peripheral into memory using a simple data and address buses, making the integration process easier to implement. However, in the case of the Rocket Chip, an AXI or NASTI interface is also required. Figure 6.1 depicts the integration block diagrams for the described scenarios.

The adoption of the Arm processor simplifies system's applications development and fosters integration time, while increasing the costs of the project due to fabrication royalty fees. On the other hand, the adoption of a RISC-V based custom processor allows for the development of a tailor-made solution, more optimize, while requiring extra validation and the implementation of tools and debug structures for applications development support. The use of a RISC-V architecture seems the most promising approach to achieve an optimized solution for the LiDAR application described in Chapter 1. A possible integrated design is presented in Figure 6.2 (using a RISC-V based processor available at [6.18]).

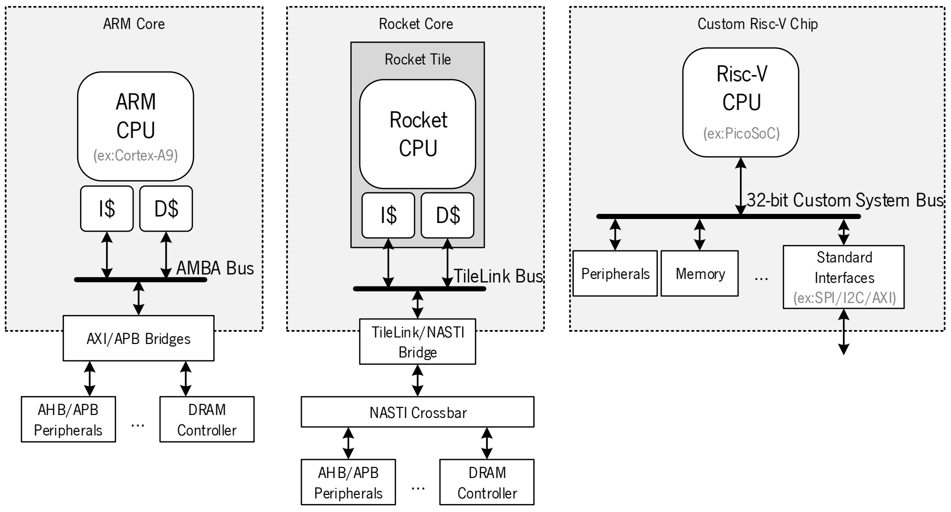


Figure 6.1- Block Diagram of Possible Integration Scenarios

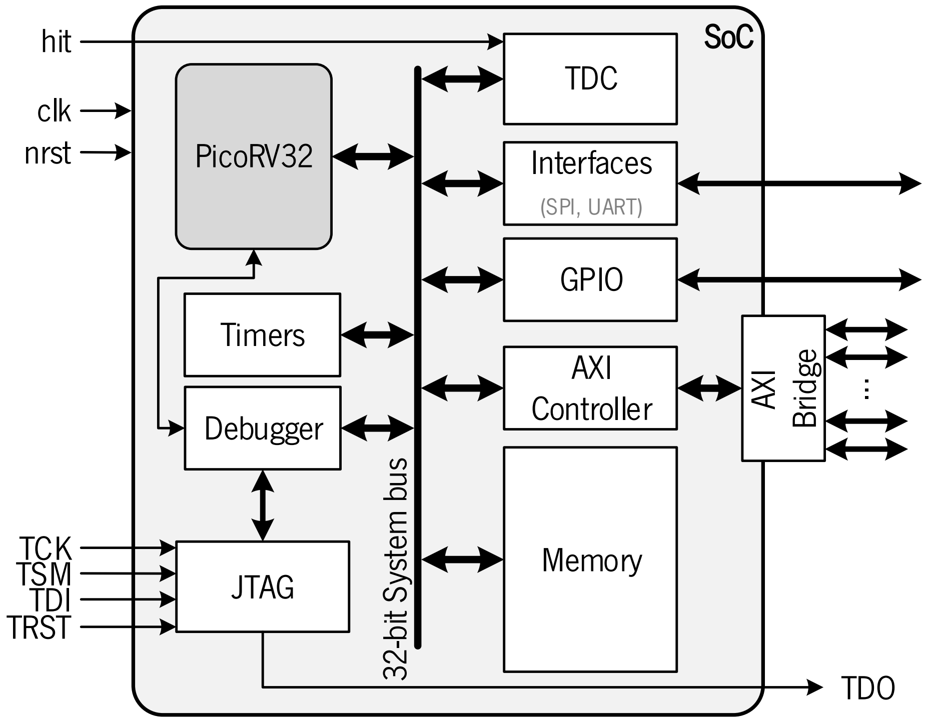


Figure 6.2- Block Diagram of the Proposed TDC and Processing Unit integration

References

[6.1] R. Szplet, P. Kwiatkowski, K. Różyc, Z. Jachna, and T. Sondej, “Picosecond-precision multichannel autonomous time and frequency counter,” Rev. Sci. Instrum., vol. 88, no. 12, p. 125101, Dec. 2017.

[6.2] Y. Park and D. D. Wentzloff, “A Cyclic Vernier TDC for ADPLLs Synthesized From a Standard Cell Library,” IEEE Trans. Circuits Syst. I Regul. Pap., vol. 58, no. 7, pp. 1511–1517, Jul. 2011.

[6.3] Y. Park and D. D. Wentzloff, “An all-digital PLL synthesized from a digital standard cell library in 65nm CMOS,” in 2011 IEEE Custom Integrated Circuits Conference (CICC), 2011, pp. 1–4.

[6.4] R. Machado, J. Cabral, and F. S. Alves, “Recent Developments and Challenges in FPGA-Based Time-to-Digital Converters,” IEEE Trans. Instrum. Meas., vol. 68, no. 11, pp. 4205–4221, Nov. 2019.

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

[6.6] R. Machado, J. Cabral, and F. Alves, “Designing Synchronizers for Nutt-TDCs,” in 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), 2019, pp. 1–6.

[6.7] R. Machado, J. Cabral, and F. S. Alves, “All-Digital Time-to-Digital Converter Design Methodology Based on Structured Data Paths,” IEEE Access, vol. 7, pp. 108447–108457, 2019.

[6.8] J.-P. Jansson, “A stabilized multi-channel CMOS time-to-digital converter based on a low frequency reference,” University of Oulu, 2012.

[6.9] R. Enomoto, T. Iizuka, T. Koga, T. Nakura, and K. Asada, “A 16-bit 2.0-ps Resolution Two-Step TDC in 0.18-um CMOS Utilizing Pulse-Shrinking Fine Stage With Built-In Coarse Gain Calibration,” IEEE Trans. Very Large Scale Integr. Syst., vol. 27, no. 1, pp. 11–19, Jan. 2019.

[6.10] J. Wang et al., “Development of a time-to-digital converter ASIC for the upgrade of the ATLAS Monitored Drift Tube detector,” Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip., vol. 880, pp. 174–180, Feb. 2018.

[6.11] V. Nguyen, D. Duong, Y. Chung, and J.-W. Lee, “A Cyclic Vernier Two-Step TDC for High Input Range Time-of-Flight Sensor Using Startup Time Correction Technique,” Sensors, vol. 18, no. 11, p. 3948, Nov. 2018.

[6.12] L. Perktold and J. Christiansen, “A multichannel time-to-digital converter ASIC with better than 3 ps RMS time resolution,” J. Instrum., vol. 9, no. 01, pp. C01060–C01060, Jan. 2014.

[6.13] J. Wu, W. Zhang, X. Yu, Q. Jiang, L. Zheng, and W. Sun, “A hybrid time-to-digital converter based on residual time extraction and amplification,” Microelectronics J., vol. 63, pp. 148–154, May 2017.

[6.14] 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.

[6.15] D. Chaberski, R. Frankowski, M. Zieliński, and Ł. Zaworski, “Multiple-tapped-delay-line hardware-linearisation technique based on wire load regulation,” Meas. J. Int. Meas. Confed., vol. 92, pp. 103–113, 2016.

[6.16] Berkeley Architecture Research, “Rocket Chip Generator,” 2019. [Online]. Available: https://bar.eecs.berkeley.edu/projects/rocket_chip.html. [Accessed: 12-Dec-2019].

[6.17] Berkeley Architecture Research, “Rocket Chip,” 2019. [Online]. Available: https://chipyard.readthedocs.io/en/latest/Generators/Rocket-Chip.html. [Accessed: 12-Dec-2019].

[6.18] C. Wolf, “PicoRV32 - A Size-Optimized RISC-V CPU,” 2018. [Online]. Available: https://github.com/cliffordwolf/picorv32. [Accessed: 15-Nov-2018].

List of Publications

Journal Papers:

J1. R. Machado, J. Cabral and F. S. Alves, "All-Digital Time-to-Digital Converter Design Methodology Based on Structured Data Paths," in IEEE Access, vol. 7, pp. 108447-108457, 2019. doi: 10.1109/ACCESS.2019.2933496

J2. R. Machado, J. Cabral and F. S. Alves, "Recent Developments and Challenges in FPGA-Based Time-to-Digital Converters," in IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 11, pp. 4205-4221, Nov. 2019. doi: 10.1109/TIM.2019.2938436

J3. R. Machado, F. Alves, J. Cabral, “Technology Independent ASIC based Time to Digital Converter”, submitted IEEE Transactions on Circuits and Systems I: Regular Papers [under revision]

Conference Papers:

C1. R. Machado, L. A. Rocha and J. Cabral, "A novel synchronizer for a 17.9ps Nutt Time-to-Digital Converter implemented on FPGA," 2018 AEIT International Annual Conference, Bari, 2018, pp. 1-6. doi: 10.23919/AEIT.2018.8577365

C2. R. Machado, J. Cabral and F. Alves, "Designing Synchronizers for Nutt-TDCs," 2019 5th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP), Vienna, Austria, 2019, pp. 1-6. doi: 10.1109/EBCCSP.2019.8836914

Other:

- J. Pereira, D. Oliveira, P. Matos, R. Machado, S. Pinto, T. Gomes, V. Silva, E. Qaralleh, N. Cardoso, and P. Cardoso, "Hardware-assisted Real-Time Operating System Deployed on FPGA", in "Informatik/Kommunikationstechnik" subseries of the "Fortschritt-Berichte VDI" series edited by VDI Verlag, 2014.
- R. Machado, S. Pinto, J. Cabral, and A. Tavares, “FPGA vendor-agnostic IP-XACT- and XSLT-based RTL design generator,” in 2016 18th Mediterranean Electrotechnical Conference (MELECON), 2016, pp. 1–6. doi: 10.1109/melcon.2016.7495380

Under Revision:

P1. R. Machado, F. Alves, J. Cabral, “Gray-Code TDC Architecture with Improved Linearity and Scalability”, submitted 2020 6th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP) [under revision]

About the Author

Rui Machado was born in Guimarães, Portugal in 1990. He obtained is Bachelor in Electronics Engineering and Computers at University of Minho in 2012. In 2014, he obtained his MSc. degree in Electronics Engineering and Computers, with specialization in Embedded Systems, at University of Minho. He started pursuing his PhD. in Electronics and Digital Systems in 2016, at University of Minho, in a project in partnership with Bosch Car Multimedia. He has been a scientific visitor at INL - International Iberian Nanotechnology Laboratory, at Braga, Portugal, since 2017. His current research interests focus in time-to-digital conversion systems and embedded and digital systems design. During his PhD. he was an invited professor at the Technology School of the Polytechnic Institute of Cávado and Ave (2016) and at the Industrial Electronics Department of University of Minho since 2018.

Orcid ID: http://orcid.org/0000-0001-9929-8705

Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57205499933

ResearchGate: https://www.researchgate.net/profile/Rui_Machado11

Google Scholar: https://scholar.google.com/citations?hl=pt-PT&user=5K6B0ZgAAAAJ