FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Optimization of LiDAR MEMS Data Computation for Point-Cloud Creation

Nuno Granja Fernandes

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor at FEUP: Doutor Cândido Duarte

Supervisor at Infineon Technologies AG: Engenheiro Pedro Costa

July 16, 2018

Resumo

A indústria automóvel a nível mundial está a mudar rapidamente, investindo nas possibilidades criadas por avanços em várias tecnologias nas últimas décadas. Uma forte tendência nesta mudança é o aumento de automação na condução e o desenvolvimento de veículos autónomos. Os produtores de automóveis estão cada vez mais a incluir sistemas de apoio ao condutor nos seus carros, muitos dos quais inovadores na indústria.

Nesta área, a tecnologia LiDAR surge como uma ferramenta promissora para gerar representações precisas do ambiente que rodeia o veículo. Isto é um aspeto extremamente importante em muitos sistemas de apoio à condução e crucial no desenvolvimento de carros completamente autónomos. Contudo, a tecnologia LiDAR ainda é relativamente nova na indústria, de maneira que a construção de um sistema que usa todo o potencial desta tecnologia de uma forma economicamente viável ainda constitui um desafio que se encontra em avaliação por parte da maioria dos produtores de automóveis.

O objetivo deste projeto consistiu na conceptualização de um sistema de processamento digital de sinal para LiDAR, de viável implementação na indústria automóvel. O projeto foi desenvolvido em ambiente industrial, na Infineon Technologies AG, na fase de engenharia de conceção do desenvolvimento do produto.

Inicialmente, o sensor LiDAR utilizado foi meticulosamente analisado e todos os aspetos relacionados com restrições temporais e ligação com o sensor foram avaliados de maneira a que o sistema de processamento de dados LiDAR pudesse ser projetado com confiança. De seguida foi proposto um sistema que melhora a qualidade do sinal adquirido, que foi detalhado a nível de hardware. Este sistema cumpre todas as restrições temporais, de performance e de utilização de recursos impostos por uma aplicação de processamento de dados LiDAR. Este sistema foi então simulado para testar a validade do seu projeto. Esta simulação comprovou o funcionamento do sistema relativo à utilização de recursos e cumprimento de restrições temporais. Provou ainda a capacidade do sistema em melhorar substancialmente a qualidade do sinal adquirido. Por fim, visto que o sistema foi desenvolvido de forma escalonável, foram propostos outros sistemas que conectem e complementem o trabalho realizado pelo sistema desenvolvido. Isto prova como o sistema proposto serve como base para um potencial sistema LiDAR que cria uma representação visual detalhada do ambiente a partir dos dados provenientes do sensor.

Abstract

The automotive industry worldwide is rapidly changing, embracing all the new possibilities created by improvements in multiple technologies over the past few decades. A big movement is currently being made towards an increase in driving automation and autonomous vehicles. Car manufacturers are including more and more advanced driver-assistance systems (ADAS) in their cars, many of which are unprecedented in the automotive field.

In this area, LiDAR technology presents itself as a promising tool for providing accurate renderings of the environment surrounding the vehicle. This is a crucial aspect for many ADAS and is itself paramount in the pursuit of fully autonomous vehicles. However, LiDAR is still relatively new in the industry, so much so that building a system which takes advantage of its full potential in a cost-effective way is still under evaluation by most original equipment manufacturers (OEMs).

The goal of this project was to conceptualize and design a digital signal processing system for LiDAR data which is viable in the automotive industry. It was developed in the industrial environment, at Infineon Technologies AG, at the concept engineering stage of product development.

Firstly, the LiDAR sensor in use was meticulously analyzed and all the aspects regarding interfacing and time constraints were evaluated in order to confidently design the subsequent signal processing system. An overall data processing architecture was achieved and then a solution was proposed for a system that increases the quality of the acquisition, which was fully detailed at the hardware level. This system complies with all the constraints regarding performance, time requirements and available resources of a LiDAR application. The proposed system was then simulated in order to check the validity of its design. This simulation confirmed the correct behavior of the system with regards to the used resources and time processing constraints. Furthermore, it proved the success of the system in increasing the LiDAR sensor signal quality. Finally, since the system was developed in a way that allowed further expansions, other systems that could interface with this one were also proposed and evaluated. This shows how the developed system can serve as the basis for a full LiDAR data processing pipeline that builds an accurate visual representation of the surroundings of the sensor.

Acknowledgments

I would like to start by thanking my supervisor at FEUP, Professor Cândido Duarte, for giving me the opportunity to develop this project and for his much valuable guidance all throughout the development of this thesis. Likewise, to my supervisor at Infineon Technologies AG, Eng. Pedro Costa, for doing a great job in helping me transition into a new workplace and for his experienced and always incisive inputs. I would also like to extend this gratitude to Rahul Rajan for always taking the time to answer my many bothering questions and everybody else at Infineon who made me feel very welcomed.

I am extremely grateful for my amazing family and specifically my parents and sister for the encouragement and unwavering support they have shown my entire life. I would like to thank my girlfriend, Jacinta Soares, for how loving and caring she is and for being my best company, even from afar. And finally, I want to thank my close friends who I am extremely fortunate to have, for providing a great balance to my life between work and leisure.

Nuno Fernandes

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

ADAS	Advanced Driver-Assistance System
ADS	Automated Driving System
APD	Avalanche Photodiode
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
B	Byte
BMW	Bavarian Motor Works
bps	Bits per second
CFAR	Constant False Alarm Rate
CNN	Convolutional Neural Network
CRC	Cyclic Redundancy Check
CUT	Cell Under Test
DFU	Data Formating Unit
FFT	Fast Fourier Transform
FIFO	First In First Out
FLM	FIFO and Lane Management
FMCW	Frequency Modulated Continuous Wave
FPGA	Field-Programmable Gate Array
FPS	Frames per Second
IFFT	Inverse Fast Fourier Transform
k	Kilo
LiDAR	Light Detection and Ranging
lsb	Least Significant Bit
LSB	Least Significant Byte
LVDS	Low Voltage Differential Signal
M	Mega
MEMS	Microelectromechanical System
msb	Most Significant Bit
MSB	Most Significant Byte
OEDR	Object Event Detection and Reaction
OEM	Original Equipment Manufacturer
OPA	Optical Phase Array
PRF	Pulse Repetition Frequency
ROI	Region of Interest
RIF	Radar Interface
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
SPS	Samples per Second

SPU	Signal Processing Unit
SRAM	Static Random Access Memory
SVM	Support Vector Machine
TIA	Trans-Impedance Amplifier
ToF	Time of Flight
UDP	User Datagram Protocol
USB	Universal Serial Bus

Chapter 1

Introduction

The automotive industry plays a critical role worldwide, serves an enormous market, heavily impacts the development of entire countries and is a strong driver for technological innovation. Around 79.02 million cars were sold worldwide, and an estimated value of 81.57 million will be sold until the end of 2018 [1]. The greatest manufacturers and exporters in the world employ a great number of people and generate huge amounts of money every year. As an example, Germany as the biggest manufacturer and exporter in Europe, recorded in 2015 a turnover of 404 billion euros, which corresponds to 20 percent of the total German industry revenue and employed around 792.5 thousand people [2].

The typical car bought in 2018 surely incorporates better and more complex systems and more computation power than the Apollo 11 spaceship that landed the first two men on the moon in 1969. Over the past decade the introduction of technology in personal vehicles has increased dramatically and that growth has been exponential ever since the early 2000's [3]. For a long period of time automakers invested mostly in mechanical, performance-oriented features for the car that would for example increase the horsepower [4] but that trend has changed recently. There are several computers in a modern day car handling different subsystems. With the continuous reductions in transistor size and advancements in the semiconductor industry, it is now possible to use a computer of extremely small dimensions, low power consumption and high computation power to perform an unimaginable number of different tasks.

Some obvious symptoms of this change in the automotive industry are the growing introduction of electric and hybrid vehicles on the market and even the innovation of automatic gear shifting. Many other driving assistance systems are already well known and accepted by the masses and can be found in the vast majority of cars produced nowadays. GPS navigation aids the driver in choosing the best way to his destination, many times incorporating traffic information. Adaptive cruise control maintains the speed of the vehicle automatically and advanced cruise control will even reduce or increase the speed depending on the behavior of the vehicles around. Monitoring systems provide information to the driver about tire pressure, oil and water levels and many other parts of the car. However, the automation levels of some of these advanced driver-assistance systems (ADAS) go even further into newer systems, incorporated only in higher end,

more sophisticated vehicles and probably not so widely known. Systems like collision avoidance, automatic braking and lane departure warning, which increase the drivers safety by gathering information from the surrounding and actively influencing the driving to avoid obstacles or steer away from them. Many systems now directly control the driving, bypassing any human input like automatic parking systems, effectively moving up in the level of automation in driving.

Among all these systems, the necessity for scanning the proximity of a vehicle and detecting objects became essential. Many technologies have been researched for this purpose so far and are in use nowadays in systems that fulfill this necessity, like frequency modulated continuous wave (FMCW) radar, LiDAR, high definition cameras, sensor fusion methods that mix all these technologies and more [5–7]. In this document, the focus will be entirely on LiDAR technology. How the information is acquired and all the different aspects and challenges of processing that information will be addressed.

1.1 LiDAR

LiDAR stands for "Light Detection And Ranging". It utilizes light and the known constant speed of light to calculate distance. There are various ways of applying this principle, which depend on the type of light source being used, the simplest being the “direct time of flight measurement”. For this, a light source will repeatedly emit light pulses and the sensor will measure the time it takes for that pulse to hit an object and reflect back (see Figure 1.1).

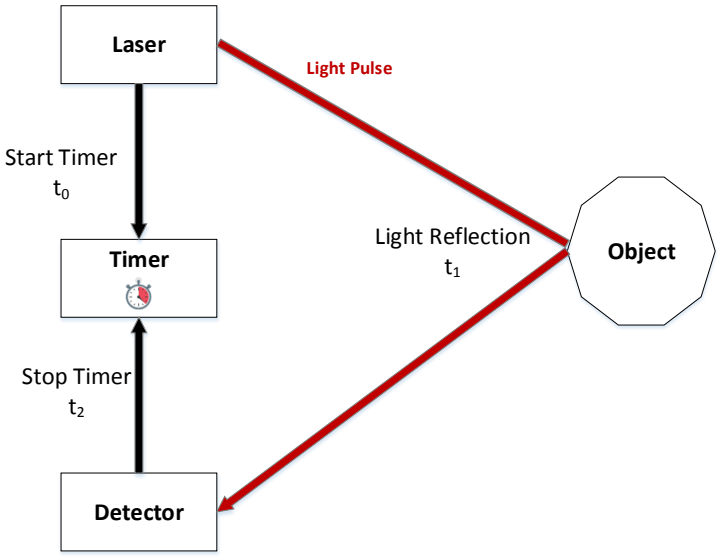

Figure 1.1: LiDAR direct ToF measurement – working principle.

As illustrated in Figure 1.1, a light source fires a light pulse into an arbitrary direction and a light sensitive detector, typically a photodiode, detects the light reflected from that pulse. The light source and detector must be equally distanced from the object.

The instant at which the light is emitted, t_0, and when its reflection is detected, t_2, are registered, resulting on a measurement of how long that light pulse was traveling: $t_2 - t_0$. That measurement is called time of flight (ToF).

Knowing light speed to be a constant, $c = 299792458\,\text{m/s}$, the distance between the sensor and the object is calculated with:

$$d = \frac{1}{2} \cdot c \cdot ToF \tag{1.1}$$

From this example it becomes apparent how LiDAR can be used to measure the distance from different objects in the vicinity of the sensor. Considering now not only the ToF measurement but also the amount of light reflection that was detected, other interesting conclusions can be drawn from the LiDAR data such as the likelihood of an object being there, reflectivity and color of the object and even its shape.

In short, the working principle of LiDAR is to measure distance based on the ToF of a light pulse. However, in order to use this technology to build a picture of the scene around the sensor, light has to be fired in many different directions. There is a considerable amount of ways of achieving this, which will be further explored in chapter 2, typically making use of 3 distinct components: transmitter, detector and scanner.

LiDAR sensors provide very good distance resolution, when compared to radar technology for example [8], and for that reason have proven useful in a number of applications relating to geography, like creating high resolution digital elevation maps for studying the surface of the Earth and also for mapping forests and individual trees heights, which is very helpful in forestry management. LiDAR technology has applications in physics and astronomy like measuring, with high precision, the distance from the moon and other celestial bodies, and even in Law enforcement LiDAR technology is often times used as "LiDAR speed guns", which are used to measure the speed of a vehicle and determine if they are moving above the allowed limit.

It also has some characteristics, not observed in radar, often preferable in the automotive industry like a wider field of view [8] and the capability of detecting the shape of an object, which is valuable for object detection software.

1.2 LiDAR Technology in the Automotive Industry

LiDAR technology has already shown its value in the robotics industry and consumer electronics applications finding use in thousands of different applications. However, over the last few years the automotive market found use for LiDAR technology for its high precision and reliability. Some vehicles like the Robo-Taxi [9] or Waymo's autonomous vehicles [10] already use LiDAR technology to achieve high levels of autonomy in driving.

Until recently, most LiDAR applications in the automotive market mostly used mechanical rotating parts or brushless DC motors to scan the environment. However, LiDAR technology offers many options for scanner components and varieties of laser. Besides using electric motors

to move a mechanical scanner, there is also the possibility of using solid state scanners like microelectromechanical systems (MEMS) to oscillate a mirror reflecting light in multiple directions, optical phased arrays (OPA) and others. There is also a choice to be made in the type of laser used in the system, being that a light source could be used that continuously emits light and uses indirect methods to measure the ToF or a pulsed laser repeatedly firing light pulses in different directions.

All these options influence everything on the system from the size of the data to the overall performance of the system. For this reason, the possibilities are vast when it comes to creating a LiDAR system and all the big original equipment manufacturers (OEMs) for the automotive industry are now investing in different LiDAR sensors and researching better ways of creating more accurate, efficient and cost-friendly LiDAR systems [11]. It is estimated that the global LiDAR market size in 2016 was of 1058.4 million dollars and is projected to grow to 5842.37 million dollars in 2024 [12]. It also is estimated that the market size for LiDAR designed into ADAS in the USA in 2017 was of 107 million dollars and expected to grow to 1.7 billion dollars in 2023 [11], which is remarkable growth.

There is still a lack of consensus in the automotive industry about the best way of employing LiDAR technology. Most OEMs currently use a combination of cameras and radar technology to perform object detection but other big automotive entities like BMW choose to use LiDAR and cameras for that matter and Tesla opted out of LiDAR technology in their vehicles. Implementations using pulsed lasers, flash lasers that illuminate the entire scene or lasers that continuously illuminate the environment imply different values of power consumption, dimension of the sensor and data processing system, memory requirements and degree of complexity of the data processing algorithms. LiDAR is still too recent in the automotive industry for all these variations to have been fully implemented and tested to say without a doubt what is the best way of using LiDAR in a vehicle, if there is one. Despite the lack of consensus, the interest in LiDAR has been, and still is, growing immensely [12, 13].

1.3 Problem Statement

Acquiring the data generated by a LiDAR sensor and sending it to a microcontroller for processing constitutes a challenge for any current state of the art microcontroller in the automotive industry. This is due to the large amount of data generated, the strict time restrictions of a LiDAR application and consequently the required resources. Balancing these LiDAR requirements with a cost-effective system, applicable in the automotive industry, constitutes the fundamental problem of this project.

Figure 1.2 illustrates the generic sensor concept in use for the development of this thesis. It consists of a 1D MEMS mirror, a pulsed laser, a 1D photodiode sensor array and the necessary lenses to both spread the emitted light pulses and focus the reflected light towards the sensor array. In order to ultimately produce an accurate 3D visual representation of the environment, the scene in front of the sensor must be interpreted as a grid of size $N \times M$, as illustrated in Figure 1.2, that

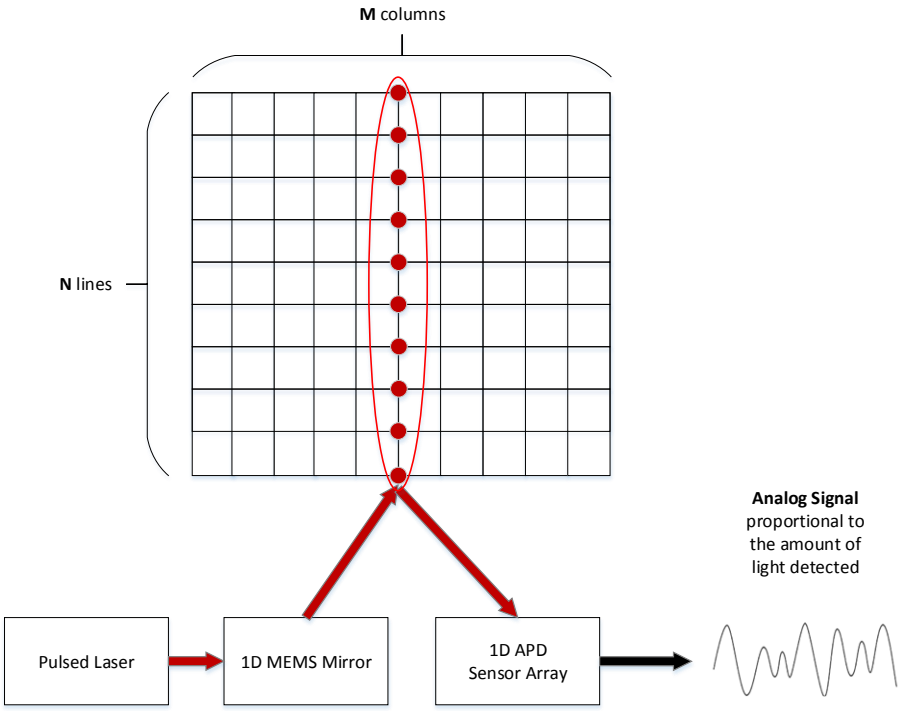

Figure 1.2: Lidar sensor and grid schematic.

covers the entire desired field of view of the sensor and in which each point of the grid is a pixel. A number of samples, depending on the desired performance of the LiDAR system, which will be explored in chapter 3, is acquired for each pixel. The grid must be scanned in its entirety and all the data must be gathered and processed. A visual representation of the environment is only possible after acquiring all the samples from all the pixels on the grid. Typically, the amount of data constituting a grid is very large, implicating that LiDAR systems are clocked at a high frequency and dispose of large memories, increasing the overall cost of the system. It is not a trivial task to conceptualize and design a system that finds an optimal balance between these aspects.

As represented in Figure 1.3, each photodiode in the sensor array outputs an analog signal corresponding to the amount of reflected light measured and that signal is amplified and discretized by an ADC. Each ADC connected to the array will produce samples at a high rate every time a laser pulse is fired and the laser must fire light pulses at a high frequency in order to quickly scan the entire frame. This implies that the data is quickly generated in short bursts of time, requiring high data transfer speeds for sending it to a microcontroller and processing it. For this reason, it is challenging to find the best way of interfacing the sensor and the microcontroller without jeopardizing the performance of the system.

Furthermore, a choice needs to be made on what signal processing algorithms shall be applied and a digital system that implements those algorithms must also be conceptualized and designed. These systems increase the delay between scanning the grid and producing the result of that scan, affecting the rate at which frames are produced. These aspects must be taken into account along

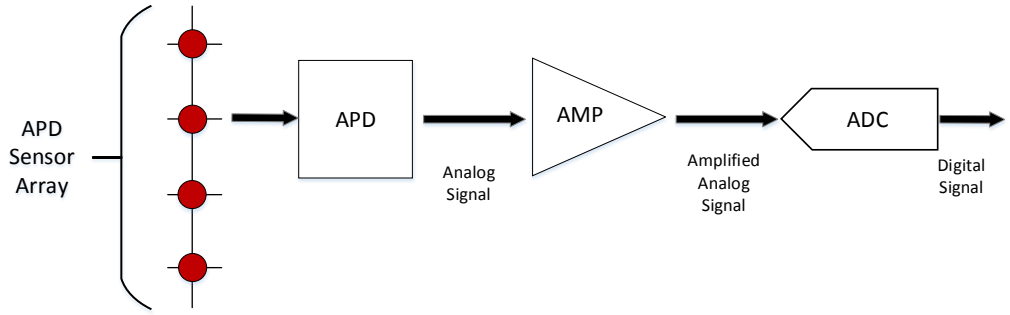

Figure 1.3: Electronic components connected to each photodiode.

with the performance requirements of the system and its cost. After these algorithms are applied, a representation of the environment is ready to be used by object detection software. In LiDAR systems, the set of points displayed in a 3D referential resulting from the processed acquired signals is called a point-cloud representation. This representation intends to create an accurate visual depiction of what the sensor measures. The processed point-cloud data must be stored in memory in a comprehensible way to facilitate interfacing between the microcontroller and the following software navigation systems, which increases the complexity of the signal processing system.

1.4 Goals

This project was developed in an industrial environment, at Infineon Technologies AG, at the concept development stage of the product, which largely consists in carefully analyzing the full scope of the application and proposing a realistic and implementable solution that meets the needs of the application.

The goal of this project is to conceptualize and design a system for processing LiDAR data and producing a point-cloud representation of the environment. This system must answer to performance, cost and real-time requirements. All the external variables that influence the system and how they impact its performance must be duly evaluated and it must be specified exactly under what circumstances the proposed system operates. The algorithms used to produce the point-cloud representation must be clearly described and how those algorithms are implemented in a digital signal processing system must be detailed as well. Plus, it must be implementable with the LiDAR MEMS sensor concept by Infineon Technologies and making use of the necessary components in order to interface the AURIX microcontroller with the LiDAR sensor.

1.5 Methodology

In order to achieve the desired goals, the technology involved in this application must be thoroughly analyzed. Other system architecture proposals for processing LiDAR data shall be studied

to define a proper starting point for this project.

The LiDAR sensor will be examined and how it operates must be fully understood. This includes knowing how the grid data is obtained and what viable options there are to send the data to the signal processing system. Then, the data processing algorithms required to produce a point-cloud from the LiDAR data will be analyzed and defined.

The interfacing challenges between the proposed system and the sensor will be addressed and a solution for reading the LiDAR data will be put forth, accounting for all the LiDAR application requirements. The signal processing system will then be detailed at a hardware level in a comprehensible way and the design of the system will be simulated to be properly analyzed and tested.

1.6 Thesis Structure

This document will be structured with the following chapters:

2. State of the Art Review − This chapter will provide knowledge about the technology used in the system, which is necessary to understand the work developed in this thesis. It shall also explore other LiDAR data processing systems that relate to this project.

3. Proposed System Overview − This chapter describes in detail the initial steps taken to achieve the generic architecture of the proposed solution. It begins by contextualizing the problem, showing how the LiDAR sensor operates and what challenges ensue for acquiring and transferring the LiDAR data, as well as proposing and evaluating some solutions to those problems. It then describes how the LiDAR data can be subjected to signal processing algorithms to increase the signal to noise ratio (SNR) and which will be implemented in the developed system. Finally, the proposed peak detection algorithm is described and in the concluding remarks of the chapter, an overview of the proposed system is provided.

4. Computation Model − This chapter presents the proposed system for processing the LiDAR data, increasing its SNR by performing signal averaging, in a detailed hardware description of the system. Initially, an overview of the entire designed system is provided along with the specific conditions under which it operates. Then, the hardware components used to interface the sensor with the microcontroller are detailed and an analysis of the performance of the system is completed. Afterwards, each individual component of the proposed system is exposed and finally, all aspects regarding memory are presented, such as mapping the grid data in static random-access memory (SRAM) and how memory addressing occurs.

5. Simulation Results − A simulation of the designed system is described. From this, performance measurements are analyzed comparatively to how the system was designed to perform. Furthermore, the simulation is used to evaluate if the system functions as intended, performing the signal averaging technique and mapping the data in memory correctly. Finally, the following steps in creating a point-cloud representation, included in the proposed

solution, are performed on the averaged signals, illustrating exactly how a point-cloud is obtained from the LiDAR sensor data.

6. Conclusions – All the relevant concluding remarks from this project are presented in this chapter. Moreover, in the future work section, other possible algorithms for LiDAR data processing are mentioned and important aspects of the following systems are specified. A possibility for designing a peak detection system is then introduced, interfacing with the one described in chapter 4.

Chapter 2

State of the Art Review

LiDAR technology relates to a various number of fields both in and outside the automotive industry. This chapter will provide deeper knowledge on some of the work already developed using LiDAR, the tendencies currently observed in the automotive industry towards ADAS and autonomous driving, the technology involved in a LiDAR sensor and some current digital signal processing architectures for LiDAR data.

2.1 Automated Driving

2.1.1 Autonomous Driving

Autonomous driving is a young, fast growing and promising industry. Every day it becomes clearer that autonomous vehicles will be massively introduced in the market and many countries and several states in the U.S.A have already started implementing rules to regulate the introduction and circulation of autonomous vehicles on the road. Major automotive companies already plan for the introduction of autonomous vehicles on the roads and are currently producing many automated driving features like automated parking, traffic jam pilot systems, highway pilot systems, etc [3].

Two different schools of thought emerge when talking about the automation levels of driving a car and no final conclusion has been reached yet [3]. The first one argues that a steady increase of automation in driving, integrating new techniques with the emerging technology until full autonomy is reached, is the best way. The second one defends that there must be a leap from semi-autonomous to fully autonomous somewhere along the progression. Especially considering that some systems work by autonomously driving the car yet relying on the driver to intervene in case of object event detection and reaction (OEDR), which might be ineffective given that when a driver is removed from the driving process for a long period of time he will not be as quick to react.

Many advantages emerge with the increase of autonomous vehicles in circulation: fewer collisions would occur due to the elimination of the human error in driving, those incapacitated of driving would be able to have a personal vehicle, commutes would generally be smoother and more comfortable and the users could make use of the time to be productive. Plus, the traffic

flow control would increase, reducing the amount of time spent on traffic and also making it more appealing for car-sharing systems to emerge [3].

However, the shift towards autonomous vehicles still faces some difficulties because even though technology might currently be very advanced, it still takes a lot of work and resources to turn it into an industrial product that can compete in the automotive market. Also, the cost on initial production on these kinds of systems would be relatively high. Even though safety would ultimately increase with the introduction of these sophisticated ADAS, it is not certain that it would not worsen in the beginning stages and there is still a lot of resistance offered by the general public.

2.1.2 Automation Levels in Automated Driving

In autonomous driving, six distinct levels of automation were defined based on functional aspects of the technology implemented. These definitions were put forth by the Society of Automotive Engineers (SAE) in 2014 with the goal of providing a common terminology for automated driving [14].

In Table 2.1 a simple and clear definition of each level of automation is provided. These definitions rely on the following actions:

- execution of steering and acceleration/deceleration
- monitoring of driving environment
- fallback performance of dynamic driving task
- system capability – driving modes.

An important observation that should be made about table 2.1 is that there is a clear distinction between levels 0-2 and 3-5, which is that up until level 2 the human driver is the one performing the dynamic driving task whereas in level 3 and upwards the automated driving system (ADS) performs that task. The dynamic driving task is comprised of the operational and tactical aspects of driving, where the operational side is steering, accelerating, braking and monitoring the vehicle and roadway and the tactical side is the one responsible for deciding waypoints and the destination [14].

2.1.3 Pre-Crash Safety Systems

Many new ADAS are appearing that can identify a collision milliseconds before it happens and make quick adjustments to diminish the impact and damage of the crash [15]. These systems contain information like:

1. Distance from the obstacle
2. Time at which the collision will occur

SAE Level	Name	Description	Steering and Acceleration	Monitoring driving environment	Fallback performance of dynamic driving task	System capabilities
0	No automation	Full-time performance of dynamic driving task by the human driver.	Human	Human	Human	N/A
1	Driver Assistance	Execution of either steering or accelerating by a driver assistance system.	Human and system	Human	Human	Some driving modes
2	Partial automation	Driver assistance system performs steering and acceleration using information about the driving environment.	System	Human	Human	Some driving modes
3	Conditional automation	System performs all dynamic driving aspects, expecting that the human driver will respond adequately when asked to intervene.	System	System	Human	Some driving modes
4	High automation	System performs all aspects of dynamic driving even if the human driver does not respond appropriately to a request to intervene.	System	System	System	Some driving modes
5	Full automation	Full-time performance of all dynamic driving aspects by the system under all the circumstances that a Human could perform in.	System	System	System	All driving modes

Table 2.1: Automation levels definition in automated driving

3. Information about the vehicle and obstacle (absolute speed, relative speed and direction of the motion)
4. Estimate of where the collision will take place
5. Decision on whether the impact will actually occur.

A great number of these systems use radar or LiDAR sensors. Radar sensors can directly acquire the speed and distance of the obstacle but LiDAR can provide high resolution information of the distance from objects with wide amplitude ($\approx 160°$) and at great distances ($\approx 100\,\text{m}$) [15]. This way, many configurations utilizing multiple sensors simultaneously are being researched in order to increase reliability and robustness of environment recognition [5–7, 16, 17].

Even though LiDAR systems were rarely used because they performed poorly in unfavorable weather conditions, this problem has been partially rectified recently using "multi-echo" measurements [15], which consists in making multiple measurements in the time of the emission of one light pulse. LiDAR sensors still remain relatively expensive when compared to other sensors already being used on the market [18].

2.2 Sensor Fusion

As mentioned previously, a growingly popular choice among automotive manufacturers is to build systems that make use of multiple sensors, which in some way compensate for each other's flaws. For example, while a LiDAR sensor is capable of providing accurate measurements of distance with good resolution they lack some information that a camera can gather like color and good

shape description of the detected objects. When coupling LiDAR with Radar, although LiDAR gives a better 3D visual representation of the environment, covers greater distances and a wider field of view [8], it also generates a considerable larger amount of data and does not provide information about the velocity of the objects like radar does. Also, in adverse weather conditions like snow and fog, the performance of the LiDAR sensor drops significantly while radar still performs the same [8].

A very common combination is the use of LiDAR sensors with cameras [6, 7, 16, 17]. LiDAR offers good depth resolution in obstacle detection but it is much easier to use cameras in object classification, using color to identify signs and traffic lights, applying computer vision algorithms. One way of mixing these two sensors is by first using the information from the camera to find regions of interest (ROI) where it might be likely to find a vehicle or other object and then use the LiDAR data, which can provide more information like the amount of light reflected from those regions, their distance with greater resolution and even their shape to help better classify the object. This and other similar sensor fusion approaches increase the complexity of the system because there must always be a subsystem responsible for accurately adjusting the coordinates from data originating from one sensor to the coordinates from other sensors, otherwise there is no way of utilizing the data from both sensors to extract information about the same point in space [6, 7, 16, 17].

Furthermore, machine learning algorithms like support vector machines (SVM), adaboost classifiers and convolutional neural networks (CNN) are regularly used for image classification so the information resulting from mixing multiple sensors must be preprocessed and adapted to valid image features that can be used as inputs for these algorithms.

2.3 LiDAR Sensors

LiDAR sensors are typically constituted by a transmitter, detector and a scanner device, although a scanner-less LiDAR sensor is also a viable option.

For the receiver, which detects the reflected light, different photo sensitive devices can be used like Indium Gallium Arsenide (InGaAs) photodiodes, single-photon avalanche diodes (SPAD), which are a type of avalanche photodiodes (APD) that generate larger currents than normal photodiodes but require a higher amount of incident energy (more light) to trigger a small-duration, high-value current. Silicon Germanium (SiGe) photodiodes and also single photomultiplier (SiPM) devices are also frequent choices.

There are two kinds of lasers that can be utilized as a transmitter: continuous wave lasers (CW lasers) and pulsed lasers. A continuous wave laser emits a continuous laser beam in which the beam duration and intensity can be controlled [19]. FMCW lasers are a kind of continuous wave lasers in which the frequency of the emitted light changes with time. Pulsed laser, on the contrary, emits light in short duration bursts instead of one continuous beam and also allows for control over beam duration and intensity [19]. There are many available options of lasers that can

be employed as pulsed lasers like fiber lasers, erbium lasers, double heterostructure laser diode, quantum cascade laser diodes, and many more.

2.3.1 Flash LiDAR

Flash LiDAR is used when there are no scanning components in the LiDAR sensor. Therefore, the laser must be able to illuminate the entire desired field of view in order to measure the reflected light for the whole frame. Adequately designed lenses are used so that the light is spread for a wide horizontal field of view and a comparatively smaller vertical field of view, e.g. 60° horizontal field of view and 16° vertical field of view.

The light is fired and spread for the entire frame, reflects back from some objects, and a lens redirects the reflected light towards a photodiode sensor array. The detected signals are digitalized and processed in order to create a 3D accurate representation of the environment.

This solution poses some restrictions because, firstly, a 2D sensor array must be used so that each photodiode on the array detects the light reflected from a distinct position in the frame. Secondly, since all the information from the frame is gathered in the same instant, a large amount of data is generated in a short period of time, which requires resources to store and process.

2.3.2 Optical Phase Array

An optical phase array is a device frequently used in optical communications. It uses Snell's law to steer passing light beams in different directions. It does so by controlling the optical properties of the material in which the light is propagating, in turn controlling the angle at which the light beam leaves the device. This can be done without moving parts, electrically controlling those optical properties.

Optical phase arrays are often used in LiDAR technology as a way of making a sensor capable of scanning segments of the frame without any mechanical moving parts.

2.3.3 MEMS Mirrors

It is common practice in LiDAR technology to couple the use of a Laser with a MEMS mirror because this technology typically provides a wide optical deflection angle, high mirror reflectivity and low power consumption [20]. This technology also proves to be more compact and robust when compared to fully mechanical scanning mirrors. This technology is used so that the laser can steadily fire light pulses directed to the MEMS mirror, which in turn oscillates at a high frequency making the light pulses reflect in many different directions, as illustrated in Figure 2.1, obtaining information from a wider portion of what surrounds the vehicle.

There are single-axis and dual-axis MEMS mirrors used to reflect light in 1 or 2 dimensions of space [20]. 1D MEMS mirrors are composed of a mirror surrounded by a coil and suspended by a torsion bar, illustrated in Figure 2.2, which can bend to a certain extent, defining the mechanical deflection angle and consequently the optical deflection angle seen in Figure 2.1.

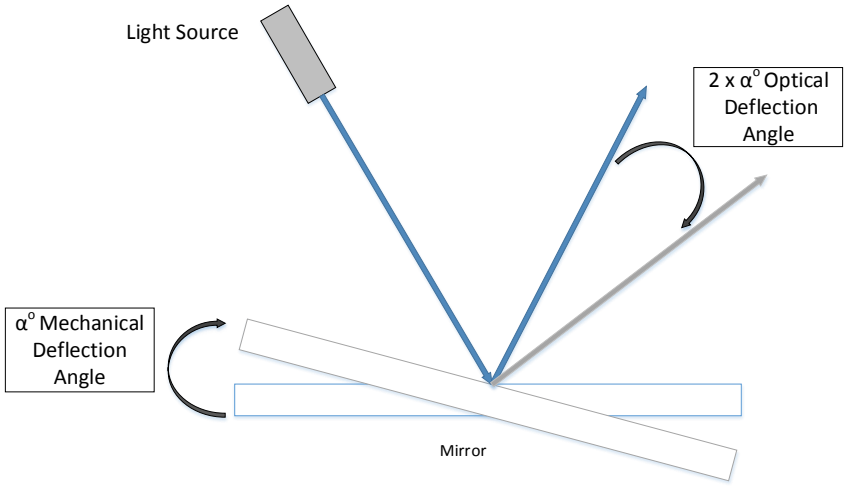

Figure 2.1: Oscillating mirror spreading the light pulses.

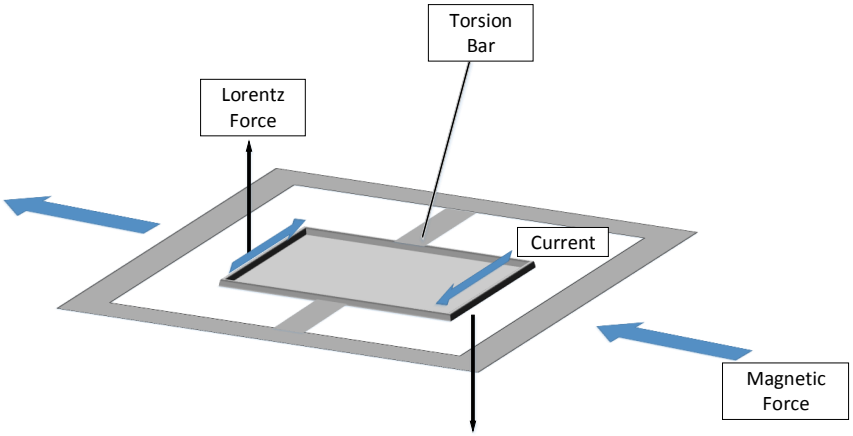

Figure 2.2: MEMS mirror and forces acting on it.

An electric current present on the coil is subject to a perpendicular magnetic force, which in turn exerts a force – *Lorentz* force, illustrated in Figure 2.2, that turns the mirror. According to *Lorentz* law, this force is proportional to the magnitude of the magnetic field and the electric current and perpendicular to both of them.

When the *Lorentz* force acts on the mirror and the mirror turns, a force appears contrary to the *Lorentz* force, due to the elasticity of the torsion bar. The electric current is then diminished, consequently diminishing *Lorentz* force, which allows the elastic torque to spin the mirror in the other direction. This way, the value of the electric current changes in a way that the mirror oscillates resonantly at a frequency defined by its mass, spring constant and structure obtaining big deflection angles with small currents [20].

2.3.4 2D MEMS Mirror

Besides single-axis MEMS mirrors, a dual-axis MEMS mirror scanning architecture is also a possibility for LiDAR sensors.

A dual-axis mirror, unlike the single-axis, can oscillate around a vertical and a horizontal axis simultaneously. It operates under a very similar principle as the 1D MEMS mirror, except that the outer frame of the mirror observed in Figure 2.2 is also suspended by a torsion bar, perpendicular to the inner one, subjected to a magnetic force and current, making it oscillate around this new horizontal torsion bar.

This way, light directed towards the mirror, can be steered in any direction in space, giving the LiDAR sensor many new options in ways of scanning an entire frame.

However, this technology is typically larger, more expensive, has a lower field of view and is harder to control than a single-axis MEMS mirror [21].

2.4 LiDAR Signal Processing

Prior to the development of this project, some significant progress was made towards the development of a LiDAR system. This section will detail some of the more relevant aspects in that progress. Namely, analysis of different possible types of LiDAR sensors, a study on the variables involved in the sensor, interfacing between the sensor and a microcontroller and further signal processing both on the microcontroller and in a separate system.

2.4.1 LiDAR Demonstrator

A project developed at Infineon Technologies led to a functional LiDAR demonstrator – CES2018 demonstrator – using the AURIX microcontroller. This system is comprised of 5 modules as represented in Figure 2.3:

- detector module
- controller module
- power supply module
- laser module
- MEMS module

and it also includes the following on-board interfaces:

- radar interface (RIF) for forwarding the averaged data to the microcontroller. This component will be described in greater detail further in the document.
- serial peripheral interface (SPI) for writing on the field-programmable gate array (FPGA) registers and controlling other peripherals.

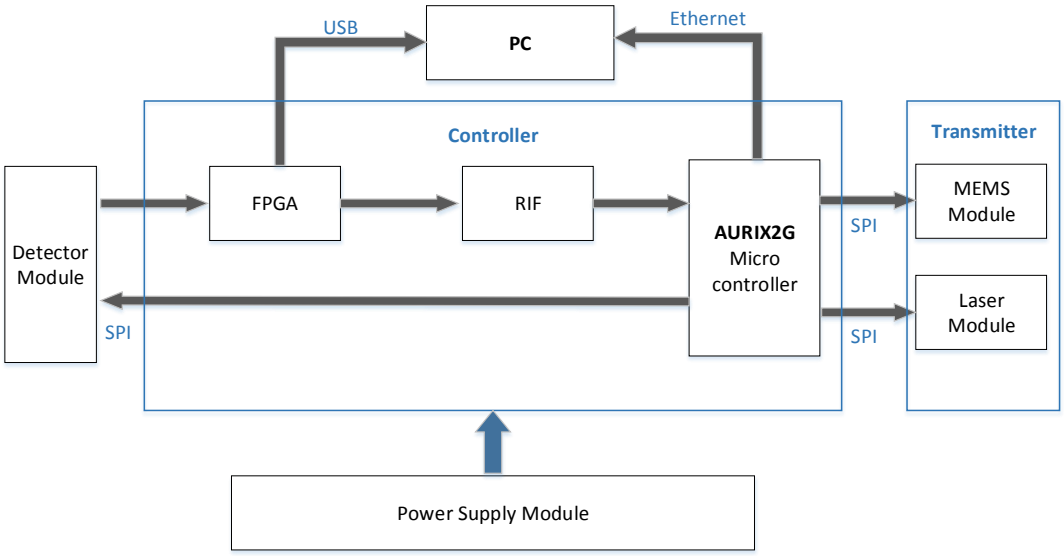

Figure 2.3: Infineon Technologies CES2018 LiDAR demonstrator.

and the following PC interfaces:

- ethernet for sending the final point-cloud representation from the microcontroller to a local computer
- universal serial bus (USB) for the FPGA to send the raw LiDAR data to the computer.

The Detector module uses an APD sensor array for detecting the light emitted and converts that into a digital signal sending it to the FPGA. The FPGA is where the signal averaging takes place and the result is delivered to the microcontroller through the built-in RIF.

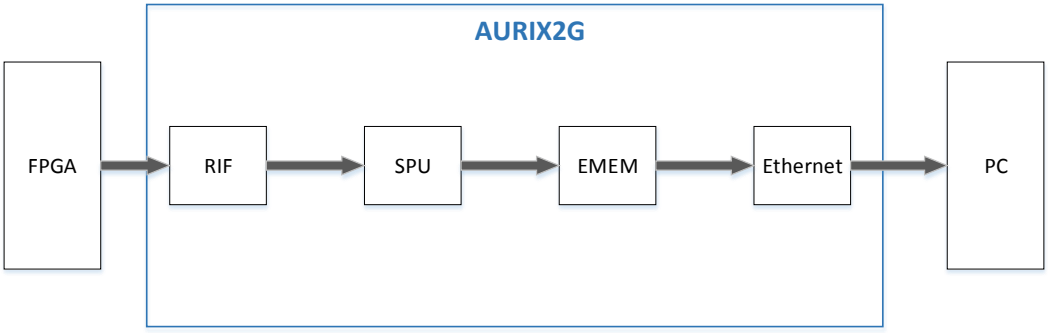

Figure 2.4: Data path and formats in CES2018 demonstrator.

As represented in Figure 2.4, RIF delivers the LiDAR samples to the signal processing unit (SPU) where match filtering, thresholding and peak detection take place and the necessary data is

stored in the internal memory, which then uses a user datagram protocol (UDP) over an Ethernet connection to deliver the point-cloud to the computer.

2.4.2 LiDAR Signal Processing Using the AURIX Microcontroller

On an approach for interfacing the LiDAR sensor with the AURIX microcontroller, a system with the architecture presented in Figure 2.5 was proposed. In this approach, all data processing algorithms besides averaging are performed on the SPU of the microcontroller, which disposes of 4 MB of SRAM memory.

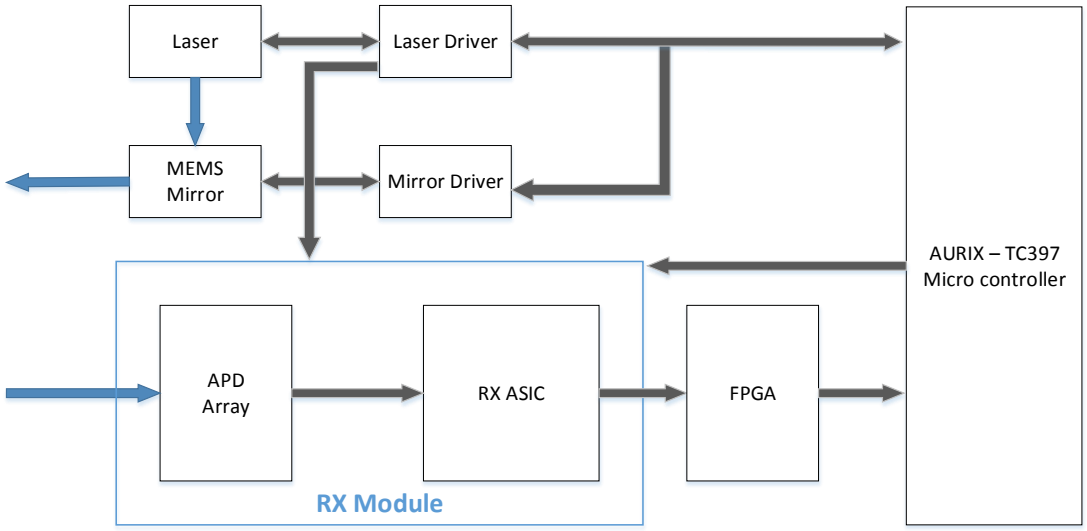

Figure 2.5: LiDAR Chipset using AURIX2G.

In this architecture, the microcontroller guarantees synchronization between the laser pulses and the MEMS mirror angle position and oscillation. The FPGA is included as an optional block for processing the raw data incoming from all the APDs, forming histogram representations of the data (to be explored further) and averaging them at high rates. The RX Module incorporates the APD array for light detection and the RX application specific integrated circuit (ASIC).

The RX ASIC can be decomposed in subsystems that amplify the signal incoming from the APDs with use of trans-impedance amplifiers (TIA) that convert the current signal from the photo-diodes to an amplified voltage one, the ADCs that discretize the signal into an arbitrary number of bits representing the intensity of the light captured by the APDs and a buffer for allowing a lower data transmission rate from the RX Module to the FPGA.

Figure 2.6 shows the block diagrams of the system that simulate the acquisition and signal processing of the LiDAR data, included in both the RX Module and the AURIX2G.

The cross correlation of the signal with the emitted light pulse – Match Filtering – is performed on the SPU of the microcontroller, even though an architecture where this would take

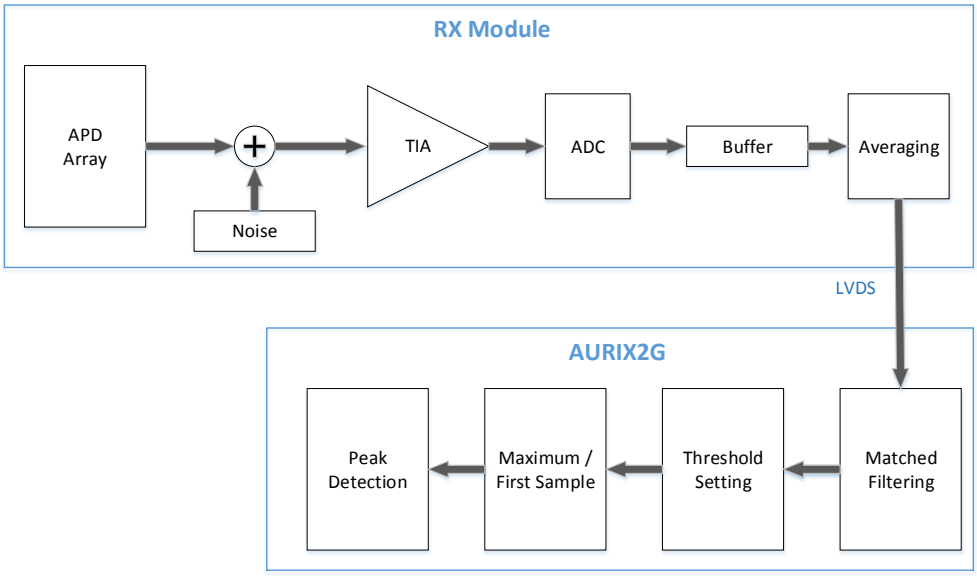

Figure 2.6: RX ASIC and AURIX2G subsystems for LiDAR data processing.

place in the ASIC was evaluated as well. In order to perform the match filtering on the incoming samples, the following steps are taken:

1. The Fourier transform of the emitted signal – a 10-15 ns wide Gaussian pulse – is stored in the configuration memory of the SPU.
2. When it comes the time to sample the incoming light, a fast Fourier transform (FFT) is performed on the time signal.
3. The Fourier transforms of both the emitted light pulse and the received signal are multiplied.
4. An inverse Fourier transform is performed on the resulting signal, completing the filtering process.

Still on the SPU, a constant false alarm rate (CFAR) algorithm is applied, which results in a threshold value to be applied to each sample indicating if it corresponds to an object detection or simply to noise. Finally, the peak detection, using the resulting threshold value, is performed.

This study concluded that all approaches for direct interfacing between ASIC and AURIX2G require at least Averaging on chip (FPGA or ASIC). All implementations require from 3 to 6 AURIX2G microcontrollers to process all the data. It also concluded that the possibility of splitting between short and long range LiDAR reduces the number of required microcontrollers.

2.4.3 LiDAR Signal Processing on FPGA or ASIC

In order to deal with the high volume of data incoming from the sensor, implementations where the sampling, acquisition and processing would be performed by an FPGA or ASIC.

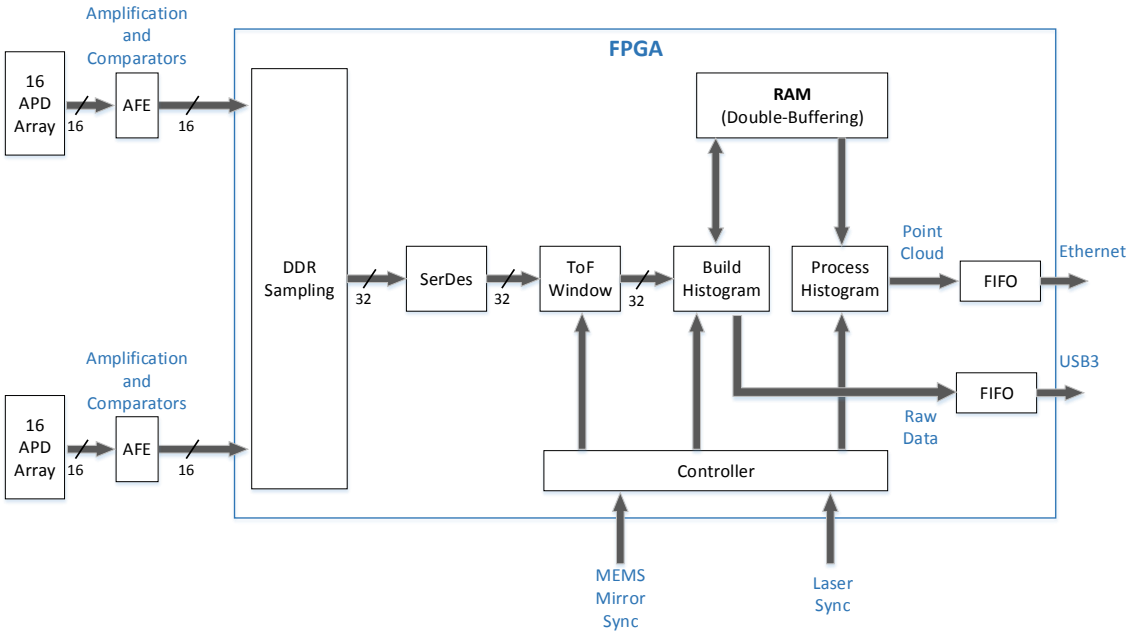

Figure 2.7: LiDAR signal processing overview – sampling on FPGA.

The illustrated system on Figure 2.7 makes use of an FPGA to both sample the incoming analog signal and produce the point-cloud representation.

Two 16-channel APD arrays generate signals that go through amplifiers and then connect to the FPGA through a low voltage differential signal (LVDS) protocol (also known as TIA/EIA-644). The "DDR Sampling" block outputs 32 serial lines, one for each APD, to a de-serializer that then outputs thirty two 10-bit wide buses where each 10 bits corresponds to a sample. The ToF block accumulates samples for 2 μs and then adds them in the "Build Histogram" subsystem, making use of a RAM block of up to 32 MB. Filtering and peak detection algorithms are applied in the "Process Histogram" subsystem, on the resulting histogram representation of the data.

The Controller module receives information about the mirror position and the time when light pulses are fired and uses that information to start and stop sampling for the duration of 2 μs and controls when the histograms should be averaged, by how much and when the filtering and peak detection can take place. Both the point-cloud representation and the raw LiDAR data are sent to a local PC.

In this second architecture, seen in Figure 2.8, the acquired signal is transmitted to the FPGA through an LVDS link for histogram building, averaging and point-cloud representation. That representation is sent to a microcontroller and a local computer for analyzing the result.

This architecture differs from the previous one since there is an ADC for each APD that acquires and sends the LiDAR samples through an LVDS connection. In a similar fashion, the FPGA combines the data building a histogram representation, filters and performs peak detection on them, outputting both the raw data and the point-cloud. The Controller block has now an extra

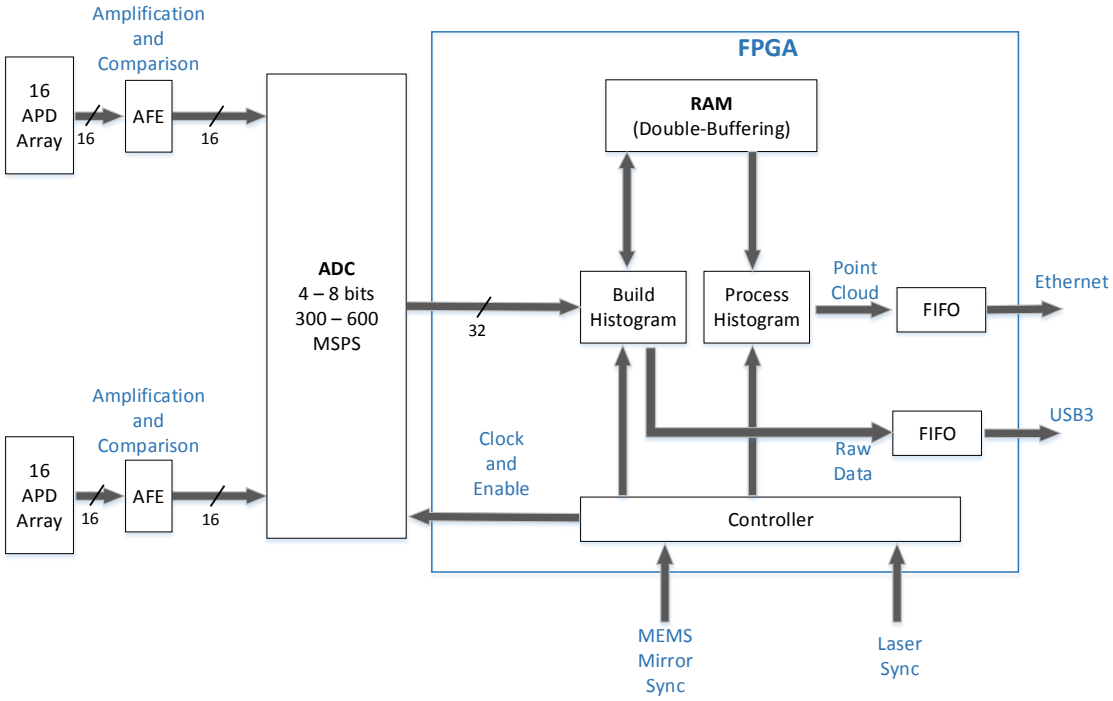

Figure 2.8: LiDAR signal processing overview – sampling by an ADC and signal processing on FPGA.

"Enable" output for acquiring samples only for the desired amount of time.

2.4.4 The RX ASIC Concept

The RX ASIC and DSP frontend concept for the LiDAR system was designed and proposed with the ASIC architecture as seen in Figure 2.9.

In this concept, a 16-APD sensor array detects light and sends the current signal to the RX ASIC in which a pixel amplifier block stores and amplifies the detected current value and converts it into a voltage one. Afterwards, a 4-bit ADC samples the analog signal, which is stored in a first-in first-out (FIFO) buffer. This FIFO is used to absorb the read/write data rate difference since the ADC data cannot be serialized and read out through the LVDS links in real time. The Laser fires light pulses periodically meaning that the light reflections will be acquired in short periodic bursts, which will need to be stored and then use the time until the next acquisition to be read out, as illustrated in Figure 2.10.

Each ADC samples 2048 4-bit samples for each light pulse at 1 GHz sampling frequency during 2.048 μs and writes them to the FIFO. Afterwards, the stored bits are read out through a serial LVDS connection at a rate of 1 Gbps, which adds up to a total time of 10.24 μs between two successive laser pulses.

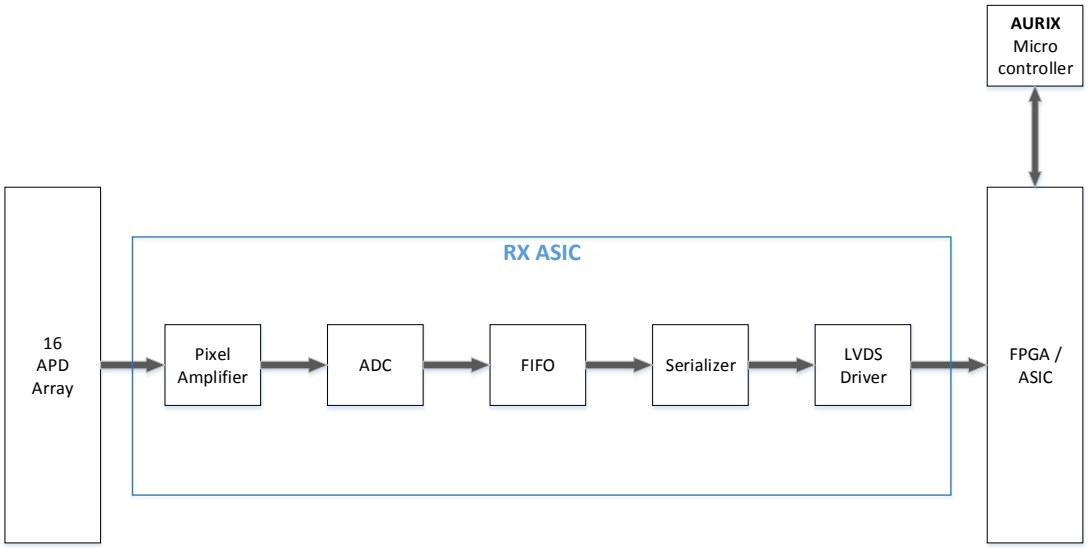

Figure 2.9: RX ASIC and DSP frontend proposal.

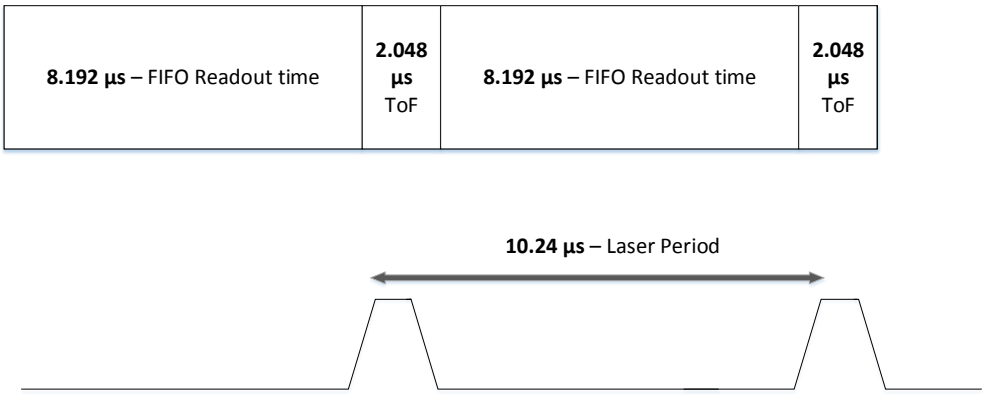

Figure 2.10: Time diagram of storing and read out times in the FIFO.

The FIFO architecture, presented in Figure 2.11, consists of 8 FIFO segments of size 256×4 bits. This way, the "Clock Generator and Controller" block guarantees that the ADC stores the 4-bit samples sequentially in each segment, one at a time. Thus, samples are shifted in each segment at a frequency of 125 MHz. The readout frequency from each segment is $\frac{1000}{32} = 31.25$ MHz since the multiplexer picks one bit at a time for serial transmission also sequentially, one segment at a time.

Furthermore, an on-chip adder system was proposed as well, which, as illustrated in Figure 2.12, is implemented between the ADC and the FIFO.

Its size is described as $D \times M$ in which D is the number of samples acquired for an emitted

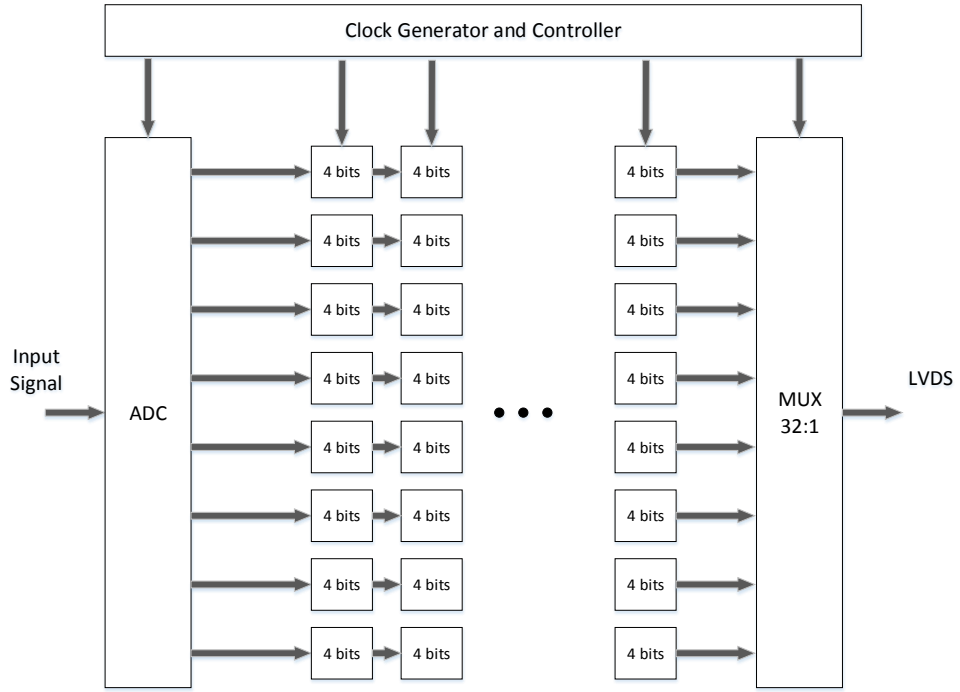

Figure 2.11: RX ASIC FIFO architecture.

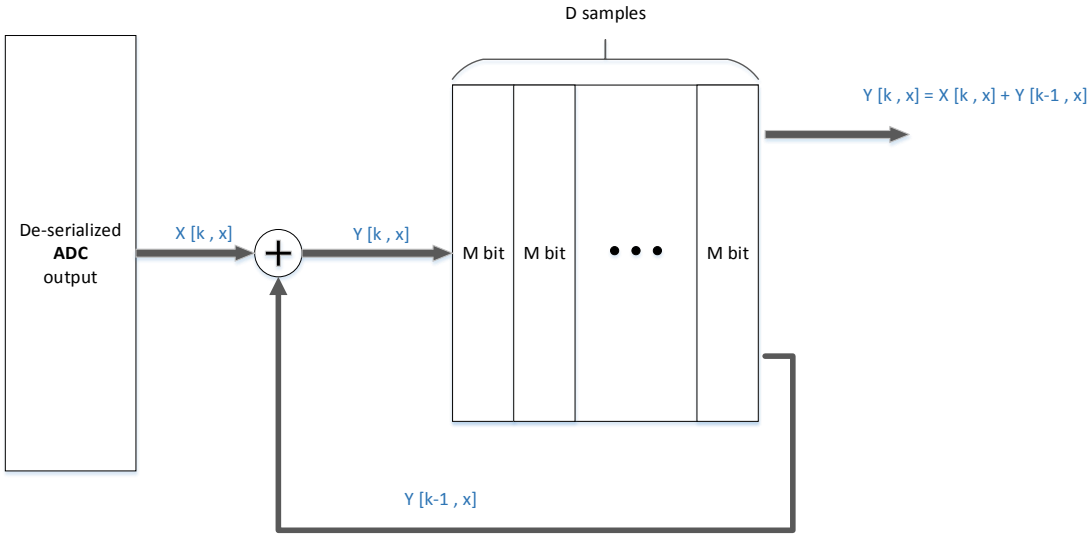

Figure 2.12: On-chip adder for accumulating ADC samples and increase SNR.

light pulse (e.g. 2048) and M is the bit size of each sample after adding it 'k' times, for 'k' light pulses.

A sample acquired by the ADC, represented as X[k,x] – sample 'x' from light pulse 'k', would be added with the corresponding stored sample produced on the previous light pulse – Y[k-1,x].

This would result in the new sample − Y[k,x], which would eventually be stored in the same position in the FIFO buffer where X[k,x] previously was.

Chapter 3

Proposed System Overview

This chapter presents an overview of the proposed system in this thesis, for processing LiDAR data. The development of the system followed four major steps as illustrated in Figure 3.1.

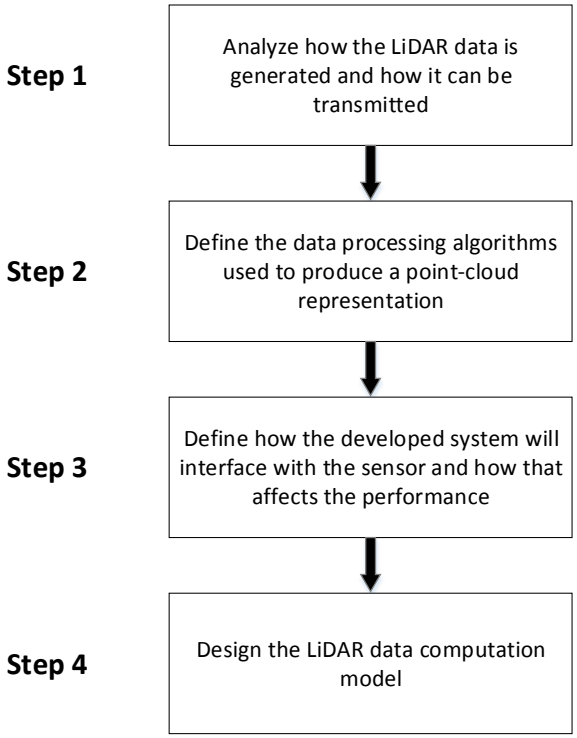

Figure 3.1: Flow diagram followed in the development of the system.

This chapter will focus on the first two steps and chapter 4 will describe the last two steps in a hardware description of the developed system.

Throughout this chapter, a deeper analysis will be presented on how the sensor operates, what parameters are involved in it and what is the dynamic between all the variables in the sensor system. Moreover, solutions to the time restrictions inherent to the LiDAR sensor will be analyzed. Afterwards, the choices made in the data processing algorithms will be duly described. By the end

of this chapter, the proposed system shall be clear and the ground-work for fully understanding it will be completed.

3.1 LiDAR Sensor

This section presents the performed analysis of the LiDAR sensor, which is the cornerstone of the developed data processing system since that is where the data originates. A clear understanding of how the sensor operates is crucial in designing a system that processes data to create a point-cloud representation.

The sensor concept in use for the development of this project is accurately depicted in Figure 3.2. The first consideration in choosing this sensor was that, as mentioned in chapter 2, a 1D MEMS mirror presents advantages in controlling, power consumption and cost when compared to a 2D MEMS mirror and has the same advantages plus durability when compared to scanners with mechanical parts, which break down quicker. Additionally, since this mirror is only allowed to oscillate around one axis, therefore scanning the environment along one dimension of space, a good compromise is reached between how fast the frame is scanned and how much data is generated per light pulse. While a 2D MEMS mirror would allow to focus smaller portions of the frame individually, it would take considerably more time to scan it and a flash laser would illuminate the entire frame at once but would generate considerably more data in a very small portion of time.

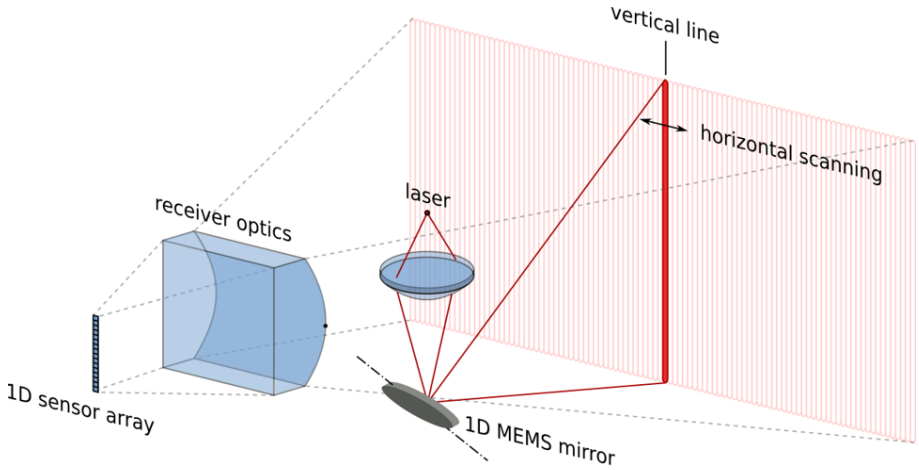

Figure 3.2: LiDAR sensor scanning the environment.[1]

A MEMS mirror is resonantly oscillating at a fixed frequency $-f_{MEMS}-$ and a pulsed laser, aimed at the mirror, is firing light pulses at a certain pulse repetition frequency (PRF). As illustrated in Figure 3.2, a lens is placed in front of the laser, focusing the light towards the mirror and reflecting in a way that covers the entire vertical field of view. A second lens, capable of acquiring incoming light from the entire frame, is positioned so that it redirects the totality of light into the

[1] Courtesy of Innoluce B.V.

sensor array. Since a 1D MEMS mirror is used, the frame is scanned horizontally. A vertical APD sensor array was considered the most adequate to receive the light from the frame because, as show in Figure 3.2, the sensor illuminates a vertical strip of the frame every time a light pulse is fired. Each APD on the sensor array will measure the reflected light from a pixel on that vertical strip.

Furthermore, since the mirror is oscillating at f_{MEMS}, the oscillation period, $T_{MEMS} = \frac{1}{f_{MEMS}}$, is the time it takes the mirror to complete a full oscillation, sweeping through the entire frame twice, offering the possibility to direct light towards the entire horizontal field of view. Since the laser does not move, it utilizes the range of motion of the mirror, correctly timing the laser pulse emissions, to direct light towards different positions of the frame in front of the sensor. In half a mirror oscillation, or one mirror sweep, the laser will fire multiple pulses, which is given by the ratio in (3.1):

$$Light\ pulses\ per\ sweep = \frac{PRF}{2 \times f_{MEMS}} \tag{3.1}$$

3.1.1 Grid Sampling

As already briefly stated in section 1.3, the LiDAR sensor is used to scan a grid of N lines and M columns in its entirety to complete an acquisition of a θ° horizontal field of view. Since the frame is scanned horizontally, every time a light pulse is fired, data from an entire grid column is acquired and each APD on the sensor array will receive light from a pixel in that grid column. The images the system will produce have a horizontal resolution dependent on the number of columns on the grid and the FOV (θ°), given by (3.2).

$$Horizontal\ resolution = \frac{\theta^\circ}{Number\ of\ Columns} \tag{3.2}$$

The process of scanning the entire grid can be described as follows. First, the reflected light from a grid column is acquired. As represented in Figure 3.3, a light pulse is fired aiming at a certain position in the grid. Objects in that position will reflect part of the light back to the sensor. The detector will acquire that reflection and that data will be processed to give information about the objects positioned in that column, relative to the sensor.

Secondly, for an entire mirror sweep, data from multiple columns is acquired in the same way, as shown in Figure 3.4. The described process of sampling data from a grid column is repeated as the MEMS mirror moves and laser pulses are fired directed at different positions of the frame.

Once the mirror completes a sweep (half mirror oscillation), it will come back to its initial position, completing an oscillation (two sweeps). Two options are proposed after the first sweep: acquiring the entire frame first or repeating the columns in the sweep multiple times.

Acquiring the whole frame first implies that the sensor will continue sampling data from other columns of the grid, firing light pulses at different angular positions, until all the columns on the grid are gathered, therefore acquiring the entire frame. This operation mode is good for providing low SNR representations of the grid at a high rate, which is helpful in quickly detecting objects

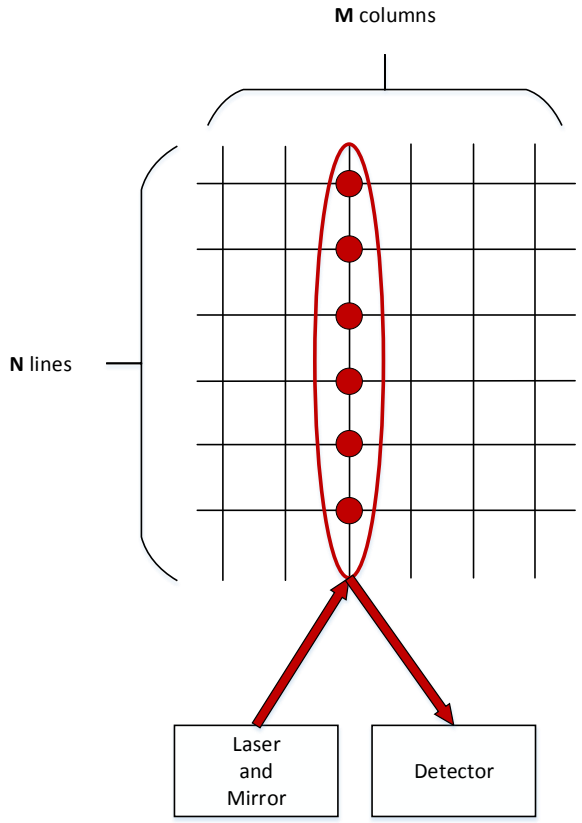

Figure 3.3: Acquiring the reflected light from a grid column.

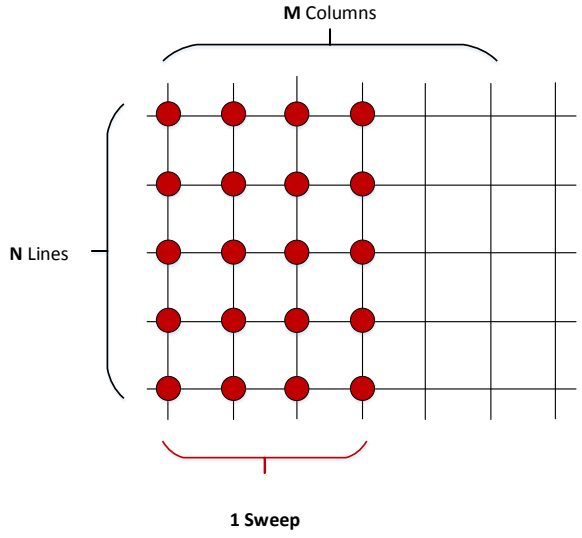

Figure 3.4: Amount of columns sampled in one sweep of the MEMS mirror.

when the vehicle is moving at a high velocity. However, it poses some constraints regarding memory due to the amount of data in a frame, which will be analyzed further.

In the second option, the same columns gathered in the first sweep are sampled an arbitrary

amount of times more. As the mirror comes back to its initial position, the same number of light pulses are fired and the same columns are sampled, essentially repeating the process as the mirror completes the other sweeps. Sampling the same columns multiple times is performed for increasing the SNR of the data collected on those sweeps. It has advantages that will be further explained on section 3.4. This mode is only capable of producing high SNR frames at a lower rate but is more appealing regarding memory aspects.

3.1.2 Sensor Parameters

In this section, an analysis is performed of the parameters involved in the sensor so that the groundwork for discussing how to interface the sensor with the proposed system is set. Many parameters are involved in how the sensor works, which can influence each other and consequently have implications on the performance of the data processing system.

The number of APDs on the sensor array equals the number of lines on the grid, which in turn indicates the number of ADCs that will be generating data every time a light pulse is fired. After that light pulse emission, the ADCs will sample the incoming analog signal for a certain amount of time $-sampling\ duration$. During that time, each ADC will output samples at a predetermined sampling frequency $-f_{sampling}-$ each with value proportional to the amount of light detected. The number of columns on the grid also affects the horizontal resolution, as stated in (3.2).

The PRF directly translates to the rate at which columns from the grid are being acquired. For this reason, independently of the order that columns are sampled in, the greater the PRF value, the faster the rate at which frames are acquired. Reading columns at a faster rate also implies that the same amount of data is generated in shorter periods of time, which brings difficult interfacing challenges that will be explored in section 3.1.3 in greater depth.

On the ADC side, some very important variable parameters are involved as well. As a practical example, consider an ADC working at a sampling frequency of 1 GHz for a total sampling duration of $2\,\mu$s. Light speed is considered $c = 3 \times 10^8$ m/s for simplification purposes. The ADC outputs a sample every $\frac{1}{f_{sampling}} = 1$ ns and will do so for a sampling duration of $2\,\mu$s, producing a total of 2000 samples. When the first sample is acquired, it is known that the emitted light pulse only had 1 ns to travel, covering a total distance of $3 \cdot 10^8 \times 1 \cdot 10^{-9} = 0.3$ m. Since any detected light had to travel the distance from the object to the sensor twice (to hit the object and reflect back), it would only be possible to detect light reflected from objects distanced at most 0.15 m from the sensor in the first 1 ns. For this reason, samples are often represented in a distance axis instead of time axis, where this first sample for example would occur at 0.15 m instead of 1 ns. This consideration is taken for every sample. The value of each sample can only correspond to an object distanced from the sensor a maximum distance allowed by how much time has passed since the light pulse emission. Thus, the sampling duration has implications on the maximum distance light can reach and travel back to the sensor. If the last sample is taken $2\,\mu$s after the light pulse was fired, it means that it could travel $3 \cdot 10^8 \times 2 \cdot 10^{-6} = 600$ m and the sensor would detect objects distanced

at most 300 m. This value is called depth range and is therefore given by (3.3).

$$Depth\ range = \frac{c \times sampling\ duration}{2} \tag{3.3}$$

This way, the time between two sample acquisitions translates to an extra distance that light could cover in that time. That distance, called depth resolution, depends solely on the sampling frequency of the ADC, as indicated in (3.4).

$$Depth\ resolution = \frac{c}{2 \times f_{sampling}} \tag{3.4}$$

From a data processing perspective, one of the most crucial outcomes of changing the parameters is the amount of data produced for every light pulse. Knowing that whenever a column is sampled, all ADCs produce samples at a rate given by the sampling frequency during the entire sampling duration, the amount of data generated by an ADC (in bits) can be obtained by:

$$Data\ per\ ADC = f_{sampling} \times sampling\ duration \times bits\ per\ sample \tag{3.5}$$

3.1.3 Time Restrictions

Having now better understood what parameters are involved in how the sensor operates, this section will present the executed analysis on how changing them affects the amount of data generated and consequently the time restrictions involved in interfacing with the sensor. Some solutions to those restrictions are proposed and analyzed as well.

When a light pulse is fired and a grid column is sampled, a total amount of data is generated during the sampling duration time. That total amount depends on the number of ADCs being used and it can be simply deduced using (3.5), resulting in (3.6).

$$Data\ per\ column = Data\ per\ ADC \times Nr.\ of\ ADCs \tag{3.6}$$

Considering an example where the parameters are set as follows:

- PRF = 80 kHz meaning that light pulses are fired with a periodicity of 12.5 μs
- 32 ADCs
- $f_{sampling} = 1$ GHz
- sampling duration = 2 μs
- 8-bit samples

equations (3.5) and (3.6) indicate that every 12.5 μs a total data amount of 512 kbit is generated. This data must be acquired and sent to the microcontroller in the appropriate time frame. Considering that, as exemplified in Figure 3.5, a buffer is used to store all the samples generated for 2 μs and then uses the remaining time before the next light pulse to empty out, the samples would have

to be read out at a rate of $\frac{512000}{10.5\times10^{-6}} = 48.76\,\text{Gbps}$ to avoid overflowing, which is an impractical value for this application.

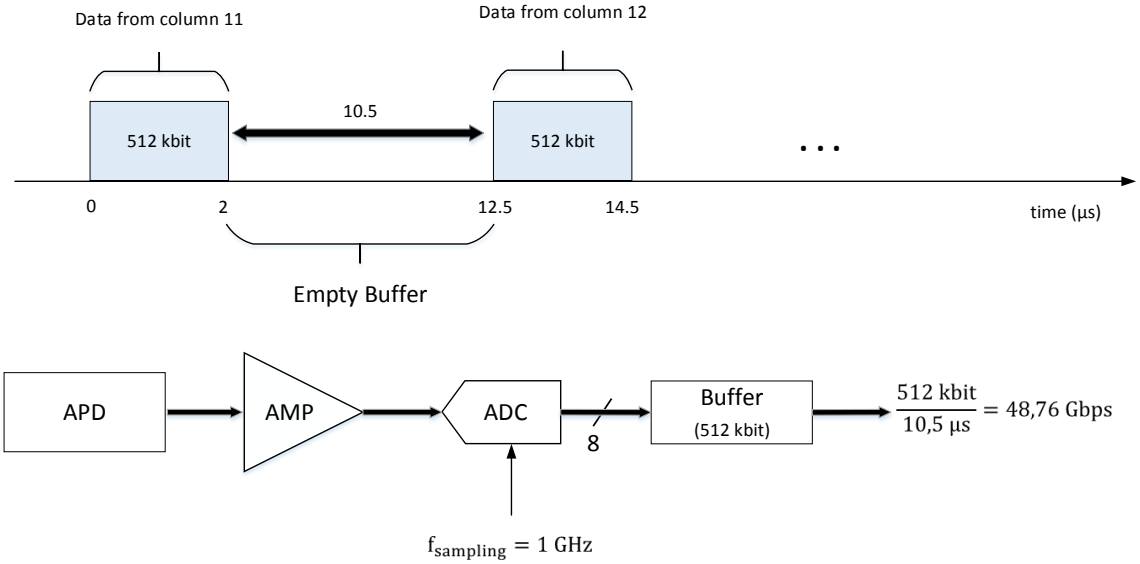

Figure 3.5: Time diagram for 80 kHz PRF and one buffer for storing data from one grid column.

However, more buffers could be included that hold more than the data size of one column, so that while one buffer is being filled, the other one is being emptied out and vice-versa. Figure 3.6 illustrates such a solution and the corresponding time diagram.

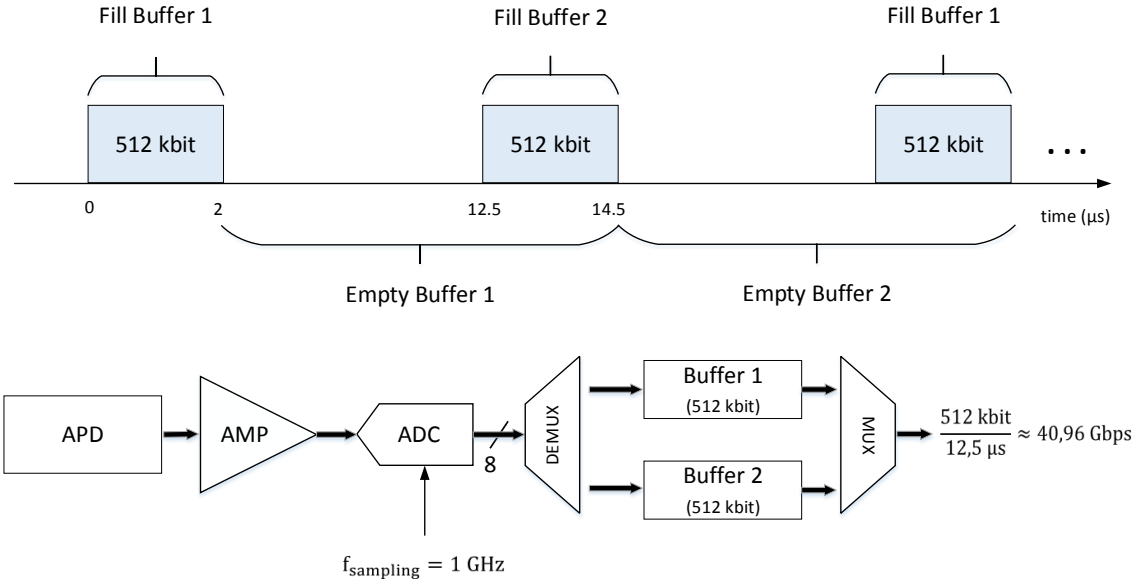

Figure 3.6: Using two buffers for acquiring data with 80 kHz PRF.

This time, each buffer can take a total of 12.5 μs to empty because the other buffer will be taking in the data from the next column. Still, with this architecture the required data rate is

$\frac{512000}{12.5\times10^{-6}} = 40.96\,\text{Gbps}$, which is still a considerably high value.

Since the PRF dictates how fast the columns from the grid are sampled and consequently the output rate in frames per second (FPS), a solution proposed for this problem is to set the PRF in regards to the desired FPS value, instead of the other way around.

Let us consider a minimum value of 25 FPS, which might not be desired for a vehicle traveling at a high speed or in adverse weather conditions but could still be used in more favorable conditions and for that reason constitutes a valid minimum threshold for the FPS output rate. This FPS value indicates that a frame would have to be produced every $\frac{1}{25\,FPS} = 40\,\text{ms}$. In order to produce a final frame to be used by the vehicle, the entire grid data would have to be acquired multiple times because multiple acquisitions are combined to produce a final frame (to be further explained in section 3.4). Adding to the example that the grid has 120 columns and needs to be acquired 8 times before producing a final frame, the number of columns that need to be sampled to produce a frame is $120 \times 8 = 960$, which is also the number of light pulses that need to be fired by the laser. Evenly distributing the required amount of columns to be sampled in the 40 ms time period necessary to produce a final frame results in the required PRF: $\frac{120\times8}{40\times10^{-3}} = 24\,\text{kHz}$. Therefore the PRF can be achieved in function of the desired FPS, using (3.7).

$$PRF = Number\ of\ columns \times Number\ of\ acquisitions \times FPS \tag{3.7}$$

A PRF value of 24 kHz means that a light pulse is emitted every $\frac{1}{24000} \approx 41.67\,\mu\text{s}$. As illustrated in Figure 3.7, by maximizing the time between two light pulses as described, using a system with two buffers would require a read out data rate of $\frac{512000}{41.67\times10^{-6}} \approx 12.29\,\text{Gbps}$ which is a considerably more reasonable value. This indicates that spreading the laser pulse emissions evenly throughout the available time to produce a final frame constitutes a good solution for decreasing the required high interfacing data transfer rates.

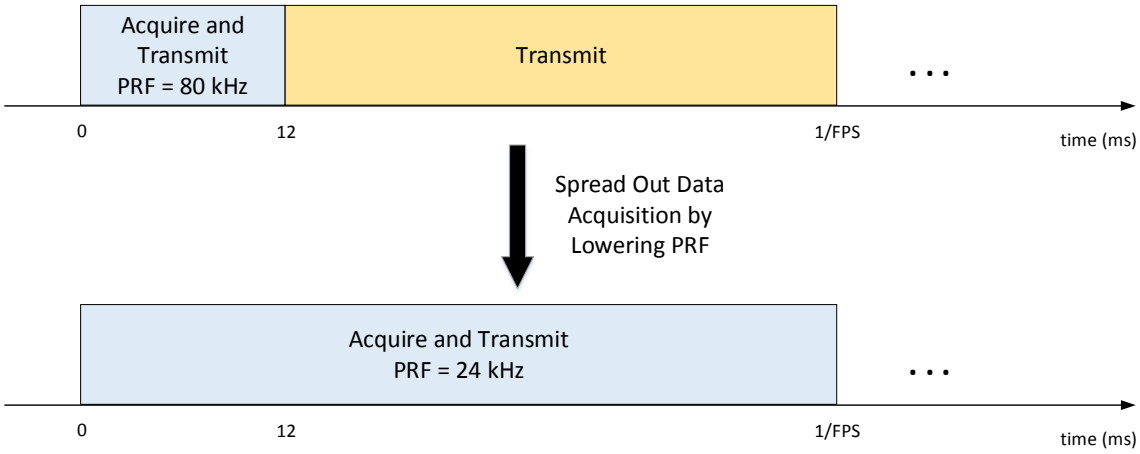

Figure 3.7: Spreading data acquisition through time to achieve the same FPS value.

Supposing now that the sensor buffers can be read out at fixed values of 8, 10, 12, 14 and 16 Gbps, which are legitimate values to consider supposing that multiple LVDS lanes can be used

to transfer data from the sensor to the microcontroller (to be explored in chapter 4), the rate at which the data is sent to the microcontroller is what ultimately dictates the possible FPS output value, assuming the buffers are constantly outputting data at the maximum allowed rate by the connection. Therefore, using 2 buffers being filled one at a time, the highest achievable FPS value is given by (3.8).

$$FPS_{max} = \frac{Read\ Out\ Data\ Rate}{Total\ Data} \tag{3.8}$$

Thus, recurring to (3.7) and (3.8), the values presented in table 3.1 ensue from the parameterized read out data transfer rates.

Data Rate (Gbps)	Maximum FPS	PRF (kHz)
8	16.276	15.625
10	20.345	19.536
12	24.414	23.437
14	28.483	27.434
16	32.552	31.250

Table 3.1: Maximum possible FPS values the sensor can output and corresponding PRF for the parameterized data transfer rates.

Nevertheless, it might be the case that the PRF can not be decreased to such low values as given in this example of 24 kHz because the laser might need to keep up with the mirror oscillation frequency and fire a minimum number of laser pulses in one mirror sweep. If this is the case, still assuming the predefined buffer read out data rates, at a certain PRF value the buffers will not be completely emptied out before the next time they are filled, resulting in overflowing. To counter this phenomenon, a possible solution is to increase the amount of buffer memory so that the excess data generated every light pulse could be stored. After acquiring all the data necessary to build a frame, an extra amount of time would need to be taken to completely empty out that memory, as exemplified in Figure 3.8. After all the data is read from the buffer, the acquisition restarts.

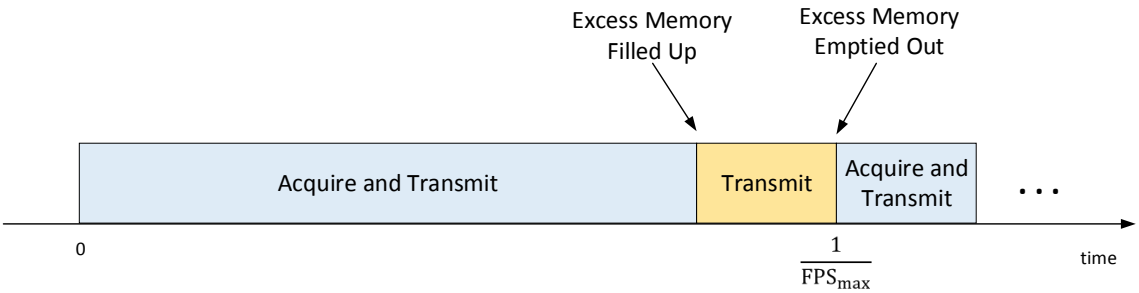

Figure 3.8: Time diagram of excess memory being built up during the acquisition period.

The situation described so far in this section can be simplified to a difference between the rate at which the buffers are being being filled and emptied, in which extra memory is required when

the input data rate is greater than the output data rate. Figure 3.9 illustrates this simplification, considering all memory used by the sensor as a single buffer.

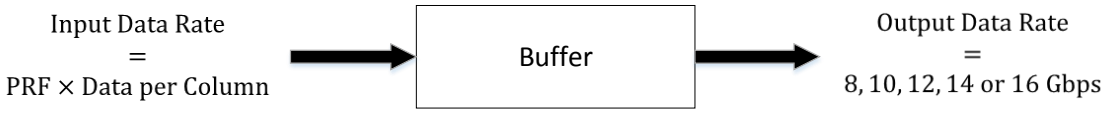

Figure 3.9: Schematic generalizing the rate at which the buffer is filled.

As indicated in Figure 3.9, the rate at which the buffers are being filled up can be calculated by the PRF and the data generated per column, expressed by (3.9).

$$Input\ Data\ Rate = PRF \times Data\ per\ column \tag{3.9}$$

The output data rate is one of the parameterized values of 8, 10, 12, 14 or 16 Gbps. So excess data is generated at a rate simply given by (3.10), if the input data rate is greater than the output one.

$$Excess\ Data\ Rate = Input\ Data\ Rate - Output\ Data\ Rate \tag{3.10}$$

For this reason, whenever excess data is being generated, the amount of memory required to hold that data is given by (3.11).

$$Memory\ required = \frac{Number\ of\ Columns \times Number\ of\ acquisitions}{PRF} \times Excess\ Data\ Rate \tag{3.11}$$

This way, assuming a constant buffer read out data rate, increasing the PRF implicates adding extra memory to store the excess data that could not be emptied out in time. This trade-off between PRF and required memory is illustrated in Figure 3.10.

Notice that for each curve in Figure 3.10, up to a certain PRF value, the required memory is zero because the output data rate is greater or equal than the input data rate. As long as that is the case, increasing the PRF effectively makes it so that the sensor outputs frames at a higher rate. However, once the input data rate becomes higher than the output data rate, the described bottleneck situation occurs and the maximum FPS values indicated in table 3.1 are achieved. This correlation between PRF and FPS is better illustrated in Figure 3.11. The doted lines indicate the value at which the maximum achievable FPS is reached for the different output data rates.

3.2 Histogram Representation

The first proposed approach for processing the LiDAR data included the use of 1-bit comparators instead of ADCs, recurring to histogram representations to aid in detecting light pulse reflections. In this scenario, the current signal from the APD goes through a comparator, with a fixed threshold value indicating the distinction between noise and a valid light reflection, and then the comparator

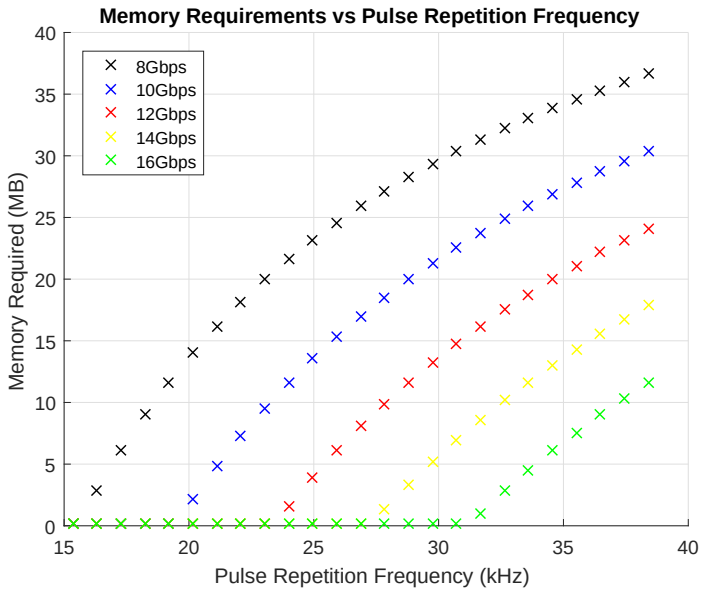

Figure 3.10: Amount of memory required as the PRF increases.

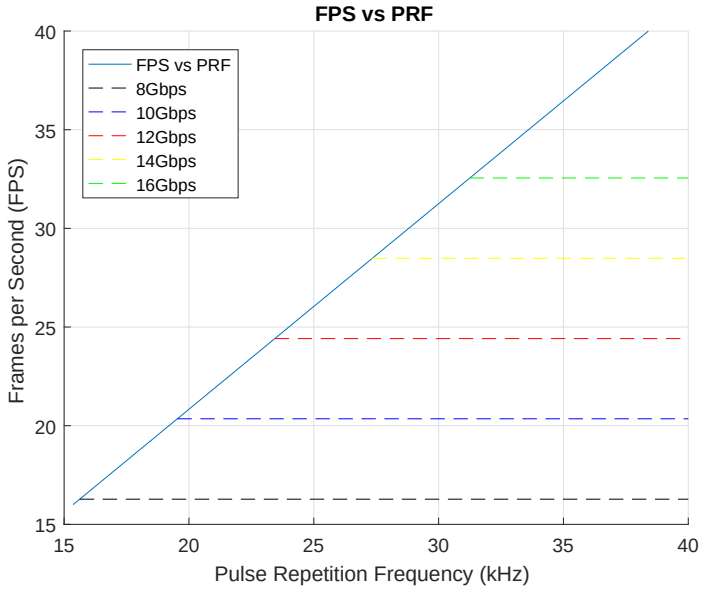

Figure 3.11: Maximum achievable FPS depending on PRF.

would output a 1-bit value corresponding to one of those two possibilities. Then, multiple such acquisitions would take place and a histogram counting the amount of 1's detected in each sample from multiple acquisitions could be formed. Figure 3.12 provides an example of how a histogram of two measurements is accomplished.

As observed in Figure 3.12 a peak in the histogram begins to form, suggesting the detection of a light pulse reflection. More than two measurements could take place and if the detection of a

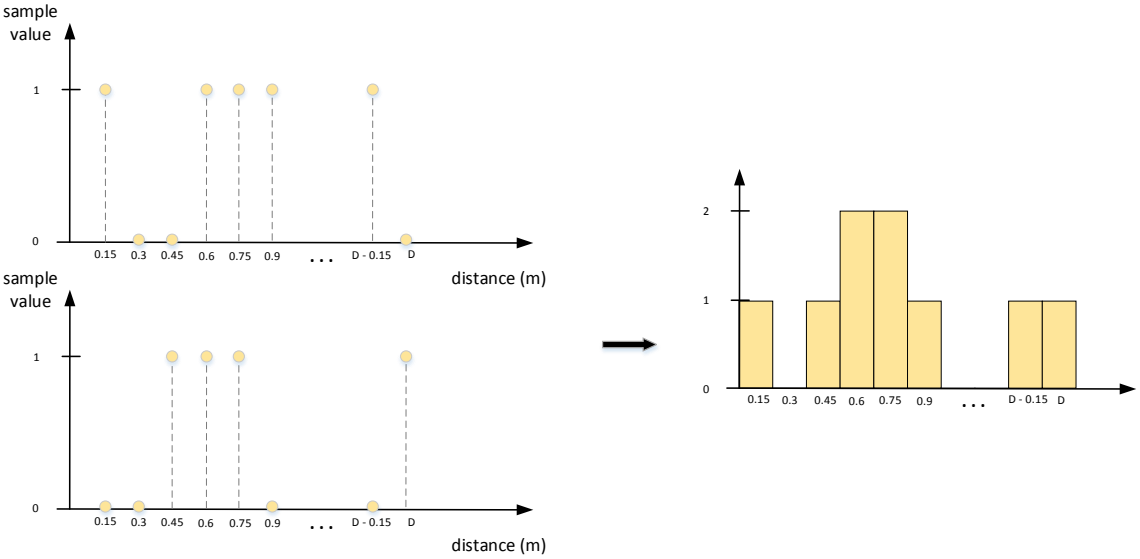

Figure 3.12: Example of a histogram representation of LiDAR data.

logic value 1 occurred consistently around the same region, a peak would form in the histogram and it could be deduced with greater certainty that an object would be present in that pixel and in that distance from the sensor.

However, converting the reflected light into a 1-bit value corresponds to a very low resolution and potentially inaccurate representation of the frame. The more bits are used per sample, the more resolution there is to represent the amount of light detected and lower is the quantization noise of the ADC. The maximum sample value in a N-bit ADC is given by:

$$Maximum\ value = 2^N - 1 \tag{3.12}$$

Now in the scenario that a higher resolution representation is achieved using N-bit ADCs, the histogram technique is replaced with the averaging algorithm. Instead of counting the occurrences of a logic 1 on each sample for all acquisitions, each detected N-bit value on a sample is added with the value obtained for the same sample in different measurements.

In LiDAR data processing, the averaging technique can stop after simply adding the multiple measurements performed for all the pixels or, after adding, the resulting value on each sample can be divided by the number of acquisitions, effectively performing an average. If only addition is chosen, the full resolution of adding N-bit samples multiple times is kept but the memory requirements for storing an entire grid increase substantially. Dividing the result after adding multiple times abdicates that resolution for an easier to store grid data.

3.3 Notation

In order to describe the developed system to increase the SNR of LiDAR data, a certain notation has to be first introduced in which each sample used in the process of building a final frame can be uniquely identified. To do so, the notation developed to refer to a sample must include the following information:

- which ADC produced it, effectively indicating to which grid line it belongs to
- what column it belongs to
- it must be identifiable in the sequence of samples the ADC produces for each light pulse emitted
- to which repetition of the same column does it belong to. In other words, what sweep through the same set of grid columns does the sample belong to.

For this reason, each grid line, and consequently each ADC, is labeled alphabetically as portrayed in Figure 3.13.

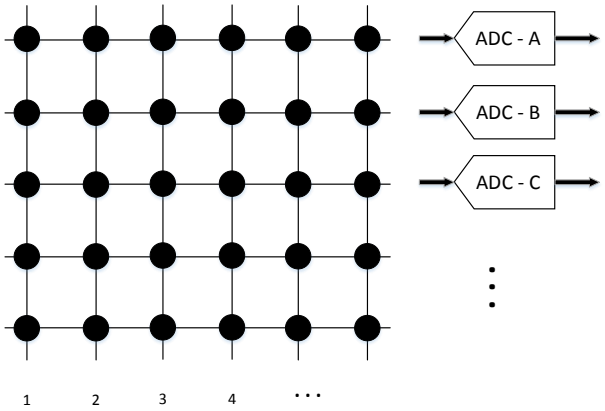

Figure 3.13: Each ADC corresponding to a grid line is labeled alphabetically.

Every sample an ADC outputs will be subjected to the averaging algorithm, which will combine that sample with the ones acquired on the same position but in previous sweeps. So, a distinction between a sample before and after averaging must necessarily be made as well. The notation used to refer to a sample will then be the following:

- $x_k^*[i, j]$ refers to the k^{th} sample produced by ADC X while sampling column i and on the j^{th} sweep of the same set of columns, before averaging.
- $x_k[i, j]$ refers to the k^{th} sample of ADC X from column i and sweep j after being subjected to the averaging algorithm.

For example, $a^*_{20}[5,2]$ is the 20^{th} sample from ADC A for the 5^{th} grid column and 2^{nd} sweep of that column. Whereas $a_{20}[5,2]$ is the sample resulting from combining the 20^{th} sample from ADC A for the 5^{th} grid column and 2^{nd} sweep with the ones acquired on previous sweeps.

3.4 Averaging

This section describes how signal averaging is used to increase the SNR of the signal converted from the acquired light and how it shall work in the context of processing LiDAR data.

Signal averaging is commonly applied in image processing and relies upon the assumptions that the noise signal has zero average and that the light reflection and noise signals are uncorrelated, as is the case with an additive white Gaussian noise (AWGN) model. An advantage of this technique is the fact that it is performed in the time domain. If any kind of filter were to be applied to reduce the noise impact on the acquired signal, like match filtering or a low-pass filter, typically an FFT and inverse FFT (IFFT) would have to be applied to perform the filtering on the frequency domain. Even though an FFT reduces significantly the computation time of the Fourier transform[2], converting a signal to the frequency domain and back to the time domain bears heavily computationally. In this sense, signal averaging is preferred since the processing happens in the time domain, which is how the data must be interpreted to build a point-cloud representation.

Moreover, as it will be clearer by the end of the section, the averaging algorithm has a greater impact on the SNR for small numbers of measurement repetitions across the frame. This aligns well with the time restrictions of the LiDAR system since that in order to maintain a reliable FPS output, the amount of acquisitions of the entire frame should be kept to a minimum.

In averaging LiDAR data, every time the pixels on a column are sampled due to a light pulse emission, a set of samples are produced containing information about the amount of light detected on each of those samples. A reflection from an object in a certain pixel of the grid would stand out as one or more consecutive samples having a higher value than the rest. In other words, it would be visually represented as a peak in the acquired set of samples. However, each pixel measurement contains a certain amount of noise as well – noise from ambient light being detected by the photodiodes, from light being reflected from objects not on the desired grid position, from the physical properties of the photodiodes, etc. Using the fact that this noise can be modeled as AWGN and that a valid reflection from an object manifests as a peak in the sample sets, adding two sets will effectively make it so that noise tends towards zero and the peak stands out, as seen in Figure 3.14.

A sweep through K columns will produce as many sample sets, like the ones in Figure 3.14, as there are pixels on that set. On the next sweep over those K columns, all the new sets will be added to the ones obtained in the previous sweep. This process will repeat until the necessary amount of acquisitions have been taken to finally divide the resulting sets by the total number of acquisitions, resulting on the averaged final acquisition, with increased SNR. For example, for averaging 4 acquisitions, 4 sets of samples from the same pixel but different sweeps would be

[2]FFT reduces the time complexity of the algorithm from $O(N^2)$ to $O(N \cdot log(N))$

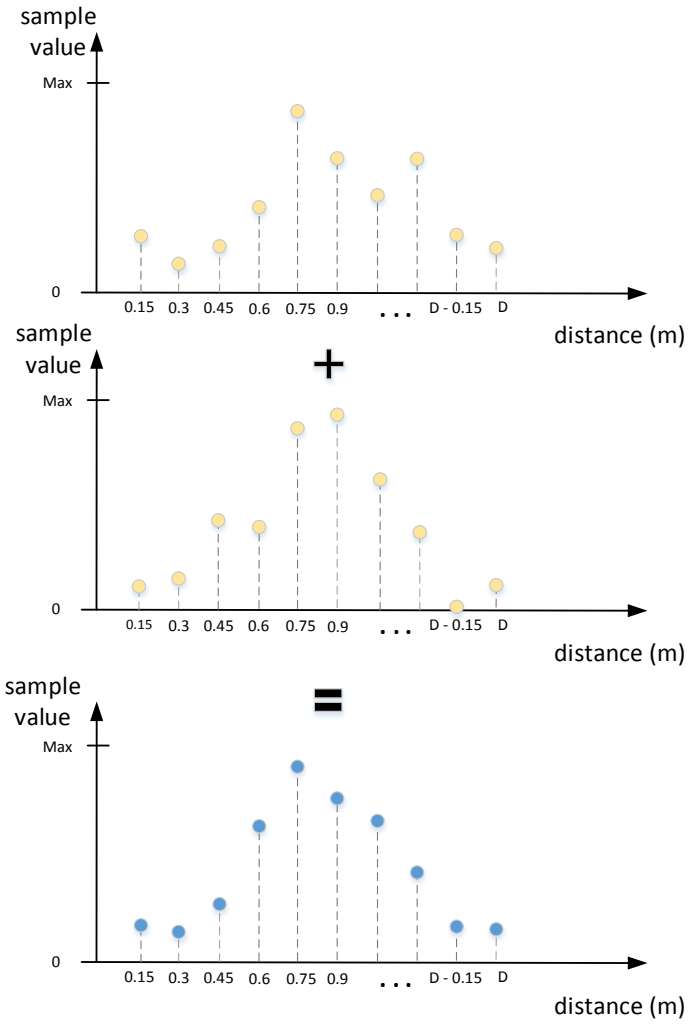

Figure 3.14: Averaging two sets of samples resulting in one with a more noticeable peak.

added and after the final addition all samples would be divided by 4. Note that since the averaging process happens for every sample produced, a sequential example can be made using the already established notation.

Assuming that averaging by 4 is desired, sample $a^*_{20}[5,2]$ is output by ADC A and added with sample $a_{20}[5,1]$ to produce the resulting sample $a_{20}[5,2]$. Afterwards, when sample $a^*_{20}[5,3]$ is produced, it is added with sample $a_{20}[5,2]$, which already has the value of the first and second sweeps, producing sample $a_{20}[5,3]$. Finally, sample $a^*_{20}[5,4]$ is added to $a_{20}[5,3]$ and the resulting sample $a_{20}[5,4]$ is divided by 4, concluding the averaging process for one sample of a pixel.

The process of averaging is a powerful and commonly used technique to increase the SNR of signals affected by noise. In order to numerically analyze how averaging influences the SNR of the signal, let us begin with the following assumptions:

1. y represents the time signal acquired by an APD, containing both noise and the reflection signal.

2. s is the time signal of the light reflection that we wish to detect.
3. r represents the noise signal.
4. r can be modeled as AWGN, meaning that it is a continuous variable following a normal distribution, i.e. $r \sim N(\mu_{noise}, \sigma^2_{noise})$, and different acquisitions of r are uncorrelated.
5. Acquisitions of s and r are uncorrelated.
6. s is present in every acquisition of y in the exact same time slot, meaning that multiple measurements of the same pixel will show the reflected light pulse signal every time in the same time slot. Different acquisitions of s are uncorrelated.

The SNR can be obtained from a ratio between the desired signal average and the standard deviation of the noise signal:

$$SNR = \frac{< s >}{\sigma_{noise}} \tag{3.13}$$

and the measured signal, using assumption 5, can be represented as expressed in (3.14).

$$y = s + r \tag{3.14}$$

Adding m acquisitions of y can therefore be described as in (3.15).

$$\sum_{k=1}^{m} y_k = \sum_{k=1}^{m} s_k + \sum_{k=1}^{m} r_k \tag{3.15}$$

From assumption 6, it can be deduced that the average value of adding multiple acquisitions of s is given by:

$$< \sum_{k=1}^{m} s_k > = < m \cdot s > = m < s > \tag{3.16}$$

and following assumption 4, it is obtained that the sum of multiple r signals follows a normal distribution as expressed in (3.17)

$$\sum_{k=1}^{m} r_k \sim N(m \cdot \mu_{noise}, m \cdot \sigma^2_{noise}) \tag{3.17}$$

where $Var(\sum_{k=1}^{m} r_k) = m \cdot \sigma^2_{noise}$.

Remembering that the standard deviation is given by the square root of the variance, and following (3.13), it is now possible to calculate that the SNR of m additions of y is given by (3.18),

$$SNR_m = \frac{< \sum_{k=1}^{m} s_k >}{\sqrt{Var(\sum_{k=1}^{m} r_k)}} \tag{3.18}$$

which following (3.16) and (3.17) results in (3.19).

$$SNR_m = \frac{m \cdot < s >}{\sqrt{m \cdot \sigma_{noise}^2}} = \frac{m \cdot < s >}{\sqrt{m} \cdot \sigma_{noise}} = \sqrt{m} \cdot \frac{< s >}{\sigma_{noise}} \tag{3.19}$$

The result obtained in (3.19) shows that adding m times a signal acquisition results in increasing the SNR by $\sqrt{m}$.

Figure 3.15 depicts this conclusion, showing what is the improvement on the SNR of a signal after several acquisitions.

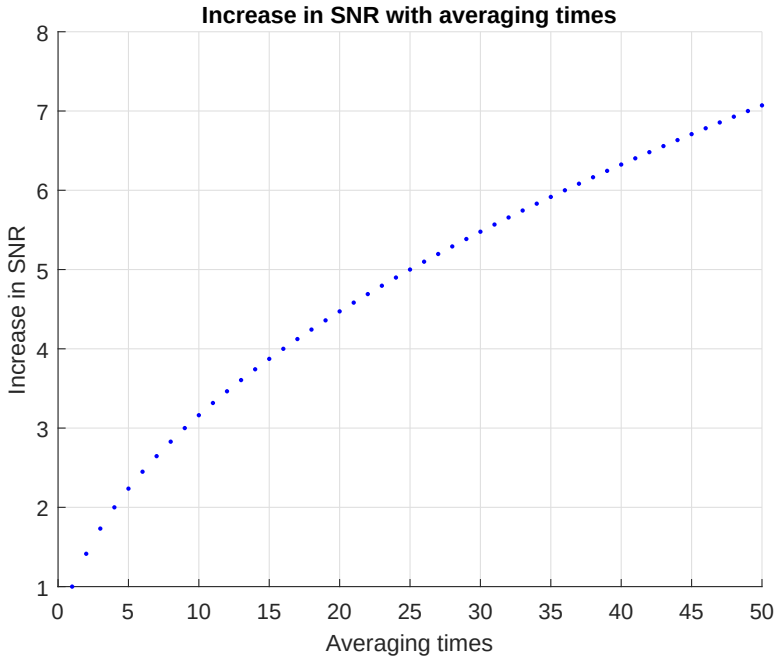

Figure 3.15: Signal SNR increase with amount of averaging performed.

As it can be interpreted from this Figure, as the amount of averaging on a signal increases, the lesser is the improvement on the SNR of that signal. In other words, the slope of the curve decreases as the number of acquisitions increases.

3.5 Thresholding and Peak Detection

After multiple acquisitions have been averaged to provide a high SNR representation of the signal detected on each pixel, the final step is to analyze that signal and determine whether a valid reflection was detected or not. To do so, two steps need to be taken. First, it is necessary to establish what is the value of the threshold that distinguishes between noise and a valid light reflection. Afterwards, the samples must be compared to that threshold to determine whether it corresponds to noise or a light pulse reflection.

3.5.1 CFAR

Several algorithms can be employed to perform the task of thresholding. Some more simplistic algorithms use a constant threshold, which means that when analyzing a signal, a threshold value is calculated once and used for comparison for all the samples on the signal under analysis. The CFAR algorithm is widely used in signal processing for radar applications. It performs adaptive thresholding, learning from the environment to constantly adapt the threshold value in a way that is more tolerant to noise and interference. CFAR applies a sliding window technique over the samples on each pixel to perform the threshold adaptation, using a cell operator on each sample. To do so, each sample is considered a cell, as illustrated in Figure 3.16, with three possible cell categories: training cells, guard cells and cell under test (CUT).

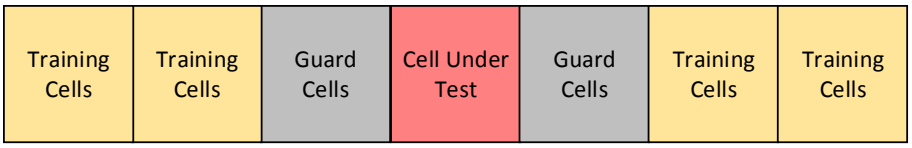

Figure 3.16: Representation of CFAR cells.

The CUT is the one for which the CFAR algorithm will determine the threshold value. For that, it uses the training cells to estimate the intensity value of the background. The guard cells are used as a margin to prevent that the training cells are affected by the presence of an object on the CUT, in which case the intensity of the guard cells might be higher than the training cells because some of the reflected light from an object might have been captured in these samples as well.

To calculate the threshold value many statistical operations can be executed like cell averaging (CA-CFAR) the intensity on the training cells or order statistics algorithms (OS-CFAR) as is the case when calculating the median of the training cells. The detection threshold value is then set above the adaptively estimated value of the background.

In CFAR, peak detection simply consists in using the value of the CUT, comparing it to the corresponding threshold value and outputting a result indicating if the sample is above or below that threshold.

3.6 Concluding Remarks

As analyzed, several steps need to take place in order to produce a point-cloud representation from the information provided by the LiDAR sensor. After all the considerations were taken, an overview of the proposed system was finalized. Figure 3.17 illustrates that system and shows what the performed steps are and where they will take place.

As illustrated, the first step is assembling the data, which translates to interfacing the LiDAR sensor with the microcontroller that will perform the data processing, in an efficient and suitable way for the following signal processing algorithms. The next step is averaging, which will combine multiple acquisitions of the frame to produce a final, high SNR, picture of the environment.

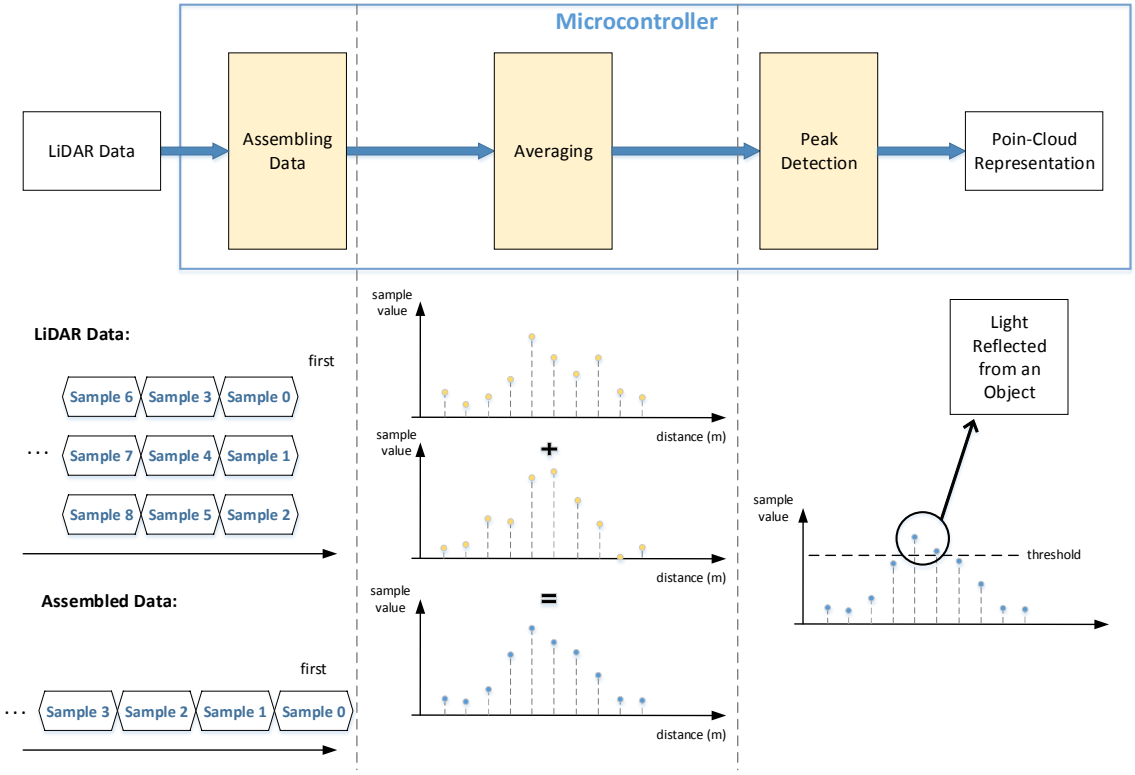

Figure 3.17: System overview – from LiDAR sensor data to a point-cloud representation.

Lastly, the peak detection algorithm uses the data generated from averaging and detects valid reflections from objects in the frame.

Chapter 4

Computation Model

The following chapter will describe in depth the developed computation model that performs signal averaging on the LiDAR data. The most important factor in the design of this system was the balance between the utilized resources and its performance. This limitation in resources was most impactful in the interfacing, memory accessing and storing aspects of the system design, even though it impacts every component of the system.

In Figure 4.1 is presented the full scope block diagram of the conceptualized system. It includes components like the SRIF and RIF, which are interfacing components that will be explored in section 4.2.1, utilized to connect the sensor to the data processing system.

As it is evident from Figure 4.1, the system is comprised of four iterations of the exact same components. The designed system is represented as the "Averager" block, which is illustrated in its entirety in the block diagram in Figure 4.2 and will be described in detail throughout this chapter. Each one of the four averager systems will receive a portion of the grid data and will therefore interface with a portion of the SRAM to store that data. All the subsystems in each averager function so that all samples are ordered, averaged and stored in an efficient and comprehensible manner.

4.1 System Restrictions

After analyzing the LiDAR sensor, the parameters involved, consequent time restrictions and introducing the data processing algorithms, the parameters under which the digital computation model would operate had to be carefully delineated. These restrictions derived from initial impositions that the system had to oblige by, otherwise it would not be implementable in a cost-effective way in the AURIX microcontroller. A maximum of 6 MB of SRAM memory was available for this LiDAR application. The data transfer rate on each LVDS connection was restricted to a maximum of 1 Gbps, even though the system should be designed so that values anywhere from 400 Mbps up to 1 Gbps could still translate into an acceptable performance. The number of RIF components used by the LiDAR system had to be minimized as well. Values from six to eight RIFs should be avoided as much as possible and more than eight would be unfeasible. Finally, the frequency of

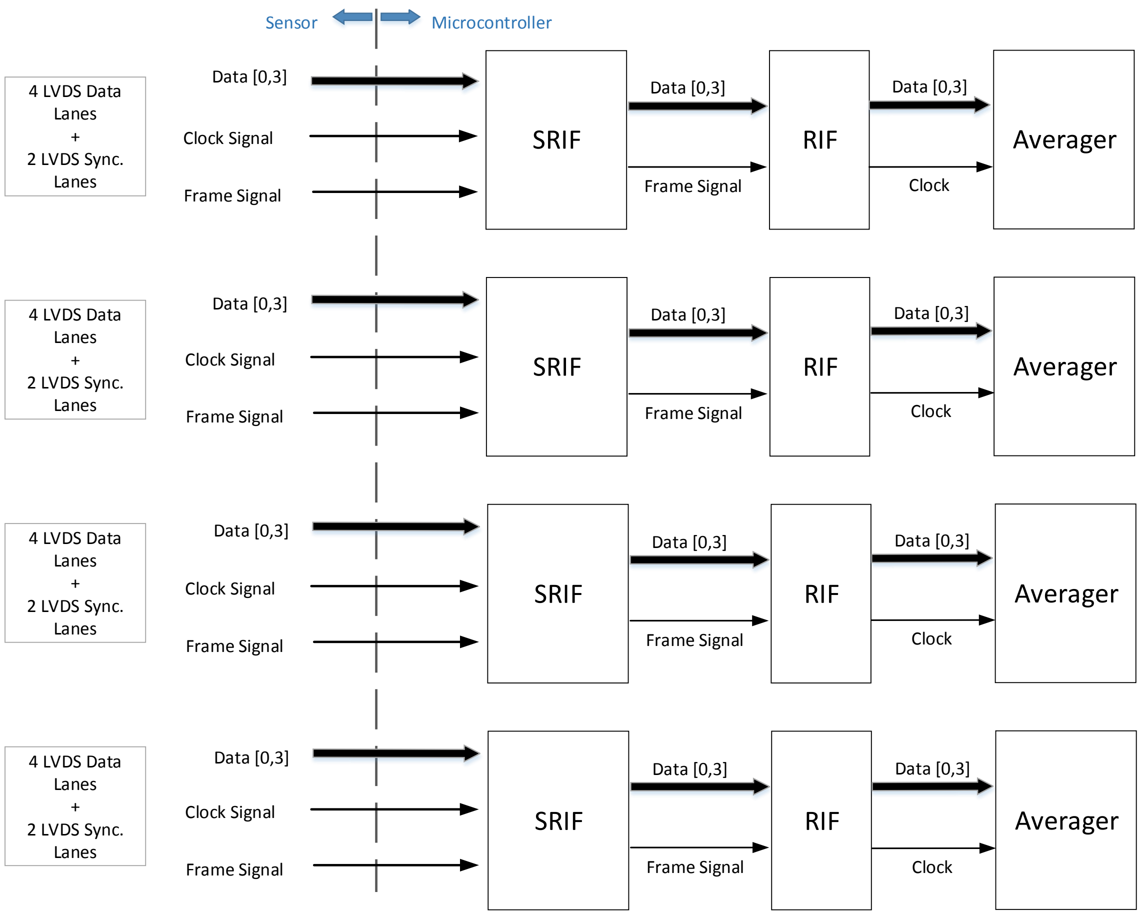

Figure 4.1: Overview of the developed system.

the RIF output clock signal could be considered up to a maximum of 500 MHz, which implies that the averager system is also restricted to that maximum clocking frequency.

The following are the considerations and restrictions, resulting from designing the system accounting for the initial restrictions and performance requirements, that were established so that the system would function correctly.

1. Each ADC sample must be sent uninterruptedly and serially through an LVDS lane, i.e. the bits in a sample can not be split for transfer.

2. There can be no pause between the transmission of any two consecutive samples in the same data lane.

3. Each ADC can generate 8, 10 or 12-bit samples.

4. Each pixel can be comprised of up to 3200 samples.

5. The sensor must split the number of ADCs equally between the LVDS lanes available.

6. The data transfer rate on each LVDS connection can not be higher than 1 Gbps.

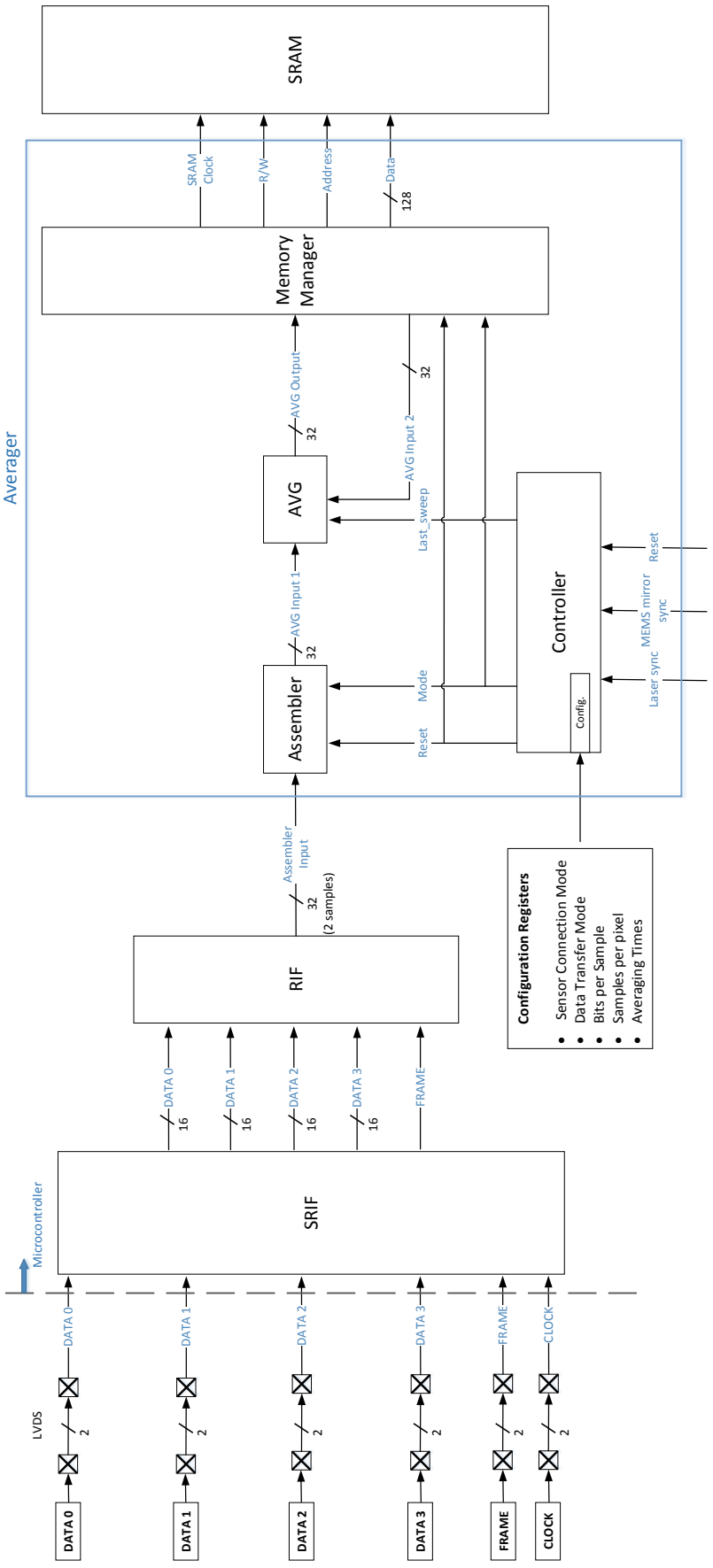

Figure 4.2: Averager block diagram with omitted clock signal.

7. The system has 4 RIF components, each one having 4 LVDS data lanes for connecting with the sensor, meaning that 16 data lanes are available.
8. Each RIF outputs 32-bit samples at a maximum rate of 333.33, 400 or 500 million samples per second (SPS), depending on the bit size of the ADC samples.
9. The system operates at a maximum clock frequency of 166.67, 200 or 250 MHz also depending on the number of bits per sample.
10. The system supports grids with 120 columns and 16 or 32 lines.
11. Averaging by 2, 4 or 8 is supported.
12. The system performs averaging on 20 consecutive grid columns at a time.
13. Each averager subsystem connects to four SRAM banks.
14. The system uses a total of 2 or 4 MB of SRAM for averaging and storing the LiDAR data, depending on the sensor connection mode being used (to be detailed further).
15. When using the sensor connection mode 1, each SRAM bank is accessed at a frequency of 83.33, 100 or 125 MHz, depending on the clocking frequency.
16. When using sensor connection mode 2, each SRAM bank is accessed at a frequency of 41.67, 50 or 62.5 MHz, depending on the clocking frequency.

4.2 Interfacing

This section describes how the system interfaces with the sensor and shows the analysis done in how the interfacing aspects affect the performance.

One of the more relevant aspects in developing the proposed data processing system for LiDAR was interfacing it with the sensor. Not only does the sensor generate large amounts of data, as explored in chapter 3, but it must also connect with the processing system in a relatively transparent way. The data rate input to the microcontroller limits the maximum FPS output it can achieve, for which reason interfacing is crucial for the entire system and must be fully analyzed before design.

4.2.1 RIF

RIF is a hardware component built in the AURIX microcontroller, designed for interconnecting with radar antennas. However, it offers the possibility of connecting to external ADCs, which makes it a viable option for interfacing the LiDAR sensor with the microcontroller. As portrayed in Figure 4.3, the connection to RIF starts with 6 LVDS lanes: 4 data connections, 1 frame signal and 1 clock signal, connecting to SRIF. Each LVDS lane is made up of 2 wires, since an LVDS connection transmits information as a differential voltage value between the 2 wires.

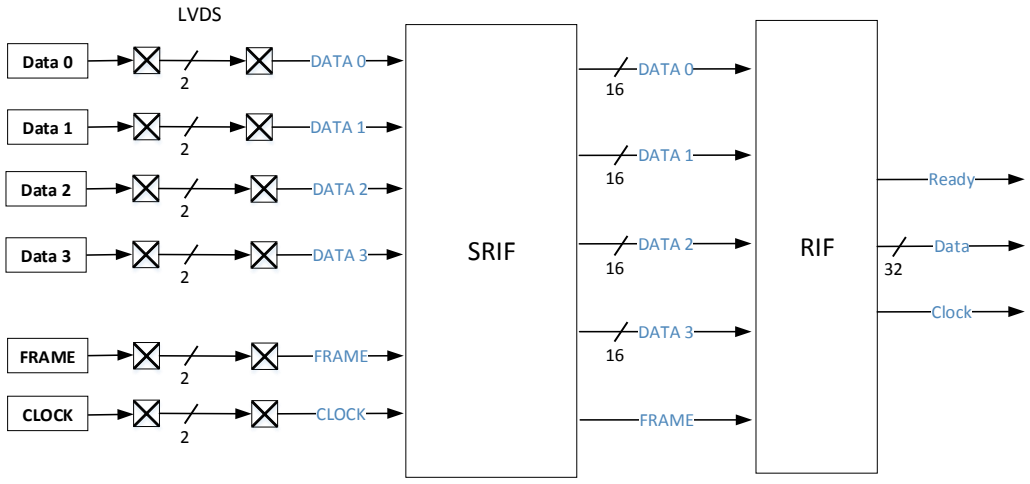

Figure 4.3: SRIF and RIF interfacing components.

Each ADC provides a continuous stream of samples, without pauses between them, through a data lane. The SRIF, which is the component that receives data at a high rate, works in double data rate (DDR) sampling, meaning that at each rising or falling clock transition it will acquire the bit value on the LVDS data lanes. The clock signal must also be shifted one quarter clock period compared to the data lanes. This way, each clock transition shall happen in between bit data time. The frame signal indicates, with a rising edge (low to high differential transition), when a sample is finished and the next one is starting.

Even though RIF supports 10, 12, 14 and 16-bit ADC samples, 8-bit samples are considered as well for being useful in LiDAR applications and providing a good trade-off between the data size of the grid and resolution on each sample, making it a possibility to include the 8-bit sample functionality on RIF in future versions of the microcontroller.

There are four 16-bit wide parallel data connections between SRIF and RIF and one frame signal. Whatever the size of the ADC samples, they will invariably be delivered as 16-bit samples to RIF, being that SRIF performs zero-padding on the msb side of the sample when necessary. Therefore, SRIF receives data from the ADCs in a serial connection, on each LVDS data lane, and outputs them as 16-bit samples. On each rising edge of the frame signal between SRIF and RIF, the four 16-bit samples are written to RIF.

RIF offers many functionalities to be configured by the user. A cyclic redundancy check (CRC) safety mechanism on the communication is available. A data formating unit (DFU) is included in order to allow samples to be sent both msb or lsb first by the ADC, in many formats like signed or unsigned integers and in the bit widths already mentioned. For this LiDAR application, all samples will be read as unsigned integers. Moreover, the FIFO and lane management (FLM) unit provides the option of using the 4 LVDS data lanes to connect to ADCs in different manners. FLM can be configured to read the values from the ADCs as complex numbers, which shall not be used in this application, and also as real numbers. Additionally, the single, double and quad lane modes are available for choosing, along with the number of active lanes. The number of active lanes can

be chosen to indicate how many of the 4 input LVDS data lanes are being used. The lane modes exist in order to choose how RIF picks samples from each data input connection and in what order it outputs them.

Finally, as illustrated in Figure 4.3, the output consists on a 32-bit parallel data bus, a ready signal and a clock signal. The 32-bit output consists on two 16-bit samples coupled together. The ready signal is used to shift out the 32 bits whenever they are ready and the clock signal is synchronized with the ready and data signals and is provided to the following systems. The clock frequency must always be equal or greater than the ready signal frequency, otherwise the following systems would potentially miss rising edges of the ready signal and consequently miss the output samples.

4.2.2 Performance

This section will take in consideration the system restrictions presented in section 4.1 and the way the RIF component operates to describe how the performance of the proposed system changes.

In the current higher end versions of the AURIX microcontroller, two RIFs are available for use. Each LVDS connection works at a maximum rate of 400 Mbps $-LVDS_{speed}$. However, in future generations of the microcontroller, the number of RIFs might very likely increase, especially considering that a LiDAR application could require the use of these kind of interfaces. The maximum allowed data rate on a single LVDS connection is already higher than the mentioned 400 Mbps and is expected to increase in the following years as well. For this reason, different values for the number of RIFs used and $LVDS_{speed}$ were considered when analyzing the performance of the system, along with other variables linked to the sensor that also impact the performance, namely: the number of lines, columns and samples in the grid, bits per sample and also the amount of averaging required.

For this analysis, a higher level of abstraction when looking at RIF becomes useful. Disregarding some of its functionalities like the CRC mechanism, data formatting aspects and data types, both the SRIF and RIF can be interpreted as a black box with an input from the sensor and an output to the microcontroller. The input data rate in bits per second depends solely on the LVDS connection speed and the number of LVDS lanes on RIF.

$$Input = 4 \times LVDS_{speed} \tag{4.1}$$

Nonetheless, given that RIF performs zero-padding on samples, when necessary, to output them as 16-bit samples, effectively adding bits to each sample, the output data rate in bits per second is always equal or greater than the input data rate. However, the input and output data rates in SPS are always the same, with a slight delay induced by the previously mentioned safety and data formatting mechanisms embedded in RIF. Thus, the RIF output data rate, in samples per second, is given by (4.2).

$$RIF_{output} = 4 \times \frac{LVDS_{speed}}{Input\ bits\ per\ sample} \tag{4.2}$$

As mentioned in section 4.2.1, the clock frequency provided by RIF to the following systems is always greater than the frame signal frequency. At the extreme, at every clock rising edge there would be a frame signal rising edge, outputting 2 samples. Therefore, the minimum required clock frequency can be obtained from the RIF output rate, as expressed in (4.3).

$$Min.\ f_{clk} = \frac{1}{2} \times RIF_{output} \tag{4.3}$$

From (4.1), (4.2) and (4.3) two arbitrary values for $LVDS_{speed}$ and bits per sample at the input can be used to obtain the RIF output rate and the minimum required clock frequency of the system.

Another important consideration in interfacing the sensor with the microcontroller is the number of interfaces used. Multiple RIFs can be used for this LiDAR application, each connecting to multiple ADCs. The more interfaces are used the better the performance of the system is in terms of FPS output, allowed bits per sample at the input, size of the grid and amount of averaging. Yet, this number must be limited because for each interface the number of LVDS lanes increases by four, more power is consumed by the system, the bigger the area of the proposed LiDAR data processing system will be and consequently the overall cost of the microcontroller.

Considering that the amount of samples in a grid is function of the number of lines, columns and samples per pixel, as given by (4.4),

$$Grid\ size = Nr.\ lines \times Nr.\ columns \times samples\ per\ pixel \tag{4.4}$$

a dependency can be established between the amount of RIFs required to achieve a certain FPS at the output and the grid size, RIF output rate and averaging times intended. Equation (4.5) expresses that dependency. Note that the number of RIFs must always be a positive integer.

$$Nr.\ RIFs = \frac{Averaging\ times \times FPS \times Grid\ size}{RIF_{output}} \tag{4.5}$$

The values in consideration for the design of the system are as follows:

- LVDS lane speed: 600, 700, 800, 900 Mbps and 1 Gbps
- averaging times: 2, 4 and 8
- input bits per sample: 8, 10, 12, 14 and 16
- grid size: 16 or 32 lines, 120 columns and up to 3200 samples per pixel.

This way, fixing the grid size to 32 lines $\times$ 120 columns $\times$ 2000 samples per pixel, corresponding to 7.68×10^6 samples in the grid, and fixing the RIF output data rate as well, which is given by (4.2), the required number of RIFs to achieve different values of FPS at the output of the system with 2, 4 or 8 times averaging can be plotted as illustrated in Figure 4.4.

From the plot in Figure 4.4, one can understand that the required averaging times impacts greatly the correlation between the number of RIFs required and achievable FPS. As the averaging

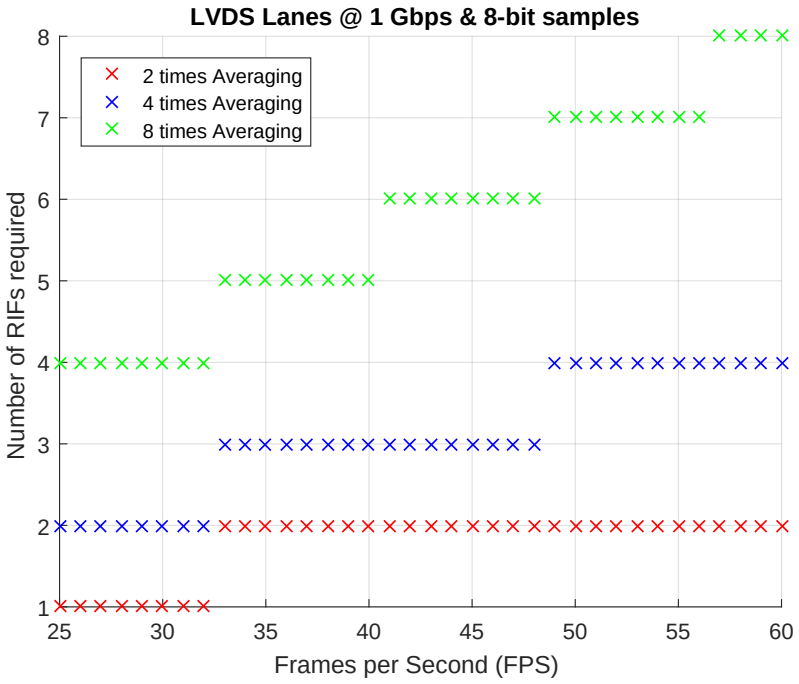

Figure 4.4: Number of RIFs required to reach a corresponding FPS value when operating with 1 Gbps transfer rate at each LVDS lane and 8-bit samples.

times increases, the number of required RIFs rapidly increases, covering a smaller range of FPS values.

So far, the grid size has been fixed to 7.68×10^6 samples, but allowing the grid dimensions to change either in number of lines, columns or samples per pixel, implicates that the achieved FPS at the output changes as well. Parameterizing the grid dimensions as:

- 16 or 32 grid lines
- 120 columns
- 1500, 2000 or 2500 samples per pixel

and using all possible combinations with (4.4) results in grid dimensions of: 2.88, 3.84, 4.80, 5.76, 7.68 and 9.60 million samples. Once established the maximum $LVDS_{speed}$ implementable for the system, the number of interfaces used and the bits per sample on the ADCs, a representation like the one in Figure 4.5 can be achieved.

The graph in Figure 4.5 illustrates the performance of a system operating with $LVDS_{speed}$ at 1 Gbps, 4 RIFs and with ADCs generating 8-bit samples. A range of FPS values from 0 to 150 is considered with regard to each of the parameterized grid sizes. This way, looking at each bar on the graph it is intuitive to understand the best achievable FPS for that grid size performing either 8 (dark blue), 4 (green) or 2 (yellow) times averaging. As expected, for larger grid dimensions, the highest FPS values are only achieved when averaging by 4 or even 2. This representation is a useful tool to simulate the performance of a system when still in the designing stage.

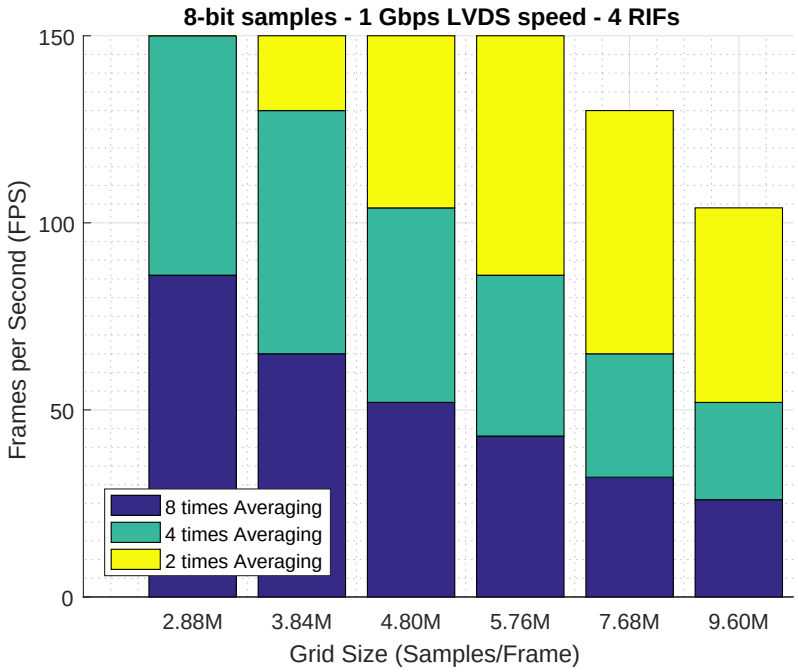

Figure 4.5: Achievable FPS values for different grid sizes when operating with 8-bit samples, 1 Gbps at each LVDS lane and a maximum of 4 RIFs.

4.2.3 Sensor Connection Modes

This section details the available ways of connecting the sensor to the developed system.

Given the performance analysis of the system, it was decided that it would use 4 RIF components, as mentioned in section 4.1. This number is still well within the expected for a LiDAR system capable of performing averaging by 8 on the AURIX microcontroller. Plus, as illustrated in Figure 4.5, for the biggest considered grid size, with 8-bit samples, 26 FPS are still achievable.

This way, the connection between the sensor and each RIF must be well defined. Given that a minimum of 16 lines per grid, using 16-APD sensor arrays, was established as the minimum value that gives a valid vertical resolution on the grid, the system was developed to support grids with 16 and 32 lines. Since RIF has 4 available data connections, this implies that each LVDS data lane will either have to connect to 1 or 2 ADCs simultaneously. That connection had to then be properly named and documented as follows.

4.2.3.1 Sensor Connection Mode 1

The first sensor connection mode applies to the use of a grid with 16 lines. This mode is simpler on the sensor regarding interfacing aspects given that each ADC must connect to an LVDS data lane as illustrated in Figure 4.6.

The clock and frame signal must also connect to the respective lanes and must generate the exact same waveforms for all RIFs so that all samples from the same depth on different pixel positions are acquired simultaneously. Furthermore, as shown in Figure 4.6, the first data lane on

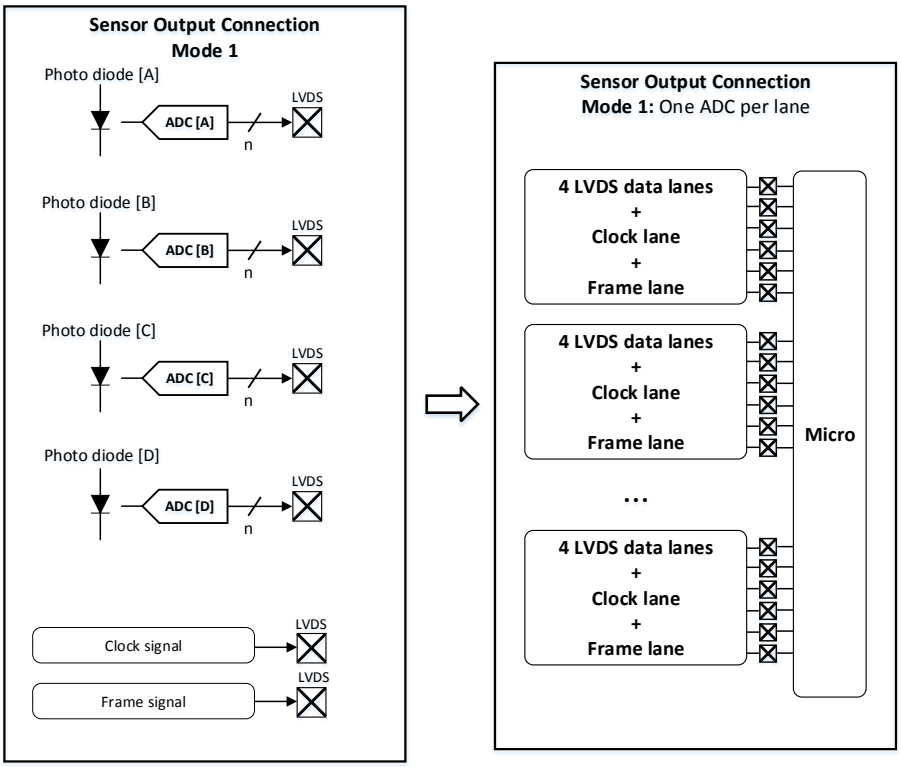

Figure 4.6: Sensor connection mode 1.

the first RIF must connect to ADC A, the second one to ADC B and so on. The connection to the second interface should include ADCs E to H and analogously for the following connections. No data connection lane can be left unused otherwise that would mean that a grid without 16 or 32 lines would be in use, which is not supported by the system.

4.2.3.2 Sensor Connection Mode 2

Sensor connection mode 2 is the one in use when a 32 line grid is wished to connect to the system. As observed in Figure 4.7, a multiplexer must be utilized on the sensor side in order to couple two ADCs on the same LVDS data lane and also correctly synchronize the data incoming from both ADCs.

Apart from multiplexing two ADCs in one lane, which entails some challenges in synchronizing the selection on the multiplexer, sampling on the ADCs and some timing and memory restrictions as already studied in chapter 3, this connection mode is identical to the sensor connection mode 1.

4.2.4 Data Transfer Modes

Following the stipulation on how to connect the sensor to the system, many ways of ordering and transmitting samples are still possible and the system can not support them all. For this reason,

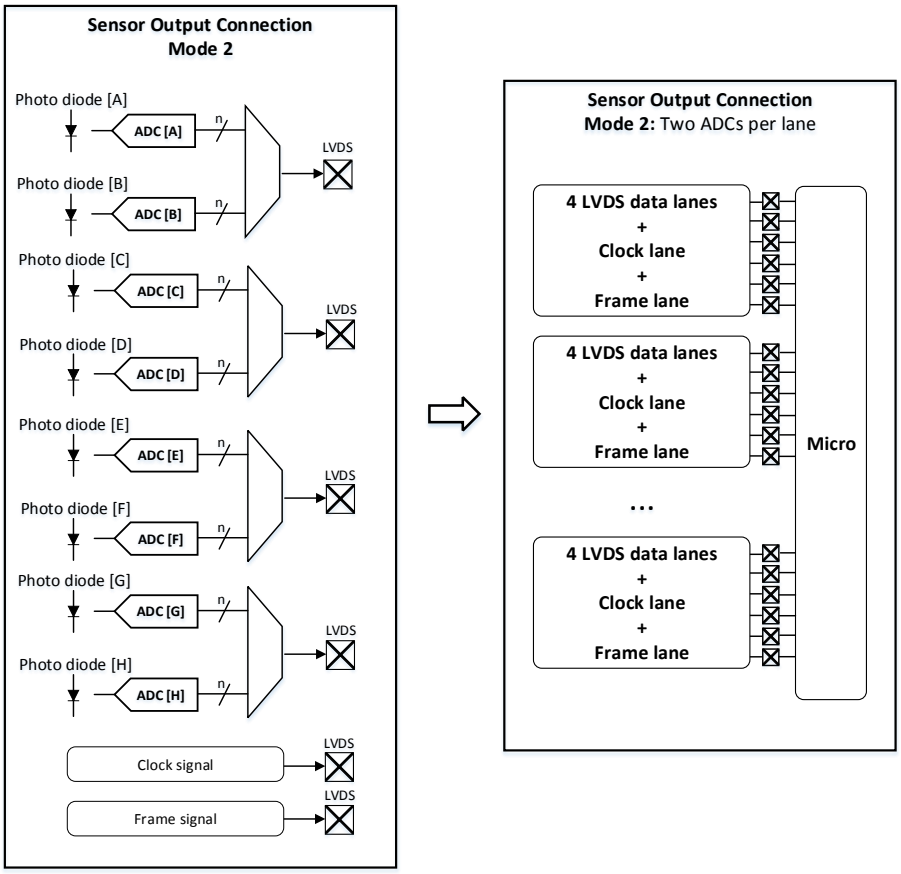

Figure 4.7: Sensor connection mode 2.

depending on the sensor connection mode, the ordering of the samples to be transmitted was defined also in a way that took into consideration how the LiDAR sensor generates samples and what restrictions it faces but also the most convenient way for the system to process and store the grid samples. This resulted in three topologies, designated as data transfer modes, for ordering and transmitting the data to each RIF, which must be replicated to all other three RIF components and respective ADCs. Data transfer mode 1 is the only applicable mode when sensor connection mode 1 is being used and data transfer modes 2.1 and 2.2 are the two available options when using the sensor connection mode 2.

4.2.4.1 Data Transfer Mode 1

When in data transfer mode 1, one ADC will be connected per data lane. Assuming, as an example, that each ADC generates 2000 samples for each column acquisition, Figure 4.8 exemplifies how each ADC connected to the same RIF must send the samples.

All samples shown in Figure 4.8 correspond to the same column and sweep. They must be transmitted from first to last one being sampled, with each ADC utilizing only one LVDS data lane, synchronized with all other samples so that at the rising edge of the Frame signal the SRIF

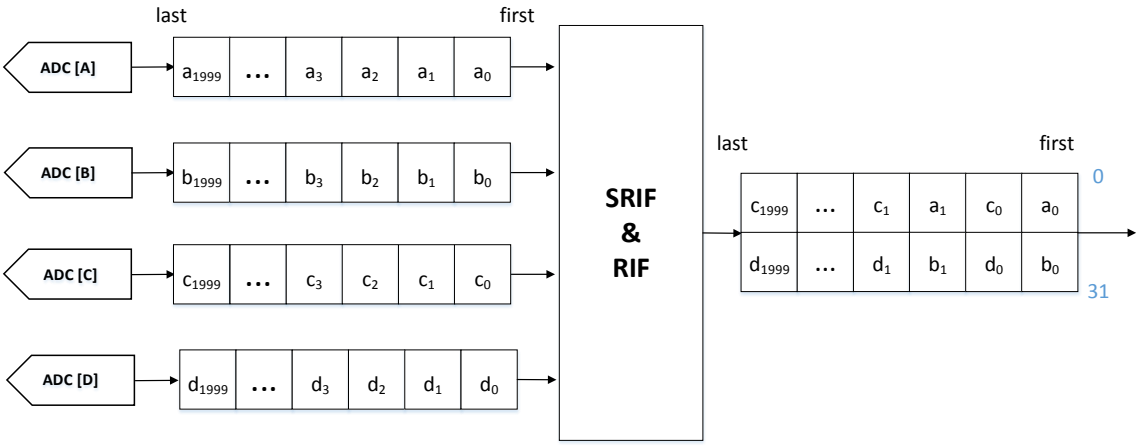

Figure 4.8: Data transfer mode 1.

acquires all samples corresponding to the same depth on the grid. When SRIF acquires sample a_{20} due to a rising edge of the Frame signal, it must also acquire b_{20}, c_{20} and d_{20}. This synchronization must be maintained across all ADCs connected to all four RIFs. The samples must be transmitted lsb first, being that the DFU in RIF shall be configured so that they leave RIF lsb first as well. As mentioned in section 4.2.1, the FLM offers many ways of choosing samples at the input of RIF and ordering them at the output. In data transfer mode 1, RIF shall be operating in single lane mode, which makes it so that the samples will be output as exemplified, two samples at a time (32 bits). This example holds true to any number of samples per pixel up to the maximum allowed of 3200 samples per pixel. On each data lane, after the last sample from a column acquisition must follow the first sample generated by that same ADC on the next column acquisition.

4.2.4.2 Data Transfer Mode 2.1

Data transfer mode 2.1 is the first option for transferring data using sensor connection mode 2. Still considering that each pixel contains a total of 2000 samples, Figure 4.9 exemplifies how data from two ADCs connected to the same LVDS lane must be multiplexed.

All constraints referred in section 4.2.4.1 still apply, being that in this mode the first samples from ADCs A,C,E and G must be synchronized and followed by the first samples from ADCs B,D,F and H, continuing analogously for the rest of the samples. Each multiplexer must output one sample from each ADC at a time, effectively intertwining all samples from two ADCs in the same data lane.

RIF is also set to operate in single lane mode with four active data lanes, meaning that it outputs two samples at a time, alternating from the first two and the last two data lanes. However, since eight ADCs are connected to one RIF instead of four, the output will be ordered as illustrated in Figure 4.9.

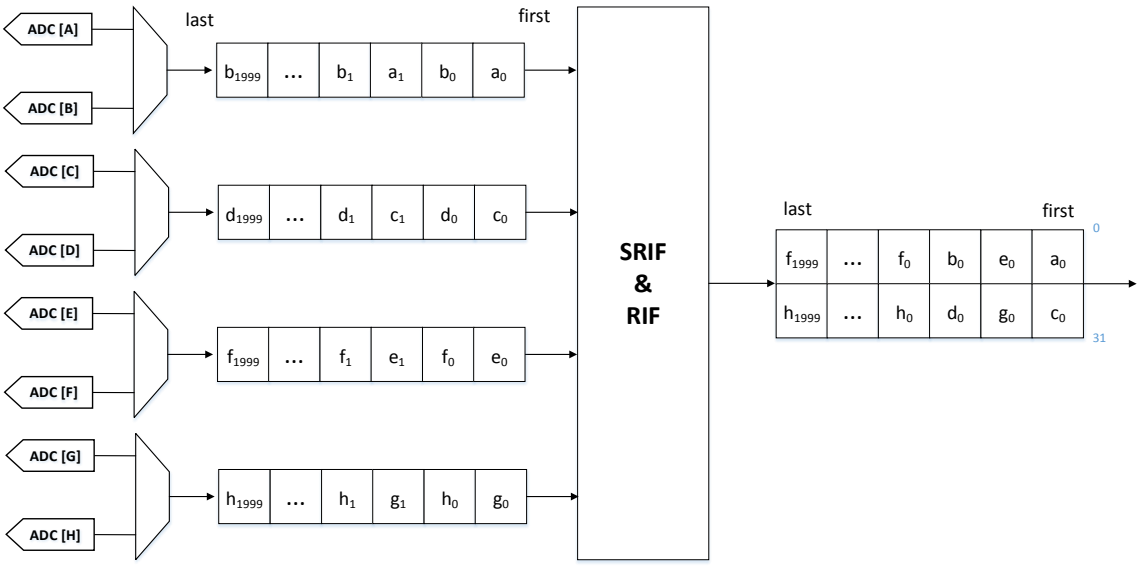

Figure 4.9: Data transfer mode 2.1.

4.2.4.3 Data Transfer Mode 2.2

Data transfer mode 2.2 is the other option available when in sensor connection mode 2. In data transfer mode 2.1, the select signal on each multiplexer would have to change every time a sample is sent through the LVDS lane, precisely synchronizing with the switch in samples from the ADCs and potentially consuming more dynamic power due to the constant switching on all multiplexers. However, in mode 2.2 that is not the case, since each ADC must send all data acquired after a light pulse emission and only after may the other ADC send all the gathered data in that same column acquisition. This mode is exemplified in Figure 4.10, being that the connection from the ADCs to the SRIF and RIF block is the same as in Figure 4.9, except for the order in which the samples are transmitted.

This mode follows the same logic as the previous two in choosing how to order the samples at the output: alternating between the first two and the last two data transfer lanes. However, given how the ADCs multiplex samples on each data lane, any subsystem connected to the output of RIF will only receive the first samples from ADCs B,D,F and H after receiving all samples from ADCs A,C,E and G for the same column acquisition.

4.3 Assembler

The purpose of the assembler subsystem is to make it so that the rest of the system is transparent to the sensor connection modes and consequently the data transfer modes. The *Mode* input signal, observed in Figure 4.11, indicates which is the mode in use and the assembler operates accordingly.

The three supported data transfer modes send the samples ordered in different ways. If the system processed the samples as they were output by RIF, they would be stored in memory in

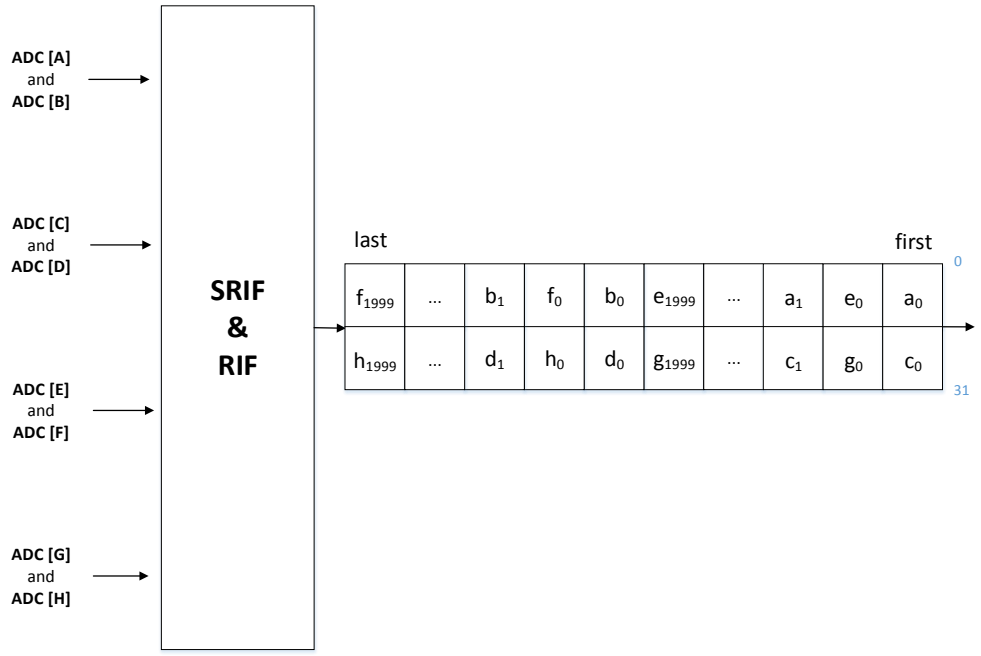

Figure 4.10: Data transfer mode 2.2.

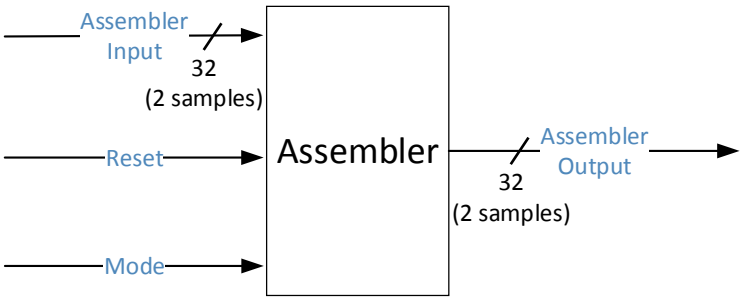

Figure 4.11: Assembler block diagram.

an incoherent manner for any following system that would then read the frame from memory for further processing. In order to process and store the data in a perceptible way, the samples would either need to be ordered before storing in memory, by the memory manager, or in a separate subsystem. Doing so in the memory manager would unnecessarily increase its complexity, bearing too much of the systems functions on one subsystem, thus the assembler was designed right after RIF, making it so that the following subsystems could function as if the input samples would always be ordered the same way.

Every clock transition, the assembler receives two 16-bit samples from RIF and outputs also two 16-bit samples, coupled in a 32-bit bus. Independently of the order the samples come from RIF, using *Mode* as the control signal, the assembler always outputs two consecutive samples from the same ADC on each clock transition, looping through all the ADCs connected to the system. For example, if the system is in data transfer mode 1, sampling grid column 9 and on

the 4^{th} sweep, with 2000 samples per pixel, the assembler will output samples in the following order: $a_0^*[9,4], a_1^*[9,4] - b_0^*[9,4], b_1^*[9,4] - c_0^*[9,4], c_1^*[9,4] - d_0^*[9,4], d_1^*[9,4] - a_2^*[9,4], a_3^*[9,4] - b_2^*[9,4], b_3^*[9,4]$ and so on, until the last samples from the pixels in column 9, namely $d_{1998}^*[9,4], d_{1999}^*[9,4]$, are sent and then the samples from the next column would follow.

To guarantee this output sample order, the assembler must buffer the first samples until it can output the correct sample pairs at the same rate as the input. The required buffer size differs depending on the data transfer modes presented in section 4.2.4. Operating in the data transfer mode 1, a $32 \times 4 = 128$-bit buffer would suffice to gather samples $a_0^*[0,0], b_0^*[0,0]$ to $c_1^*[0,0], d_1^*[0,0]$ (see Figure 4.8) after which the assembler picks samples from the buffer, producing the desired output order. When in data transfer mode 2.1, the assembler must gather from samples $a_0^*[0,0], c_0^*[0,0]$ to $f_1^*[0,0], h_1^*[0,0]$ (see Figure 4.9) before being able to start outputting. This corresponds now to a $32 \times 8 = 256$-bit buffer. However, in data transfer mode 2.2, since each 2 ADCs are multiplexed in the same lane and one can only send data after the other one has finished sending all the samples from a column acquisition, the assembler must buffer from samples $a_0^*[0,0], c_0^*[0,0]$ to $f_1^*[0,0], h_1^*[0,0]$ before outputting. This implies a much greater buffer size of $32 \times 4000 + 32 \times 4 = 128128$ bits, which corresponds to a 16.016 kB buffer (see Figure 4.10). For this reason, in order to support all three data transfer modes, the assembler subsystem incorporates a 16.016kB buffer.

4.4 AVG

The AVG subsystem is responsible for effectively averaging the LiDAR samples. Given the nature of the operations this system needs to perform, it is designed as a combinational circuit. Namely, this system receives a 32-bit input from the assembler which consists of two 16-bit samples (with zero-padding) and another 32-bit input from the memory manager block with the corresponding two samples obtained in the previous sweep. The AVG circuit adds each two corresponding samples and, depending on a signal from the controller $-Last\_sweep-$ indicating if the addition result must be divided in order to complete the averaging process, it either performs the division or not. The system was designed to perform averaging by 2, 4 or 8 so that the division can be done by right-shifting the samples, which can be done by a combinational circuit. However, note that the binary right shift operation corresponds to performing an integer division on the corresponding decimal values of the samples, which implies that after finishing the averaging process for any pixel on the grid, all samples will lose 1-bit resolution. Figure 4.12 illustrates how this system works.

Following the example illustrated in Figure 4.12, the AVG must split the two 32-bit inputs into the corresponding 16-bit samples and add $a_{50}^*[5,2]$ with $a_{50}[5,1]$ and $a_{51}^*[5,2]$ with $a_{51}[5,1]$ resulting in samples $a_{50}[5,2]$ and $a_{51}[5,2]$. If averaging by 2 was desired for example, the $Last\_sweep$ signal would indicate the AVG to right shift both resulting samples by one bit, effectively dividing each sample by 2. Finally the two resulting samples $a_{50}[5,2]$ and $a_{51}[5,2]$ would be grouped and output in a 32-bit parallel bus.

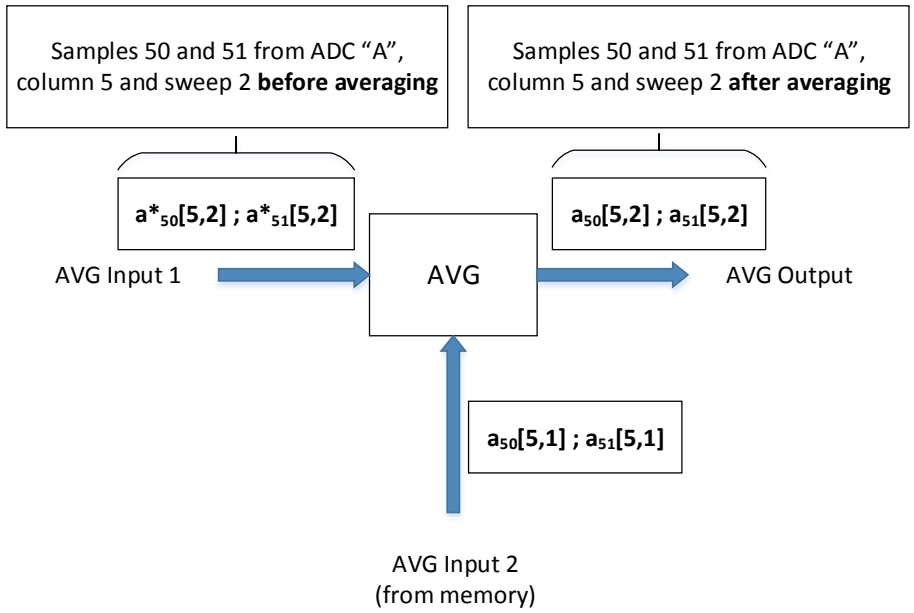

Figure 4.12: AVG subsystem schematic.

At every clock rising edge transition, the AVG system receives a new 32-bit input from the assembler containing two samples and also a new 32-bit input from the memory manager containing the same two samples obtained in the previous sweep. This timing behavior is better exemplified in the time diagram in Figure 4.13.

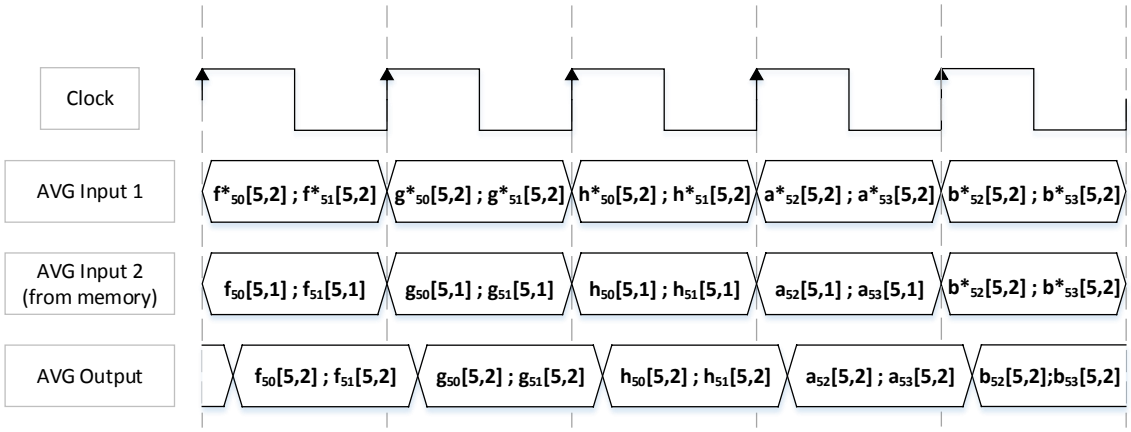

Figure 4.13: AVG time diagram.

This time diagram is an excerpt from the full time diagram of the averager system, seen in annex B.1, working in sensor connection mode 2, meaning that eight ADCs are connected per RIF. This diagram shows how at every clock the two inputs have the samples that must be averaged, producing a result that appears at the AVG output with a small delay due to the propagation delay of the combinational circuit. As illustrated, once the 50^{th} and 51^{st} samples from all eight ADCs are processed, the following two samples, 52^{nd} and 53^{rd}, start incoming sequentially from ADC

A to H. The AVG system has no capacity of controlling what samples appear at the inputs every clock transition, that functionality is built in how the remaining subsystems operate.

4.5 Controller

The controller subsystem, seen in Figure 4.14, is responsible for interfacing with the outside, receiving information about how the averager shall operate, but also for performing some overriding actions on other subsystems, like resetting them.

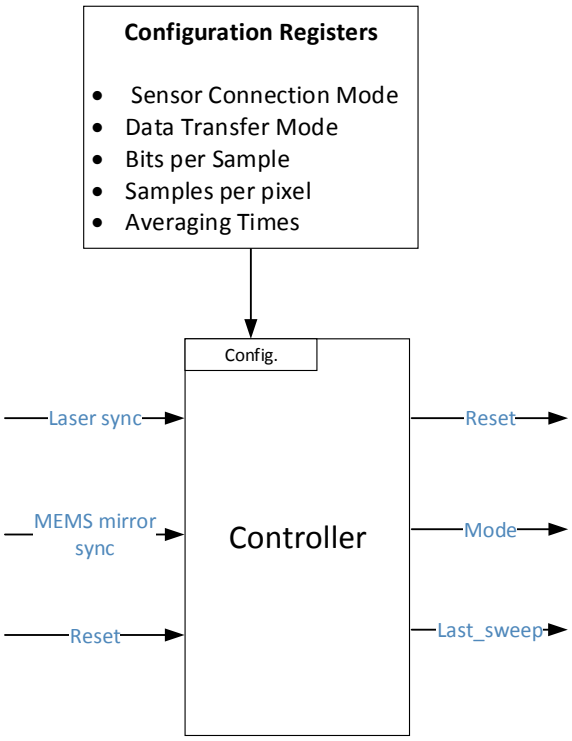

Figure 4.14: Controller subsystem and configuration information.

All the information described in Figure 4.14 – the sensor connection and data transfer modes, the amount of bits per sample generated by the ADCs, the amount of samples per pixel and the desired averaging must be loaded into the configuration registers prior to any processing by the system. The controller then compiles this information into the *Mode* output signal providing the other subsystems relevant information, like how the output samples from RIF must be ordered and how much memory the system will use.

The *Laser sync*. and *MEMS mirror sync*. input signals must be provided to the controller as a reliable way of knowing whenever a light pulse is fired and keeping up with the mirror movement with precision. The controller processes the information from these two signals as the system functions and indicates through the *Last_sweep* output precisely when the mirror is completing the last sweep through a set of 20 columns, so that the AVG subsystem can correctly complete the averaging process. Finally, the *Reset* signal is input by some other system with power to control

the averager system and the controller distributes the signal, synchronized with the internal clock, to the remaining subsystems.

4.6 Memory

The following section will describe all memory aspects of the developed system. Specifically, how the subsystem responsible for addressing the SRAM functions and how the grid data is stored in memory.

The challenges involving memory in the development of the system proved crucial because several aspects had to be taken into account simultaneously. Namely, a total of 6 MB of SRAM is available for the system, which is composed of individual memory banks that can be interfaced with the averager system in many ways, yet considering that the grid data must be stored in a comprehensible way. Each memory bank can store a total of 256 kB in which each memory address accesses 64 bits at once. Accessing a memory bank (reading or writing) can occur at a maximum frequency of 250 MHz and up to four banks can be addressed simultaneously, considering that a decoder would be used so that a single input address could be decoded into four different values, one for each bank. This way, a total of 256 bits can be accessed simultaneously, noting that accessing more than four banks at the same time would present considerable hardware routing challenges and any solution doing so should be discarded. Moreover, it must be considered that any reading or writing instruction on a memory bank requires two clock transitions, i.e., the first transition instructs a read or write action on a specific memory address of the bank and only on the following clock transition will the respective data be available.

The initial considerations regarding interfacing with the SRAM consisted in defining how each sample would be stored in memory and whether the system would allow for scanning the entire frame first and only then perform averaging or average a set of grid columns first before moving on to the next set of columns. In order to allow for input samples with 8 to 16 bits, an initial analysis was made considering that each sample would be stored as 32 bits, two samples per memory address, performing the necessary amount of zero-padding to store each sample as 32 bits. Considering this, if the entire frame is gathered before averaging, the system would need to store data corresponding to the entire grid, which would correspond to 32 bits $\times$ number of samples per pixel $\times$32 grid lines $\times$120 columns that, considering a minimum of 1000 samples per pixel and a maximum of 3200, would correspond to either 15.36 MB or 49.152 MB of required memory. These values are clearly beyond the maximum limit of 6 MB. However, averaging 20 columns at a time would reduce those values to 2.56 MB and 8.192 MB indicating that it could be a possible implementation if a lower limitation on the number of samples per pixel was applied. Yet, storing all samples as 32 bits would still entail a very large waste of memory in zero-padding and the amount of memory required would still be noticeably high. For this reason, it was decided to store all samples as 16-bit samples, effectively storing four samples per memory address. An implication of this decision is that the number of bits per sample at the input would have to be reduced and that the amount of averaging supported by the system would reduce as well. This

way, the number of supported bits per sample at the input resulted in 8, 10 and 12, which are the more common values in LiDAR applications and a maximum of averaging by 8 was imposed since averaging by 16 means that the FPS output of the system would reduce immensely. Storing an entire frame first, with anything from 1000 to 3200 samples per pixel, would reduce by half the initially considered amount of memory required, corresponding to a total of 7.68 MB and 24.576 MB respectively, which are still unfeasible values. This way, the system was designed so that 20 columns would be averaged at a time, allowing for a maximum of 3200 samples per pixel, consequently using a maximum of 4.096 MB of memory, achievable by using exactly 16 memory banks: $16 \times 256\,\text{kB} = 4.096\,\text{MB}$.

4.6.1 Memory Manager

In order to control the access to the SRAM by the averager system, the memory manager subsystem was developed. It must synchronize all reading and writing instructions on all memory banks connected to it. It must do so obeying to all restrictions mentioned, regarding memory aspects. The individual subsystem is presented in Figure 4.15 with all corresponding inputs and outputs.

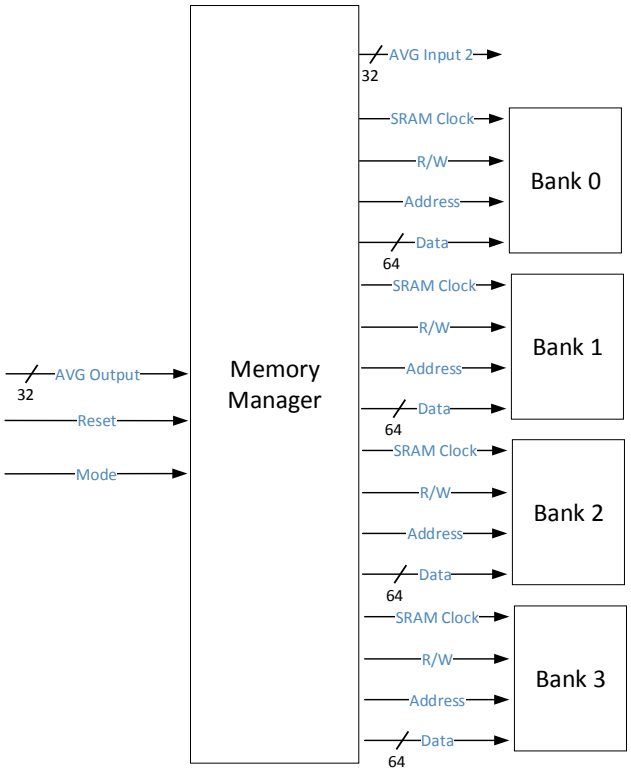

Figure 4.15: Memory manager block diagram.

Four connections are involved in interfacing with each memory bank: SRAM clock, data bus, address bus and the R/W signal. The R/W signal indicates whether the reading or writing instruction is required with a logic value 0 or 1 respectively. The address bus indicates on what memory address that instruction shall be done. The data bus is where the sample values are to be

written when a writing instruction is done on the SRAM memory and also where the read sample values will output after a reading instruction. Finally, the SRAM clock signal instructs the memory bank to perform the action indicated by the other three signals.

The memory manager takes in the output from the AVG block and writes those samples in the corresponding memory bank and corresponding address position. Also, it simultaneously reads from the necessary memory bank and address in order to output the samples that the AVG block will require, sending them through the *AVG Input* 2 connection, two samples at a time. The reset signal triggers the reset mechanism making the subsystem restart in which all the memory banks connected to the memory manager must hold only samples with value zero. This happens so that when the first sweep over a set of 20 columns is performed, the AVG block adds all incoming samples with zero, therefore not altering the value of the samples from the first sweep. The mode signal is used to indicate in what sensor connection mode the system is operating in and how many samples per pixel are being acquired, consequently dictating how the memory accessing is done.

The time diagram in Figure 4.16 demonstrates how the system shall perform when in sensor connection mode 2 (8 ADCs per RIF), how it addresses two of the four utilized memory banks, providing the correct samples to the AVG subsystem, which in turns averages the samples and outputs them on the *AVG Output* bus. In turn, the memory manager receives these samples and stores them in the correct bank and address.

In order to observe the full behavior of the system, addressing all four memory banks while in sensor connection mode 2, refer to annex B.1. As demonstrated by the diagram in Figure 4.16, the memory manager accesses each memory bank individually requiring no instruction decoding to read or write simultaneously from multiple memory banks. As the diagram also suggests, the memory manager must include a 64-bit buffer for each memory bank so that four samples can be gathered before simultaneously writing them all to the same position of the same memory bank. Moreover, a different 64-bit buffer shall be used per memory bank to receive the read samples sending them, two at a time, to the *AVG Input* 2 bus.

4.6.2 Memory Accessing Mode 1

Given the two sensor connection modes allowed, corresponding to a total of 4 or 8 ADCs being processed by each averager block (see Figure 4.1), each memory manager will address either two or four memory banks. In case of addressing only two banks, the averager system will not have to store as much data, therefore using half the memory banks. Yet, it will take less time between two consecutive instructions on the same bank. This scenario is illustrated in Figure 4.17.

Each memory bank is read and written to in alternating fashion. Opting between sensor connection modes 1 or 2 translates into a compromise between reducing the amount of memory used and increasing the frequency at which the memory banks are addressed.

The scheme in Figure 4.17 repeats for the remaining three averagers in the overall system.

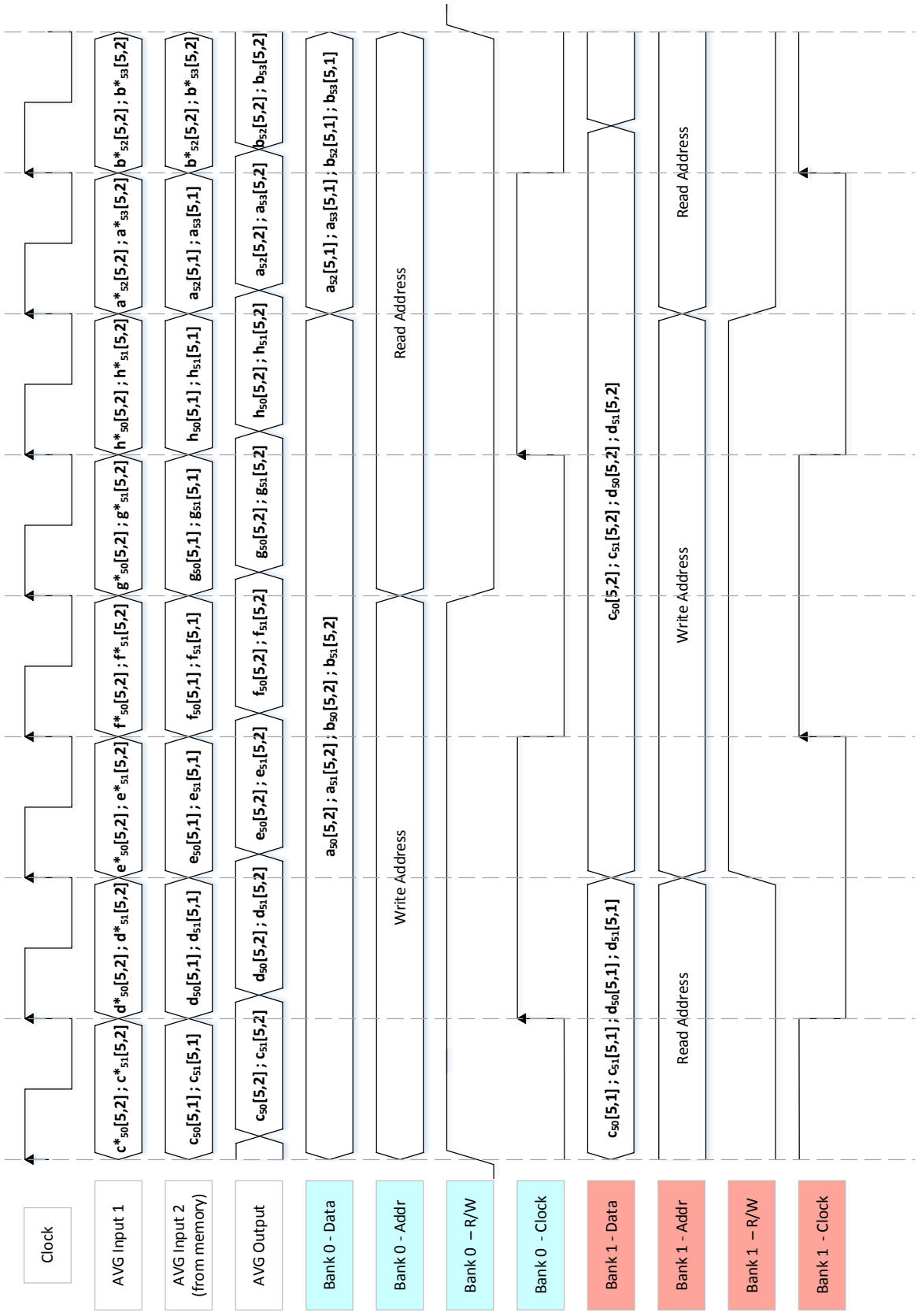

Figure 4.16: Averager time diagram operating in sensor connection mode 2 and accessing to the first two memory banks.

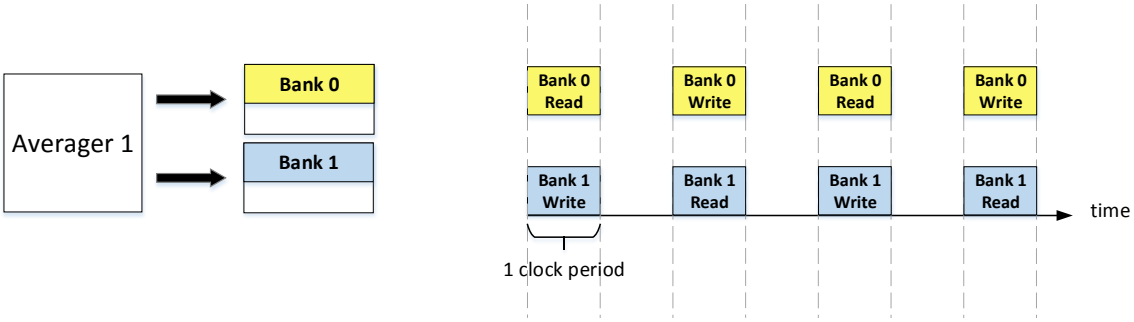

Figure 4.17: Memory accessing in sensor connection mode 1 by a single averager.

4.6.3 Memory Accessing Mode 2

Figure 4.18 illustrates now the synchronization in accessing four memory banks by an averager system, when in sensor connection mode 2.

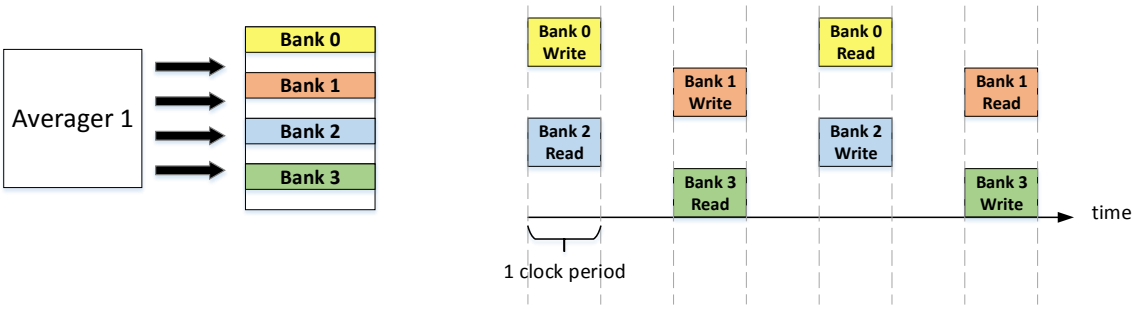

Figure 4.18: Memory accessing in sensor connection mode 2 by a single averager.

Contrary to the memory accessing mode 1, each memory bank is addressed every four clock cycles, instead of two. This happens due to the fact that the averager spends more time accumulating and writing unrelated samples since there are now eight ADCs connected to the averager instead of four.

4.6.4 Memory Mapping on 1 Bank

Independently of the sensor connection mode, the memory manager will always follow the same logic in storing samples on each memory bank. Figure 4.19 illustrates this distribution for the first memory bank connected to the first averager system, in which there are 2000 samples per pixel, as an illustrative example.

Each memory bank stores all samples corresponding to 2 grid lines and 20 columns. In the example shown in Figure 4.19, the first two grid lines, corresponding to ADCs A and B are stored in the memory bank, with a color delineation of where the pixels from one column end and where the ones from the next column begin. In blue are represented all 2000 samples from ADC A originated on the first column acquisition and also 2000 samples from ADC B from the same

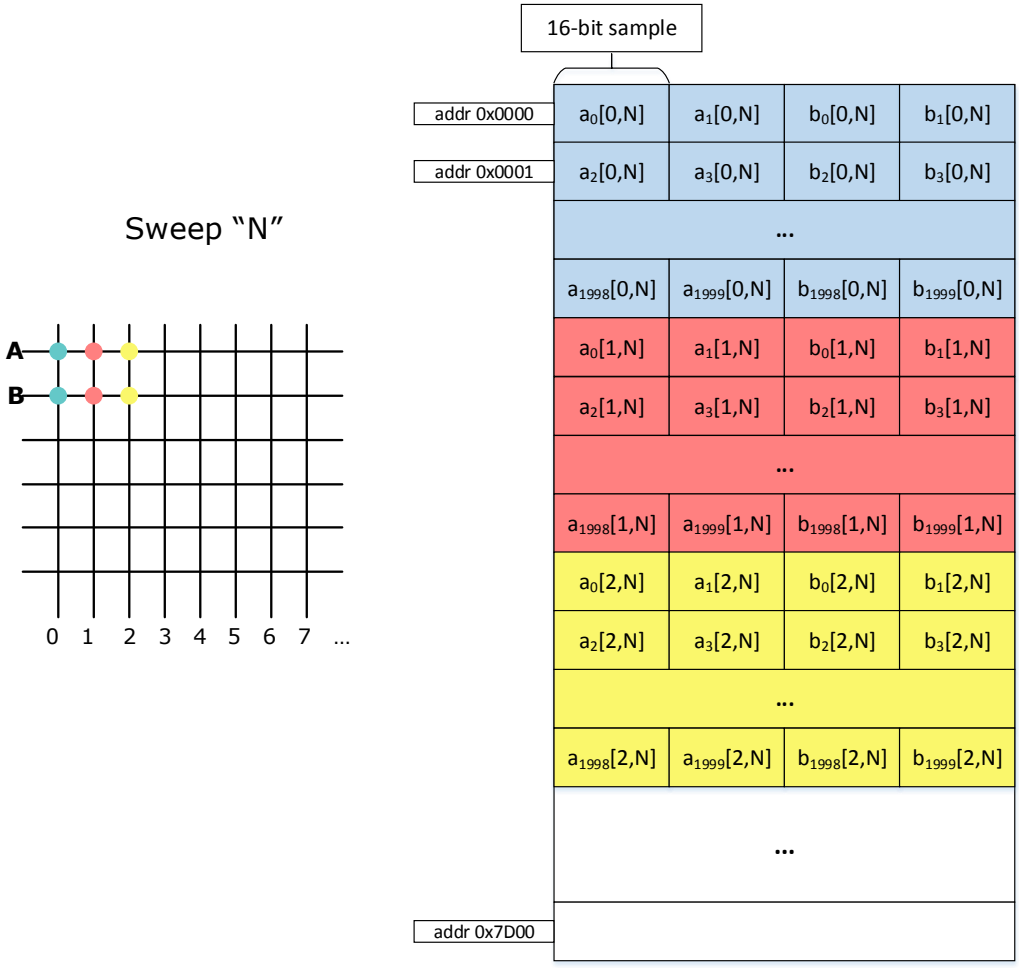

Figure 4.19: Memory mapping in one memory bank and corresponding grid data.

column. The following colors correspond to the pixels on the following columns, in which all 20 columns are stored in the same memory bank.

The memory manager efficiently adapts to how the samples are processed by the AVG, in which two consecutive samples are output from one ADC at a time. For that reason, on any used address of the memory bank, it can be found two samples from an ADC and the next two samples but from the next ADC. Considering an isolated memory bank, the instructions occur in a simple loop (see Figure 4.16): first, four samples are read from an address, then the samples resulting from averaging the samples from sweep N+1 and the ones from sweep N are stored in that same address and finally, this process continues for all following addresses until the last one being used is reached and the loop restarts at the first memory address.

However, the entire memory bank is not necessarily used. Two aspects contribute to storing irrelevant information or not using available memory: the number of bits per sample generated by the ADCs and the number of samples per pixel. For this reason, a distinction was made between unused memory and wasted memory.

Memory waste is considered the amount of memory allocated to store samples that ends up not being used. Since this value is the same on every memory bank, its percentage value can be obtained directly from the relation between the number of unused bits and the number of bits allocated for storing a sample:

$$Memory\ waste(\%) = \frac{Nr.\ Unused\ bits\ per\ sample}{16} \times 100 \tag{4.6}$$

To calculate the number of unused bits per sample, one must acknowledge that in the averaging process samples are added and stored until the last addition is performed and only then are the bits on the sample shifted right to divide by 2, 4 or 8, completing the average. Therefore, as an example, if 8-bit samples are used and the system will average by 4, for the first three additions the sum result will have to be stored in memory. Since the maximum value representable with 8 bits is $2^8 - 1 = 255$, that result can potentially take the value of $255 \times 3 = 765$ which is represented by the 10-bit binary value 1011111101. In this scenario the number of unused bits would then be $16 - 10 = 6$ bits, resulting in a total percentage of memory waste, using (4.6), of 37.5%. Following this definition, the amount of memory waste allowed by the system are the ones presented in table 4.1.

Averaging times	Bits per sample on input	Memory waste (%)
2	8	50
	10	37.5
	12	25
4	8	37.5
	10	25
	12	12.5
8	8	31.25
	10	18.75
	12	6.25

Table 4.1: Allowed values of memory waste.

However, as exemplified by Figure 4.19, depending on the number of samples per pixel a certain portion of each memory bank will remain unused. The system offers possibilities that in fact use the entirety of this memory space (using 3200 samples per pixel) and for that reason it is considered unused memory instead of wasted memory. Once again, since the amount of unused memory will be the same on all memory banks, its percentage value can be analyzed through only one bank. Specifically, the percentage of unused memory can be calculated by the ratio of unused memory and total memory in a memory bank. Each memory bank can hold 256 kB, which can be expressed as 16 bits $\times$3200 samples per pixel $\times$2 pixels per bank $\times$20 columns = 2.048 Mbit = 256 kB. The amount of memory used in a bank is given by 16 bits $\times$ number of samples per pixel $\times$2 pixels per bank $\times$20 columns. Therefore, since the unused memory depends only on the ratio between the number of samples used and the number of samples the bank can support, (4.7)

is used to calculate the percentage of unused memory.

$$Unused\ memory(\%) = (1 - \frac{samples\ per\ pixel}{3200}) \times 100 \tag{4.7}$$

4.6.5 Memory Mapping Mode 1

Even though the logic for storing samples on one memory bank is the same, whatever the sensor connection mode is, the two connection modes imply that each averager interfaces with a different number of ADCs, consequently using a different amount of memory banks (2 or 4), which results in that the 20 columns from the grid are stored differently across all memory banks.

This way, following the explained logic for storing samples from two pixels in a memory bank, as well as the assumption that 2000 samples are generated per pixel, one can intuitively deduce that one averager stores the grid data as exemplified in Figure 4.20.

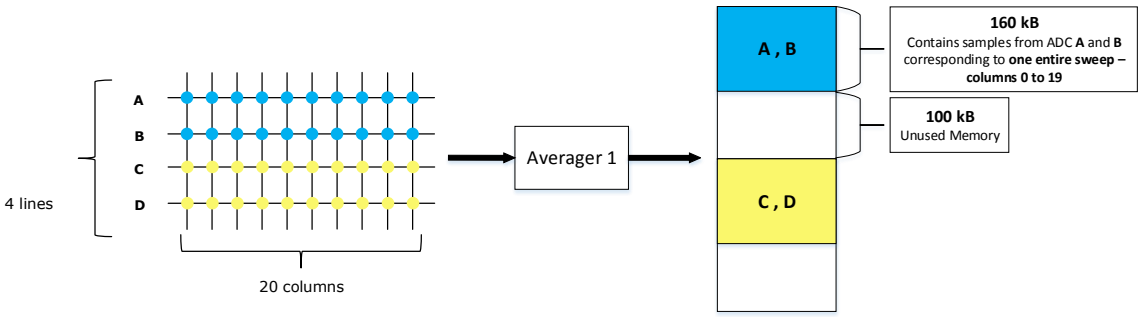

Figure 4.20: Memory mapping by one averager operating in sensor connection mode 1.

All pixel data generated by ADCs A and B for 20 columns is stored in the first memory bank (data in blue) and the samples generated by ADCs C and D, corresponding to the third and fourth grid lines, are stored in the second memory bank (data in yellow). The disposition of the samples in each memory bank is as described in section 4.6.4.

This way, zooming out to encompass the entire system with 4 averager blocks, the schematic in Figure 4.21 can be used to describe how the entire 20 columns data, on a grid with 16 lines, will be distributed on the SRAM memory.

Each averager in the schematic addresses two memory banks represented by the same color, accounting for a total of eight memory banks used to store all the data. Once the data from the following 20 columns starts being sampled, the memory will be used exactly in the same way as described so far to store those new 20 columns, overwriting the data previously stored in the memory banks.

4.6.6 Memory Mapping Mode 2

As already mentioned, once in sensor connection mode 2, each averager will address four memory banks instead of two. Consequently, an averager will store data corresponding to a total of 8 lines

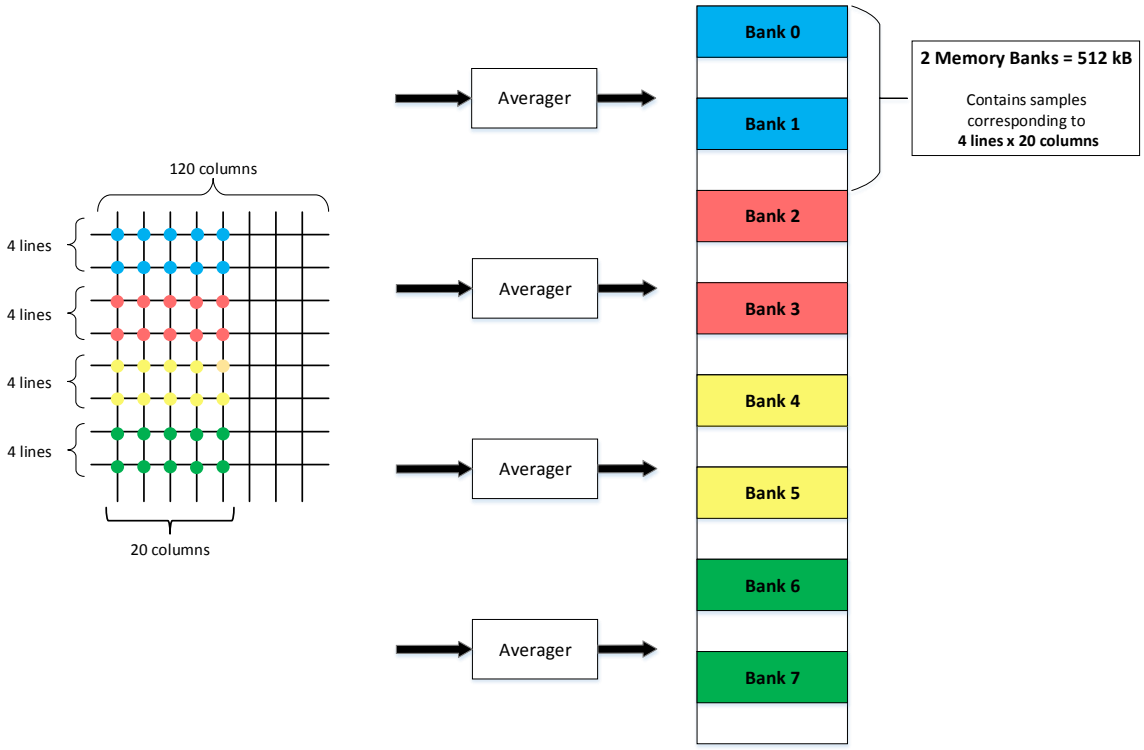

Figure 4.21: Memory mapping by four averagers operating in sensor connection mode 1.

and 20 columns, as represented in Figure 4.22, in analogous fashion as described in section 4.6.5.

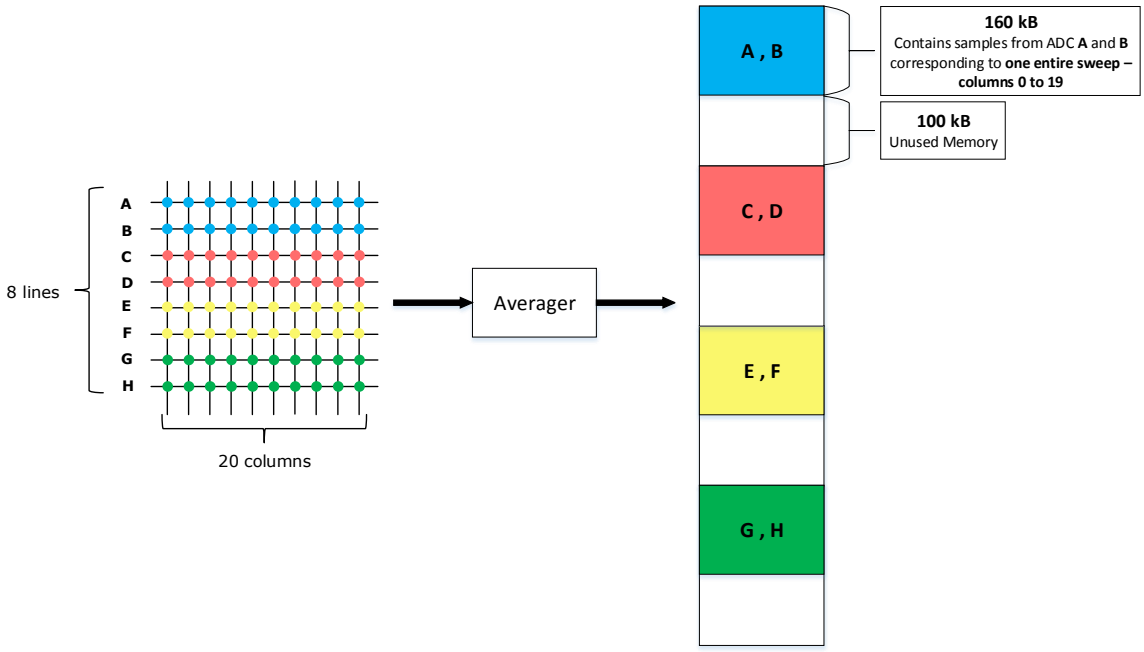

Figure 4.22: Memory mapping by one averager operating in sensor connection mode 2.

Finally, observing all 4 averagers and corresponding 16 memory banks in Figure 4.23, it is exemplified how the data from a grid with 32 lines, 120 columns and 2000 samples per pixel is stored in all memory banks.

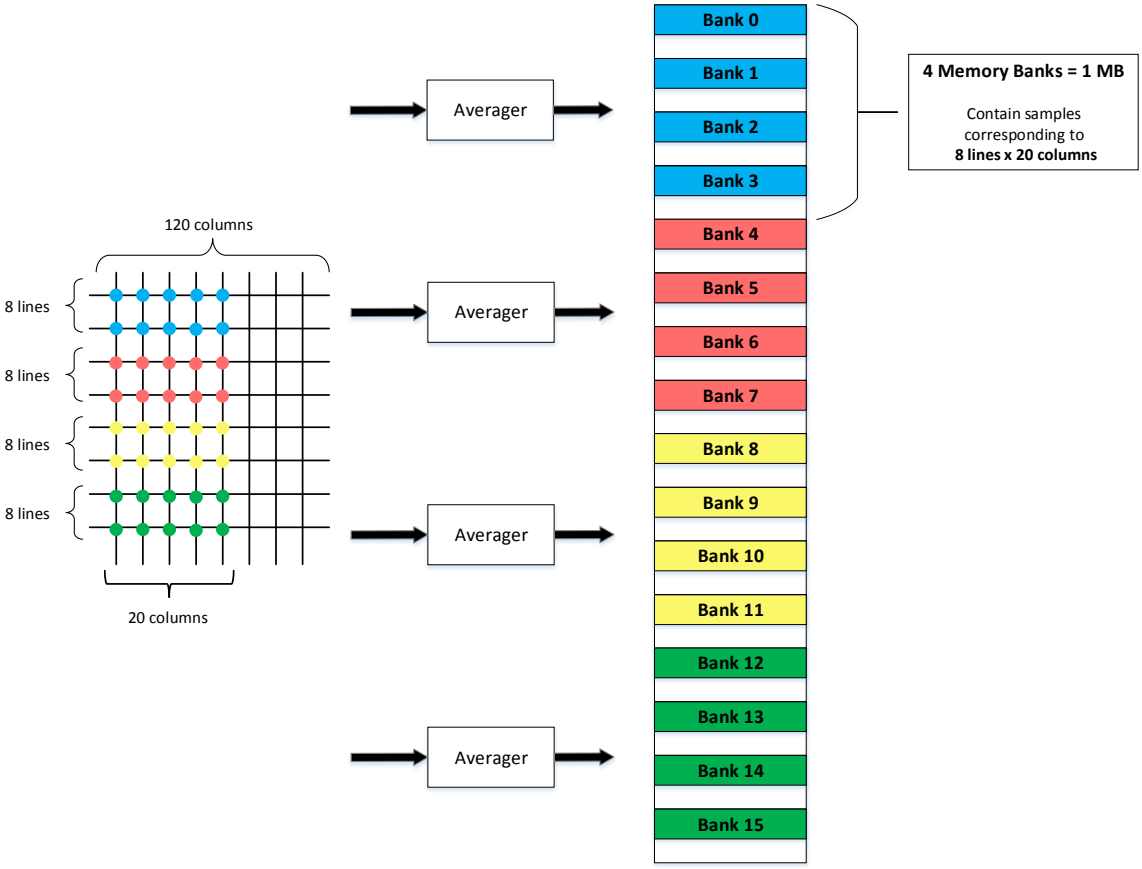

Figure 4.23: Memory mapping by four averagers operating in sensor connection mode 2.

Chapter 5

Simulation Results

This chapter will describe the developed Matlab simulation of the proposed system. This simulation aims to test the basic functional aspects of the system like memory mapping the LiDAR samples correctly, whether the signal averaging technique is being correctly applied or not and how the point-cloud representation is to be built from the averaged samples. It must also test performance aspects of the system like the achieved FPS output, the clock signal frequency and whether the amount of allocated SRAM suffices for the system as well as how much of it is being utilized.

5.1 Simulated System

This section details the parameters of the simulated system and how that was achieved.

To build a simulation of the proposed system as realistically as possible, the input signals from the sensor must be accurately simulated as well. For that purpose, a Matlab script developed in analyzing the LiDAR sensor with greater detail was used. This script allowed to generate realistic and personalized signals coming from a LiDAR sensor that would then be used by the computation model simulation. Many signal parameters can be defined, namely the ADCs sampling frequency and bits per sample, the sampling duration of the sensor that in turn dictates the depth range covered by the system, the duration, shape and intensity of the emitted light pulse and also the model used to simulate the noise signal. Furthermore, realistic models for the TIA and ADCs were used, which incorporate the inherent added noise to the acquired signal. Finally, the SNR of the signal produced by the sensor could be defined, as well as the detection of a light pulse reflection included in the received signal, corresponding to an object at an adjustable distance from the sensor.

The simulated system consists in a single averager, since the remaining three averager subsystems behave the same way, all synchronized with each other, only for different lines of the grid. For the same reason, only the processing of data from a set of 20 columns was simulated. Since the samples from the following 20 columns are transmitted in exactly the same order and without

interruptions between samples, the system would perform exactly the same operations but only for a different set of columns.

A grid with size 8 lines $\times$ 20 columns was simulated. This implies that the system is operating in mode 2 and the sensor is connecting to the microcontroller using sensor connection mode 2, with four LVDS connections per RIF, each transferring data at a 1 Gbps rate. The ADCs output 8-bit samples, at a sampling frequency of 1 GHz, for a duration of 2 μs, which implies that the grid has 2000 samples per pixel, translating into a depth range of 300 m. The system will perform averaging by 8, meaning that the sensor will sweep through the same set of 20 columns eight times and those eight acquisitions are combined to produce the point-cloud representation of that portion of the grid. Each light pulse emitted was simulated with a Gaussian shape and a duration of 10 ns, resulting in the light pulse signal illustrated in Figure 5.1.

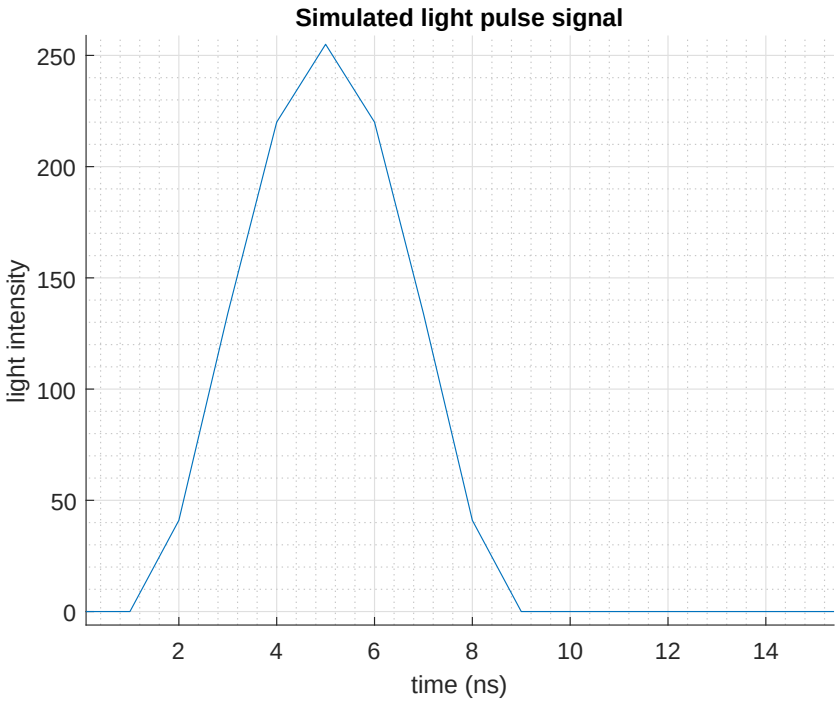

Figure 5.1: Gaussian light pulse generated for a duration of 10 ns with intensity represented by 8 bits as well.

The SNR of the acquired signal was set to 6 dB and the noise signal was modeled as AWGN. Setting an object distanced 20 m from the sensor and simulating the acquisition of a signal containing a light pulse reflection at that distance results in a signal as seen in Figure 5.2. The reflection pulse from an object distanced 20 m from the sensor appears in the plot at the 133.33 ns time mark. However, as observed in Figure 5.2, that peak is barely noticeable and completely indistinguishable from the background noise, indicating that further processing is clearly required.

For each pixel on the 8 $\times$ 20 grid, a signal was generated with an object placed at a predetermined distance from the sensor. For all pixels in the 1^{st} column, an object was placed at a 20 m

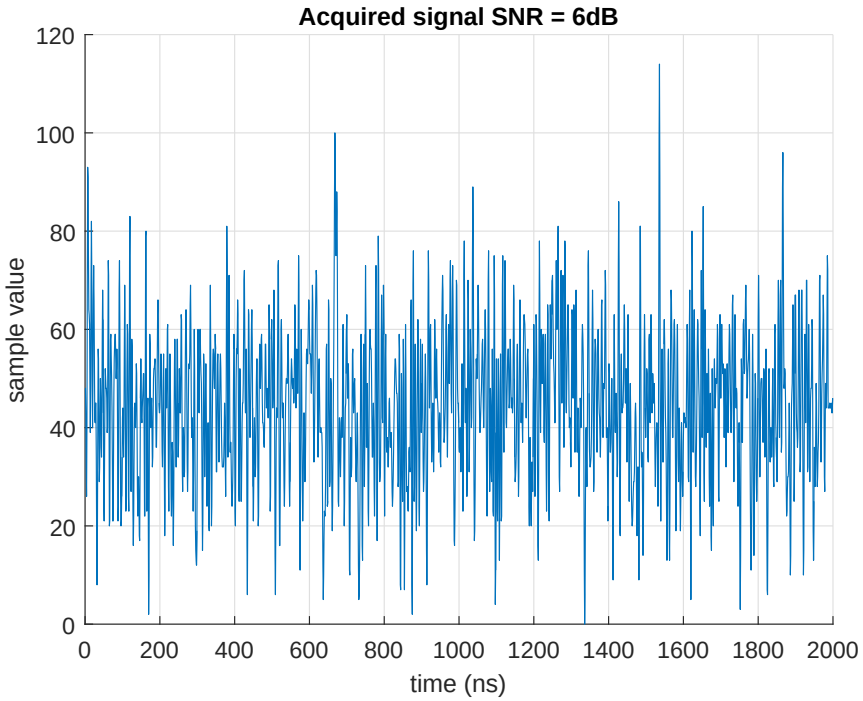

Figure 5.2: Acquired signal simulation with 6 dB SNR and peak at 133.33 ns using an AWGN model.

distance, reusing the same acquired signal for all pixels. For the pixels in the 2^{nd} column the object is at 40 m and the distance keeps increasing by 20 m until the 10^{th} column, which detects an object distanced 200 m from the sensor. Then, for the last ten columns, the pattern repeats in decreasing order. The 11^{th} column has an object at 200 m, the 12^{th} has one at 180 m and so on until the 20^{th} column in which all the pixels from that column will have an object at a distance of 20 m again. This process was repeated for all eight sweeps of the grid, generating new signals but always following the same pattern for placing objects in front of the sensor. This way, the data obtained from a sweep was represented as a 3D matrix with dimensions $8 \times 20 \times 2000$ where each position in the matrix contained a sample. Eight such matrices were generated, one for each sweep.

Each sample is represented by its decimal value, with a maximum of 255, given by (3.12), since each sample has 8 bits. However, in the averager system, all samples are processed and stored as 16-bit values, being that on each memory bank address four such samples can be stored (64 bits). Additionally, each memory bank can hold a maximum of 3200 samples per pixel $\times 20$ columns $\times 2$ pixels, which results in 128000 16-bit samples (256 kB). Since each address holds four samples, this means that each memory bank has a total of $\frac{128000}{4} = 32000$ memory addresses. For this reason, each memory bank is initialized as a 2D matrix with size 32000×4 containing only zero values.

Finally, the averager system was simulated, picking samples from each pixel in the same order

as the assembler outputs them (described in section 4.3), averaging them and storing them in memory, following the topology described in section 4.6.4 for storing samples on each bank and distributing the grid data between all banks as defined in memory mapping mode 2.

5.2 Performance Analysis

In processing the data for the portion of the grid in consideration and for the eight sweeps, a record was kept in the *clock_counter* variable, of all clock rising edges during the simulation. This variable indicates that 1.6×10^5 clock rising edge transitions are required to process one sweep and 1.28×10^6 transitions to finish processing the entire data (eight sweeps). Considering that the four averager systems work simultaneously during that time, by the end of the simulation each averager has processed 8 different lines of the first 20 columns, meaning that all the data from the first 20 columns of the grid has been effectively processed and stored. From the fact that the entire grid has a dimension of 32×120, results that processing the entire grid data takes a total of $1.28 \times 10^6 \times 6 = 7.68 \times 10^6$ clock transitions.

Simulating the system with a 1 Gbps data transfer rate on each LVDS connection and 8-bit samples coming from the ADCs, allows for calculating the clock frequency of the system, using (4.3), which results in $f_{clk} = 250\,\text{MHz}$. This entails that the clock period is $T_{clk} = 4\,\text{ns}$ and it is therefore possible to calculate the time it would take this system to process the entire grid, using the deduced overall clock count and the clock period, resulting in $7.68 \times 10^6 \times 4\,\text{ns} = 30.72\,\text{ms}$ required to process and store eight sweeps over the entire $32 \times 120 \times 2000 = 7.68 \times 10^6$ samples in the grid. This results in an output rate of $\frac{1}{0.03072} \approx 32.552$ FPS. Going back to the estimated performance of the system, analyzed in section 4.2.2, we can observe that this value is in line with the predicted performance, as demonstrated in Figure 5.3.

The extra time required by the assembler to buffer the first samples before outputting anything induces a delay between sweeping through the grid and processing that data. This delay is illustrated in the time diagram in Figure 5.4. The time required to process the data from one sweep is calculated knowing the number of clock counts for one sweep and the clock signal period, which results in $1.6 \times 10^5 \times 4\,\text{ns} = 0.64\,\text{ms}$, as represented in the time diagram.
As explained in section 4.3, the delay depends on the data transfer mode being used. When using data transfer modes 1 and 2.1 the delay is negligible when compared to the delay resulting from data transfer mode 2.2. The assembler takes four clock transitions in data transfer mode 1 and eight clocks in data transfer mode 2.1 to buffer samples without outputting any for processing. However, in data transfer mode 2.2 it takes 4004 transitions before continually outputting samples to the AVG subsystem. From the number of clock transitions required for each mode and knowing the clock period, the possible delay values for the simulated system are shown in table 5.1. Yet, any of these values is still considerably smaller than the necessary time to process data from one sweep, let alone the entire grid, and for that reason this phenomenon was deliberately neglected in the simulation.

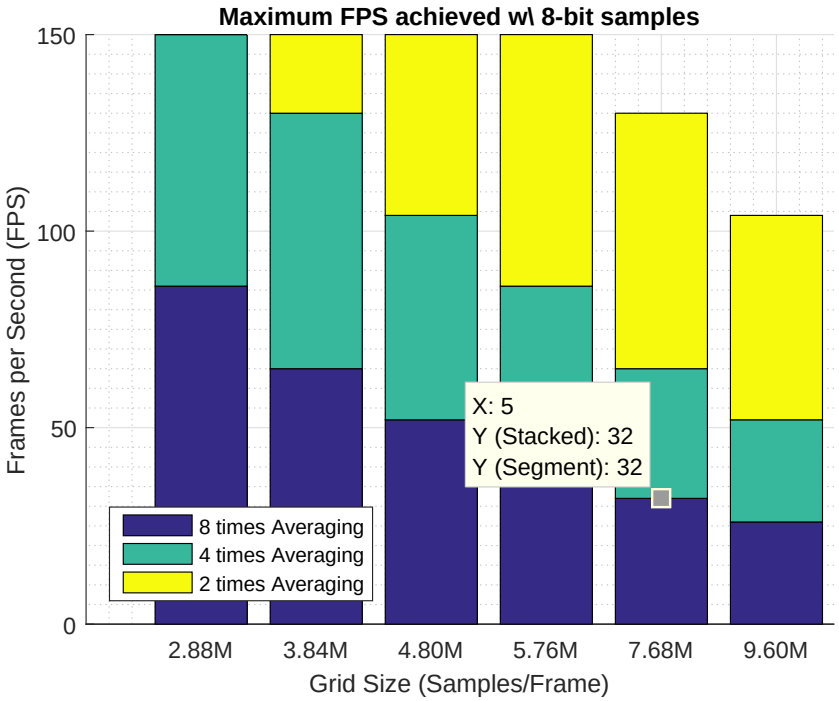

Figure 5.3: Predicted FPS output performance of the system.

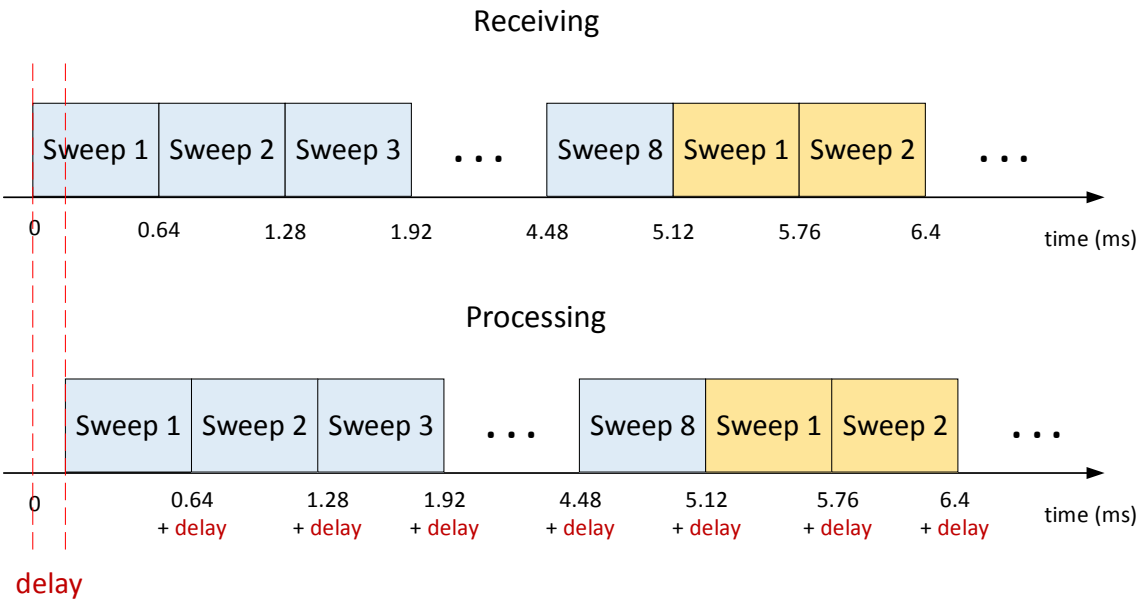

Figure 5.4: Delay between receiving and processing the data.

After the simulated averager subsystem finishes processing all the data corresponding to it, the amount of unused memory is measured directly from the matrices used to simulate each memory bank. That measurement was performed by comparing the amount of addresses containing only zeros with the total memory addresses in a memory bank. Indeed, that value is 37.5%, which is in

Data Transfer Mode	delay (μs)
2.1.	0.032
2.2.	16.016

Table 5.1: Possible delay values for the simulated system.

agreement with equation 4.7, given that the grid in consideration has 2000 samples per pixel.

5.3 Functional Analysis

After processing and storing all the data, one can verify the memory mapping by analyzing the first two samples in the first 1000 addresses of any memory bank, in accordance with the memory mapping topology in one bank (section 4.6.4). The samples obtained from such position on the second memory bank show the averaged signal in Figure 5.5.

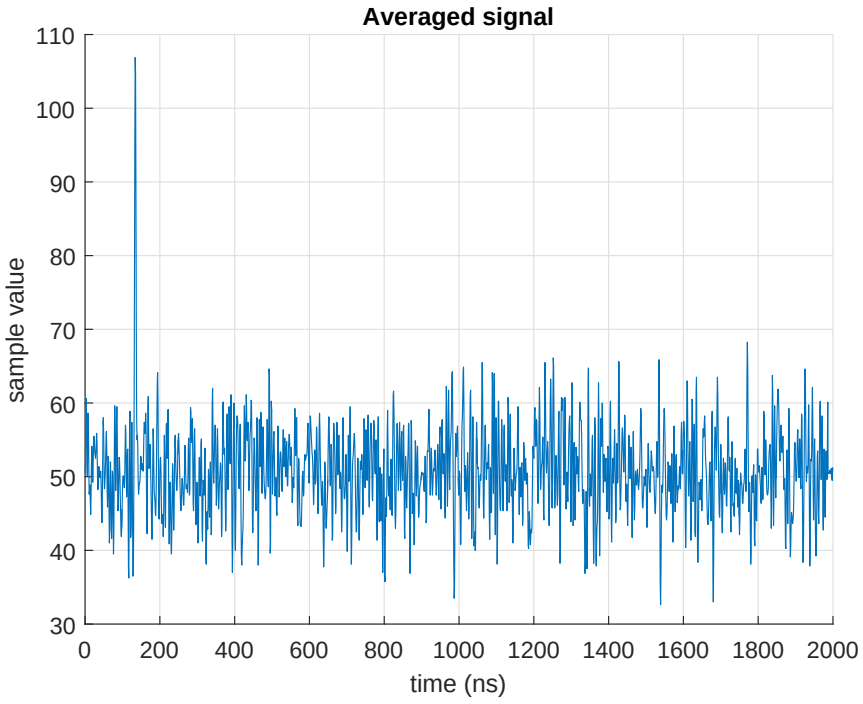

Figure 5.5: Averaged signal read from memory bank 2.

It is intuitive to verify that the peak in the signal is considerably more distinguishable than the one in the acquired signal in Figure 5.2. This averaged signal was read from all the first two samples in the first 1000 addresses of the second memory bank, which corresponds to samples $c_0[1,8]$ to $c_{1999}[1,8]$. This justifies why the peak is detected at approximately 133 ns, which corresponds to

an object distanced at 20 m, according to (1.1), as is the case for all pixels in the first column of the grid.

Figure 5.6 presents the eight acquired signals, one for each sweep, of a pixel in the first column and below is the resulting averaged signal. This representation vouches for the effectiveness of the signal averaging technique for increasing the SNR as the peak is barely noticeable on all the acquired signals but stands out immensely in the averaged signal.

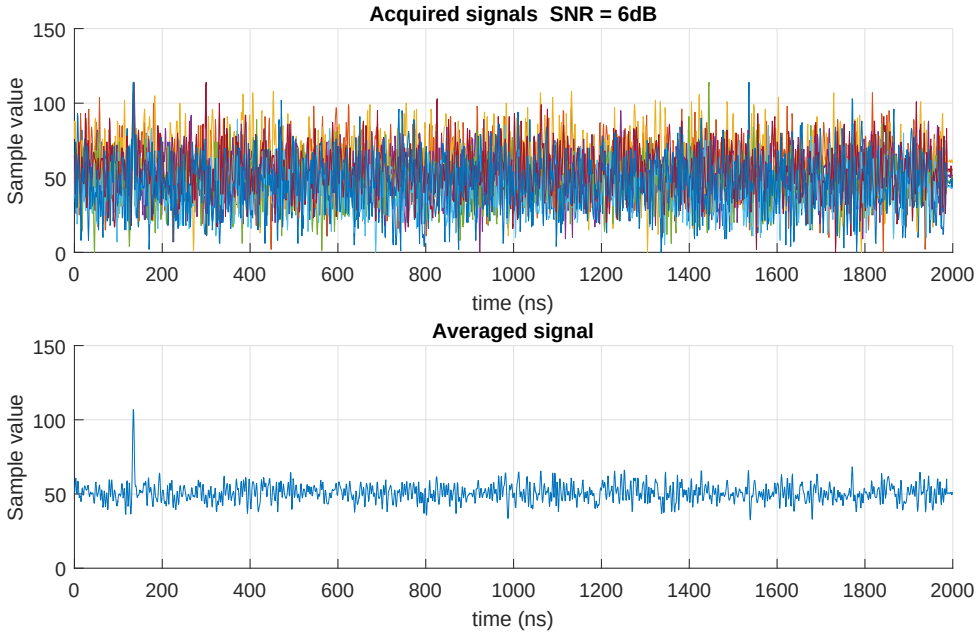

Figure 5.6: Eight acquired signals with an object distanced 20 m and corresponding averaged signal.

Following the signal averaging and memory mapping analysis, a CA-CFAR algorithm was developed to perform peak detection on the resulting signals and allow for a point-cloud representation. This algorithm was performed under the following parameters.

1. G represents the number of guard cells
2. T represents the number of training cells
3. Each sample in a signal is considered a cell
4. If the signal under evaluation has N samples, with position indexes $1,2,...N$, the algorithm analyzes only the samples from position $(\frac{T+G}{2})+1$ to $N-(\frac{T+G}{2})-1$

The CA-CFAR algorithm compares the CUT with two thresholds and if the CUT is above both of them it is considered a peak. The first threshold is set a certain amount above the average of all the training cells. This threshold is represented by the variable *noise threshold* since it aims to evaluate the average value of the background noise in the signal and ensure that the CUT is clearly

above that value. The second threshold is the maximum value of all the guard cells, which shall be called *guard cells threshold*. The CUT must be above this maximum in order to be considered a peak. From these definitions, considering sample x_k to be the CUT, belonging to a generic ADC 'X' and from any column and sweep, the threshold values can be calculated as follows.

$$noise\ threshold = 1.5 \times \frac{1}{T} \times \left(\sum_{i=k-(\frac{T+G}{2})}^{k-\frac{G}{2}-1} x_i + \sum_{i=k+\frac{G}{2}+1}^{k+\frac{T+G}{2}} x_i \right) \tag{5.1}$$

$$guard\ cells\ threshold = max\{x_i\}\ , \forall i \in [k-\frac{G}{2}; k-1] \cup [k+1; k+\frac{G}{2}] \tag{5.2}$$

If the sample value of x_k is greater than both the values given by (5.1) and (5.2), x_k is considered a peak. In the case of the performed simulation, each sample set representing a signal has 2000 samples, meaning that $N = 2000$, the number of guard cells was set to $G = 10$ given that the duration of the emitted light pulse is 10 ns and therefore represented by 10 consecutive samples of an ADC sampling at a frequency of 1 GHz and $T = 50$ for being empirically validated as a good enough value to estimate the background noise.

This peak detection algorithm was applied to the averaged signal read from memory and the peak detections were marked with a red asterisk as shown in Figure 5.7.

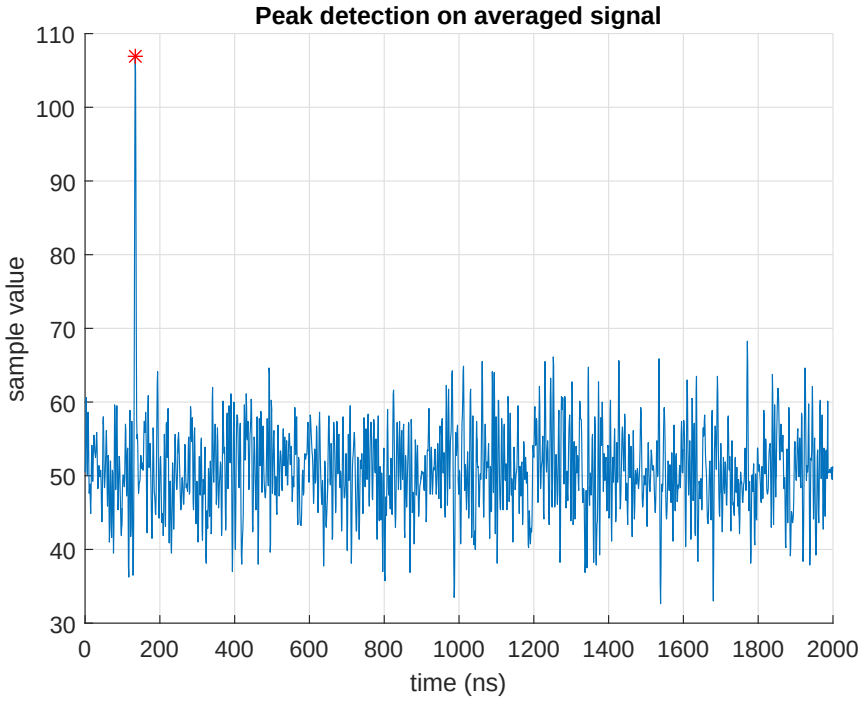

Figure 5.7: CA-CFAR algorithm applied on averaged signal.

As observed, the algorithm precisely detects the peak corresponding to the light pulse reflection, whereas the same algorithm could never be applied to the raw acquired signal because countless invalid peak detections would originate, as demonstrated in Figure 5.8.

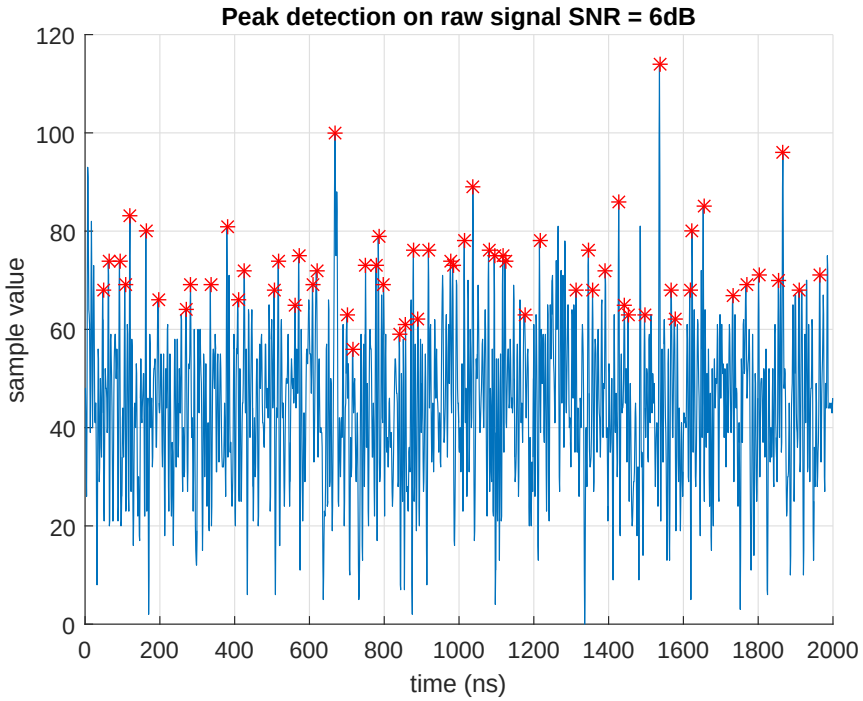

Figure 5.8: CA-CFAR peak detection algorithm applied on acquired signal.

From here, the first memory bank was addressed, reading the pixel data from all pixels present in the first line and 20 columns of the grid. Since the averaging process is finished, each one of these signals shows a peak corresponding to an object detection at the previously specified distances (20 m, 40 m, ... 200 m, 200 m, 180 m, ... 20 m). The described CA-CFAR peak detection algorithm was then applied to each one of these signals. Figure 5.9 shows the plot of the averaged signals read from the memory bank, corresponding to the first 10 columns, and the respective results from the peak detection algorithm. The signals from the last 10 columns (columns 11 to 20) were neglected from this plot since they all show peaks corresponding to objects at the same distance as the previous 10 columns and overlap with these signals. Figure 5.9 illustrates how the algorithm managed to accurately detect the peak on each signal. Furthermore, it shows that all peaks are equally spaced between them and appearing around the expected time marks of 133.33 ns, 266.67 ns, ... 1.333 μs, which correspond to the previously specified distances from the objects.

The time instants were stored at which the peaks from all 20 averaged signals were detected. Combining these ToF values with the known speed of light resulted in the distance at which the object was detected for each of these signals. Using the fact that the memory position where a signal was stored indicates its position in the grid, a visual representation of the frame can be developed. A 3D plot of the distance at which an object was detected, along with the grid line and column where that signal was measured, was constructed as shown in Figure 5.10. This point-cloud representation is in fact the one indirectly created when generating the simulated acquired

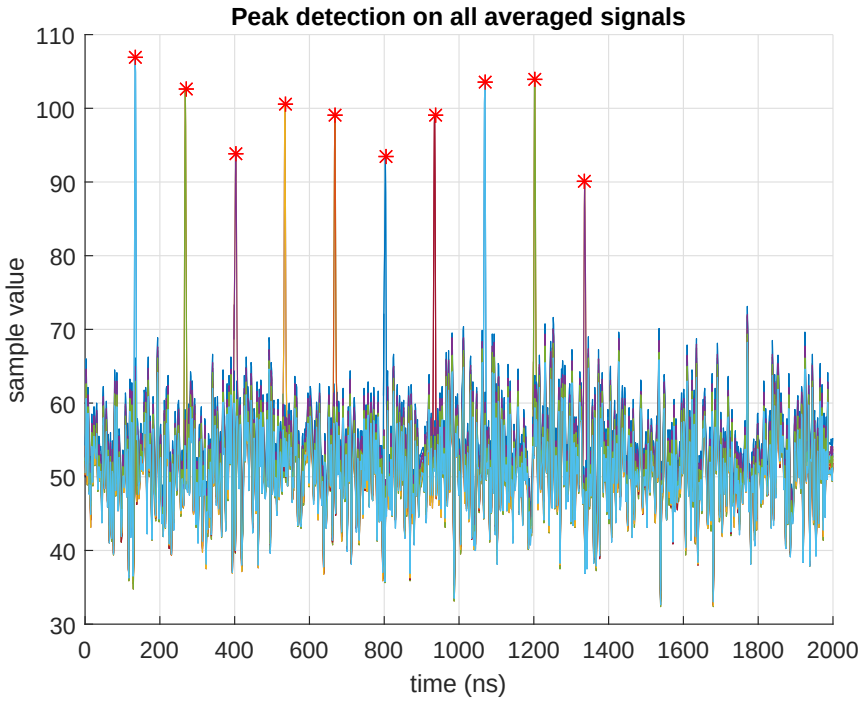

Figure 5.9: Resulting averaged signals from columns 1 to 10 and respective peak detection results.

signals. On each column an object is detected at the same distance for all pixels and then that distance increases by 20 m for 10 columns and decreases by 20 m for each of the last 10 columns.

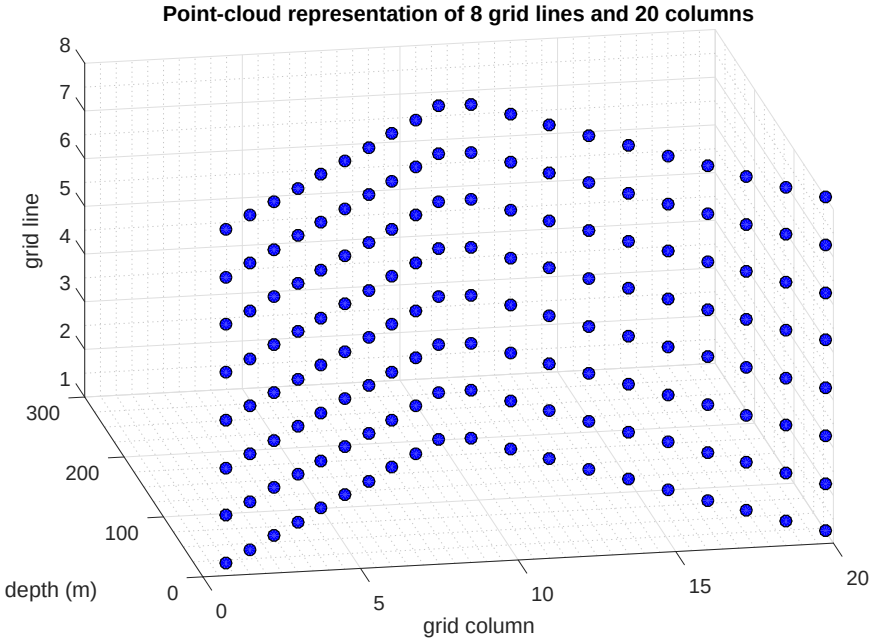

Figure 5.10: Point-cloud representation obtained for the 8×20 portion of the grid.

Extrapolating from the point-cloud in Figure 5.10, assuming that the object placement pattern repeats itself for all other 8×20 portions of the grid, the point-cloud representation after processing the entire grid is observed in Figure 5.11.

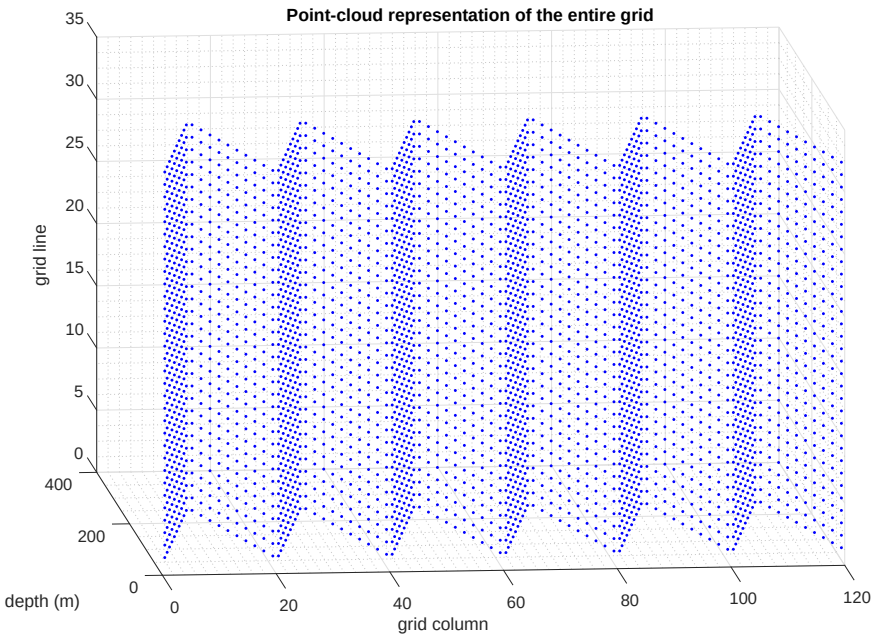

Figure 5.11: Point-cloud representation of the entire grid.

This is the final point-cloud representation, created after receiving from the sensor the entire 32×120 grid data eight times, one for each sweep, processing and storing it in the microcontroller. All the signals acquired from the entire frame in front of the sensor are processed and stored at a rate of 32.552 FPS for this simulated system. A following digital system can apply the peak detection algorithm, as exemplified in this chapter, to build a point-cloud representation like the one seen in Figure 5.11, which could be used by the navigation and decision making systems of the vehicle.

Chapter 6

Conclusion

This last chapter addresses the relevant conclusions drawn from developing this project. Section 6.1 presents the concluding statements regarding the achieved design of the signal processing system and its comparison with the developed simulation. In section 6.2 an analysis is performed on other potential algorithms to be implemented in a similar LiDAR data processing solution and how the averager system could be expanded to constitute a full LiDAR data processing system, capable of producing a point-cloud representation.

6.1 Developed System

The first and most relevant conclusion is that it is possible to create a system for processing LiDAR data, which obeys to the imposed restrictions, making it an implementable and cost-effective system. Specifically, the designed system uses only four RIF components to interface with the LiDAR sensor, accepting practical data transfer rates of up to 1 Gbps on each LVDS lane, offering a total data output rate to the sensor of 16 Gbps. The system is always clocked at a frequency of 250 MHz or lower, specified by the parameters under which the system can operate (section 4.1). With 1 Gbps LVDS speed, the developed system is capable of outputting frames at 86 FPS for a small grid with 16 lines $\times$120 columns $\times$1500 8-bit samples per pixel, whereas for a much bigger grid with 32 lines $\times$120 columns $\times$2500 12-bit samples per pixel it still achieves a 17 FPS rate (analysis in annex D). This trade-off between performance and data amount was well evaluated and the user can define the operating point of the system accordingly. Furthermore, only 16 (4 MB) of the 24 (6 MB) SRAM banks allowed for the system were utilized and the system never exceeds the maximum allowed memory accessing frequency of 250 MHz. In fact, following the analysis done in chapter 4, table 6.1 show exactly at what frequency the system is clocked and at what rate each memory bank is addressed, depending on the input bits per sample and the LVDS speed in consideration.

The interfacing requirements evaluated in section 3.1.3 were met with a system capable of interfacing with the LiDAR sensor in a realistic way.

Bits per Sample	LVDS speed (Mbps)	RIF Output Rate (MSPS)	f_{clk} (MHz)	Memory Addressing Frequency – Mode 1 (MHz)	Memory Addressing Frequency – Mode 2 (MHz)
8	1000	500	250	125	62.5
	800	400	200	100	50
	600	300	150	75	37.5
	400	200	100	50	25
10	1000	400	200	100	50
	800	320	160	80	40
	600	240	120	60	30
	400	160	80	40	20
12	1000	333.33	166.67	83.33	41.67
	800	266.67	133.33	66.67	33.33
	600	200	100	50	25
	400	133.33	66.67	33.33	16.67

Table 6.1: Clock signal frequency and memory addressing frequencies of the developed system depending on its operating point.

The simulation results showed an output rate of 32.552 FPS on a grid with 32 lines $\times$120 columns $\times$2000 8-bit samples per pixel, exactly as predicted by the analysis in chapter 3 (see table 3.1) and the performance analysis developed in section 4.2.2 (see Figure 4.5). This system operated at a frequency of 250 MHz and with 37.5 % of the allocated memory for the system unused, as anticipated in the design stage of the system. According to the simulation, the memory mapping works as defined in the conceptualization of the averager system and does so in a comprehensible way so that any further system can easily continue the data processing until a point-cloud representation is reached. Still in the simulation described in chapter 5, it was demonstrated that the developed system correctly performed signal averaging and stored it in memory as intended. Furthermore, a possible peak detection algorithm was implemented that correctly identified the peaks in each signal and those detections were then utilized to create a point-cloud representation. This representation was extremely accurate, which strongly suggests that the averager system can work with a peak detection digital system like the simulated one, resulting in a complete system that generates accurate point-cloud representations from the LiDAR data.

6.2 Future Work

The developed system was designed in a way that would allow for further expansion so that it is possible to conceptualize and design a system that produces a point-cloud from the work developed in this thesis.

The following systems could interface with the averager either by reading the samples from the SRAM, synchronized with the memory manager so that each bank would never be addressed

simultaneously by two different subsystems, or even directly connecting to the AVG subsystem output, using the averaged data directly before storing it. Even though the performance of the averager system was analyzed, the performance of the overall system (with peak detection) would need to be assessed as well.

Another important aspect in the development of any system that connects to the averager is the fact that 8 memory banks are still available for use, to complete the point-cloud creation.

A match filtering algorithm is a possibility to further increase the SNR of the signals and maximize the probability of every light pulse reflection being detected. Applying a match filter after signal averaging would increase even more the SNR by reducing the noise amplitude. This algorithm would have to store the Fourier transform of the samples corresponding to the emitted light pulse and the acquired signals would have to be converted to the frequency domain to apply the filter. This process would attenuate the effect of any set of samples that did not match with the Fourier transform of the emitted light pulse.

However, the most evident expansion on the proposed system is to define and detail at a hardware level a peak detection algorithm that interfaces correctly with the averager system. This was analyzed in greater detail for the CFAR peak detection algorithm as described in section 6.2.1.

6.2.1 CFAR System

A solution could be explored with a subsystem that performs peak detection and connects to the output of the AVG subsystem, as illustrated in Figure 6.1.

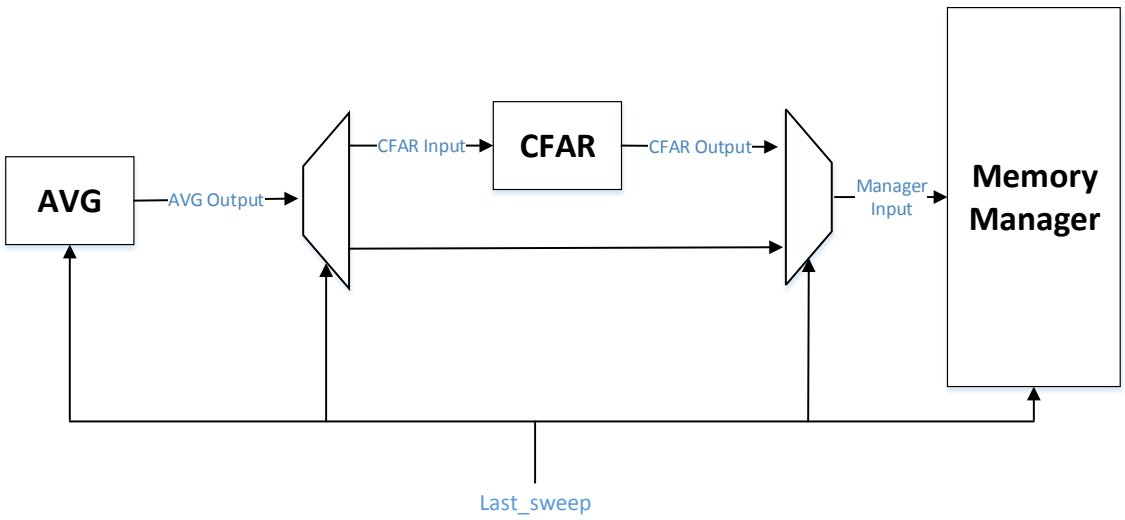

Figure 6.1: CFAR system block diagram.

When the *Last_sweep* signal would indicate that the added signal had to be divided to conclude the signal averaging process, the data output from the AVG block would go into the CFAR subsystem. The CFAR and memory manager would then work synchronously to store the result of the peak detection in a way that would allow reading the point-cloud data directly from the SRAM.

To better exemplify the analysis done on this CFAR subsystem, let us consider that the whole system is operating in sensor connection mode 1, performing averaging by 4 on a grid with 2000 samples per pixel and, for simplification purposes, that the CFAR algorithm uses 2 guard cells and 4 training cells only. Additionally, each sample is treated as a cell in the CFAR algorithm.

In Figure 6.2 is exemplified how the algorithm would sequentially organize and interpret the samples from the signal on each pixel to apply the CFAR algorithm simultaneously on all of them.

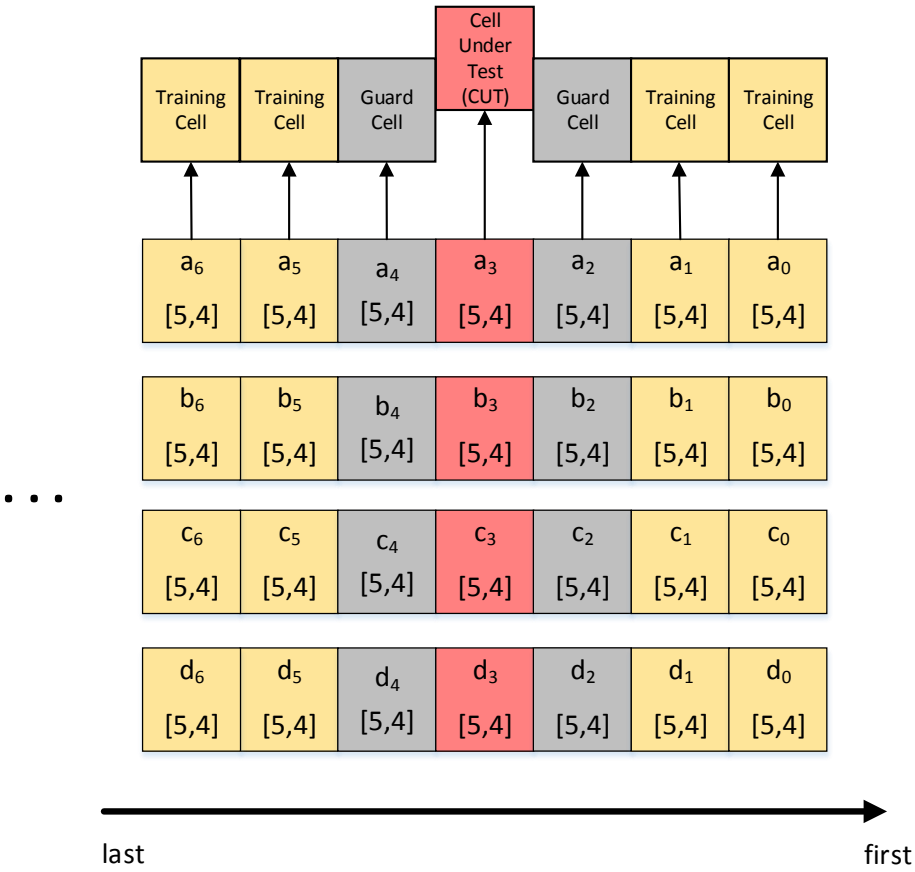

Figure 6.2: CFAR cell evaluation on LiDAR data.

As the samples from the last sweep would output from the AVG subsystem, the CFAR block would apply the algorithm in a window of 7 cells in a pipeline fashion. The threshold setting and comparison would occur on a hardware level as illustrated in Figure 6.3.

As exemplified in Figure 6.3, every time a new sample from the set $d_0[5,4] - d_{1999}[5,4]$ would enter the CFAR subsystem, a new threshold value would be set and the CUT compared to that threshold. The output would consist in a single bit value, indicating if the CUT would be above or below the threshold, with a logic value 1 or 0 respectively.

Finally, as depicted in Figure 6.4, for each CUT analysis, a 1 or 0 logic value would originate, indicating a peak detection or not.

Each memory bank address could store 64 peak detection results (64 bits) so the remaining 8 unused memory banks could store a total of $256000 \times 8 = 2.048 \times 10^6$ peak detection results.

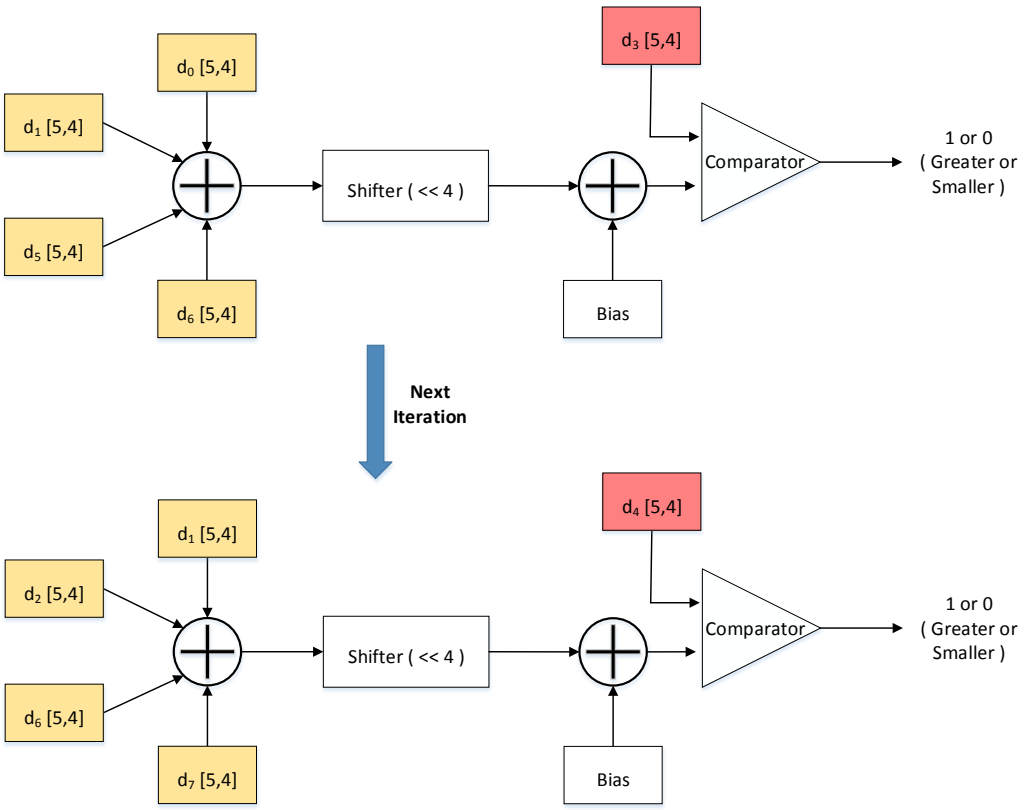

Figure 6.3: Calculating the CFAR threshold and outputting the peak detection result.

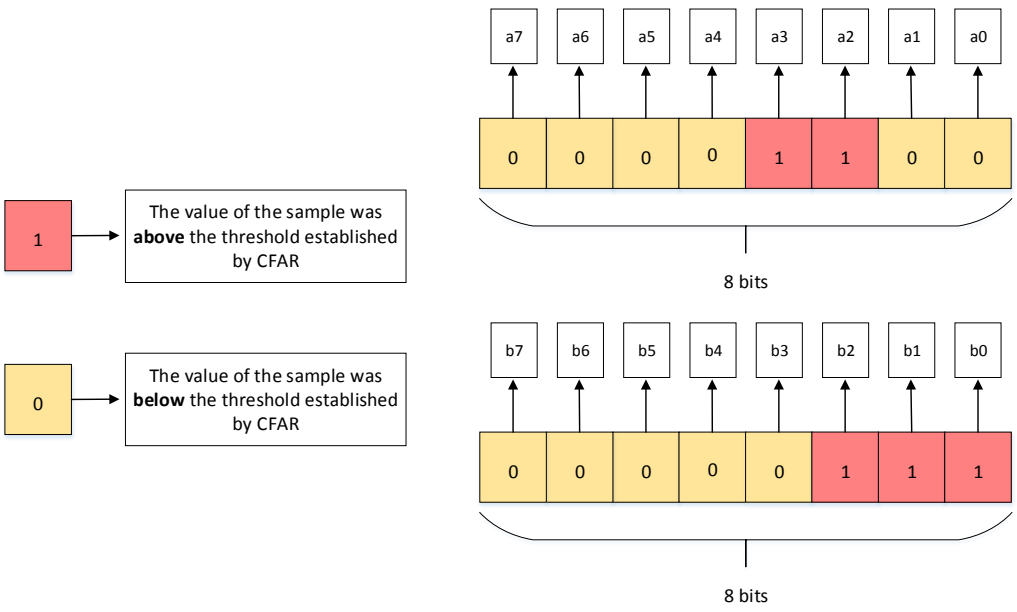

Figure 6.4: Output representation of the CFAR subsystem.

Appendix A

Proposed System Overview

In this appendix are included some results of the analysis that led to the general proposed system, presented in section 3.6, that were not included in chapter 3.

A.1 Amount of Data per ADC

Figure A.1 resulted from analyzing how much data an ADC would generate per light pulse emission, which indicates how much overall data derives from the sensor. This amount depends on the number of bits per sample each ADC generates, the sampling frequency at which it operates and also for how long they generate data every time a light pulse is fired (sampling duration). Figure A.1 takes into account all of these aspects, being that on each subplot of the image is presented the generated data by an ADC for a different sampling duration value.

A.2 Buffer Output Rate

When evaluating the solution for transmitting all the sensor generated data using only one buffer, seen in Figure 3.5, a slightly deeper analysis was performed, which resulted in Figure A.2.
This plot illustrates the required read out data rate from the buffer in a way that prevents overflowing, for a PRF value of 80 kHz. This value depends on the number of bits per sample and the sampling frequency of the ADC, as included in the plot.

A.3 Sampling Duration and Time to Transfer Data

Figure A.3 illustrates how the sampling duration and the time to transfer the acquired data vary depending on the desired depth range. This plot was evaluated for a sensor operating with a PRF value of 100 kHz.

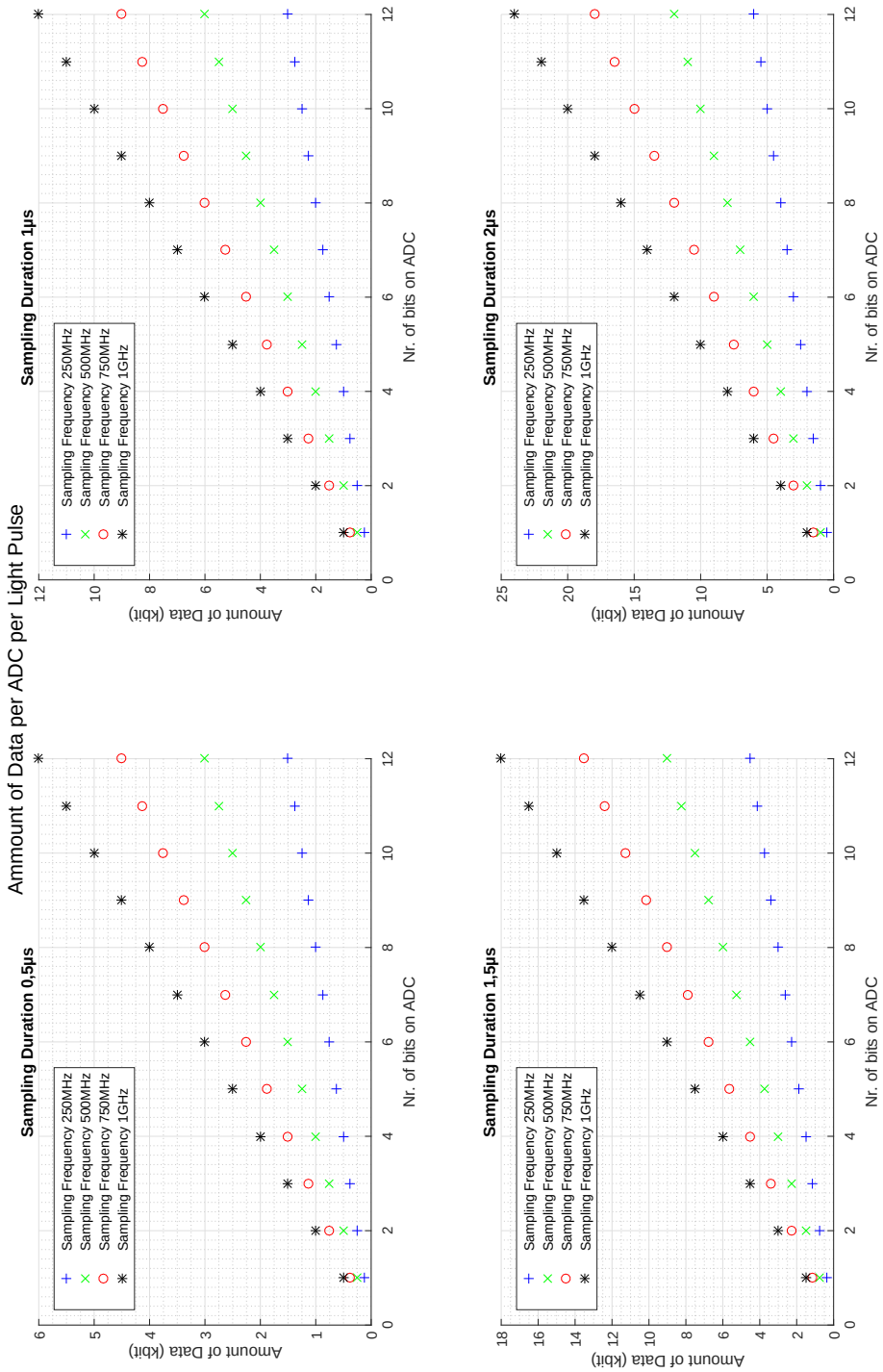

Figure A.1: Amount of data generated per ADC every light pulse emission.

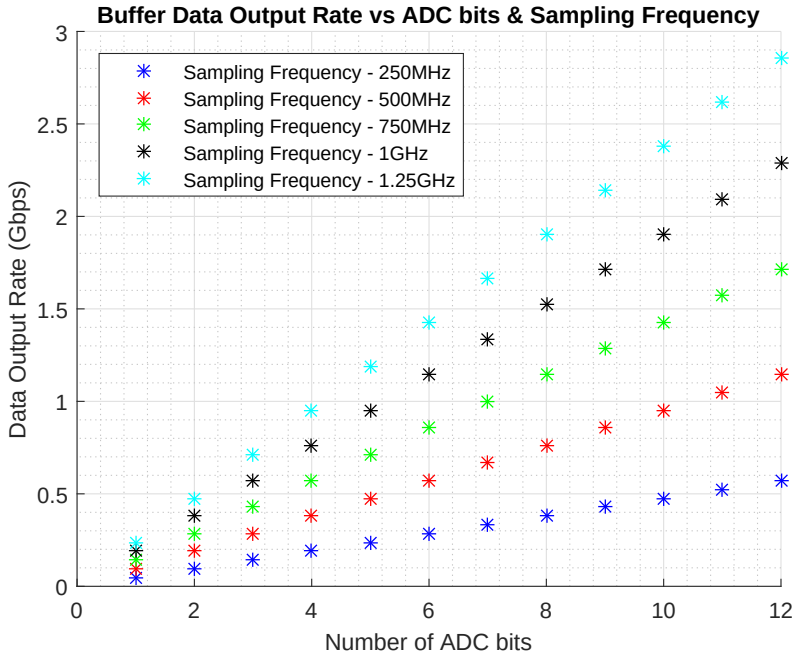

Figure A.2: Data output rate required to completely empty the buffer depending on the ADCs sampling frequency and bits per sample.

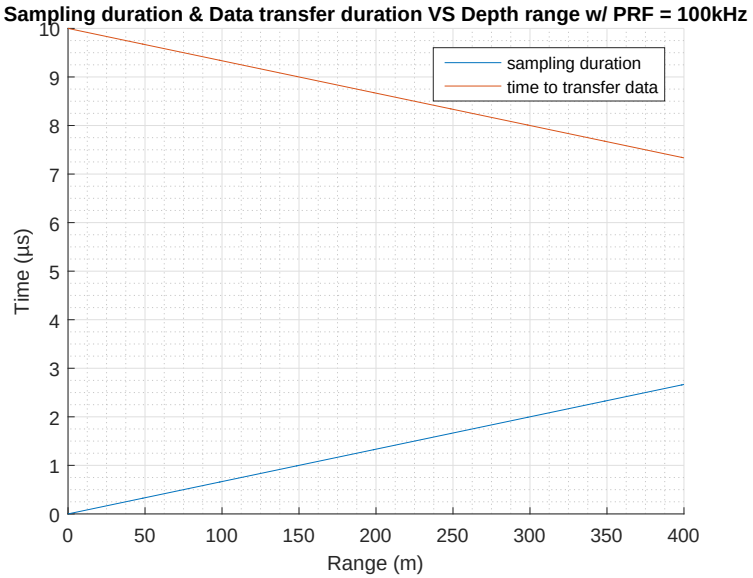

Figure A.3: Sampling duration and consequent time to transfer data for desired depth range.

A.4 Data Rate per ADC

The plot in Figure A.4 simply illustrates the amount of data an ADC generates during the sampling duration, depending on the ADC sampling frequency and for different number of bits per sample.

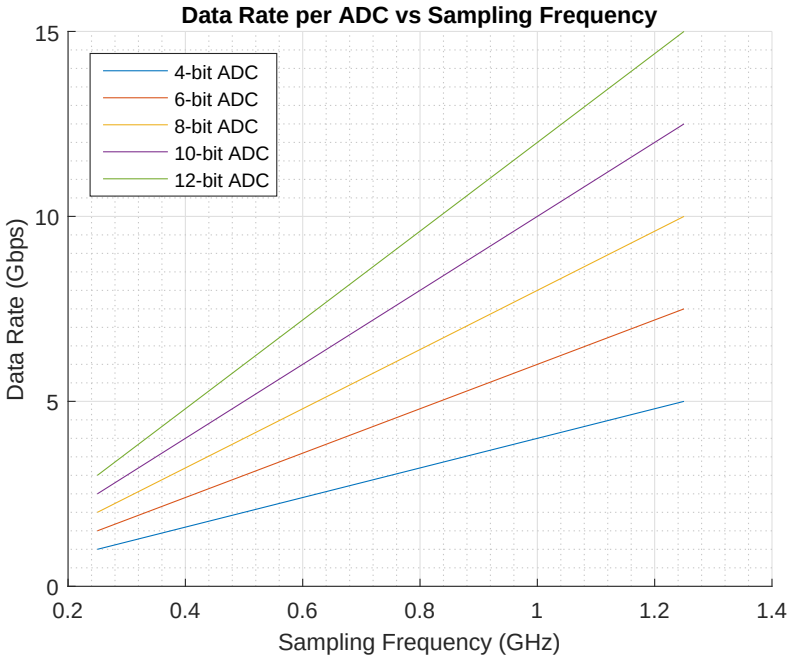

Figure A.4: Data rate an ADC generates during the sampling duration time.

A.5 Depth Resolution

The plot in Figure A.5 demonstrates how the depth resolution of the image generated by the system changes with the sampling frequency of the ADCs (see (3.4)).

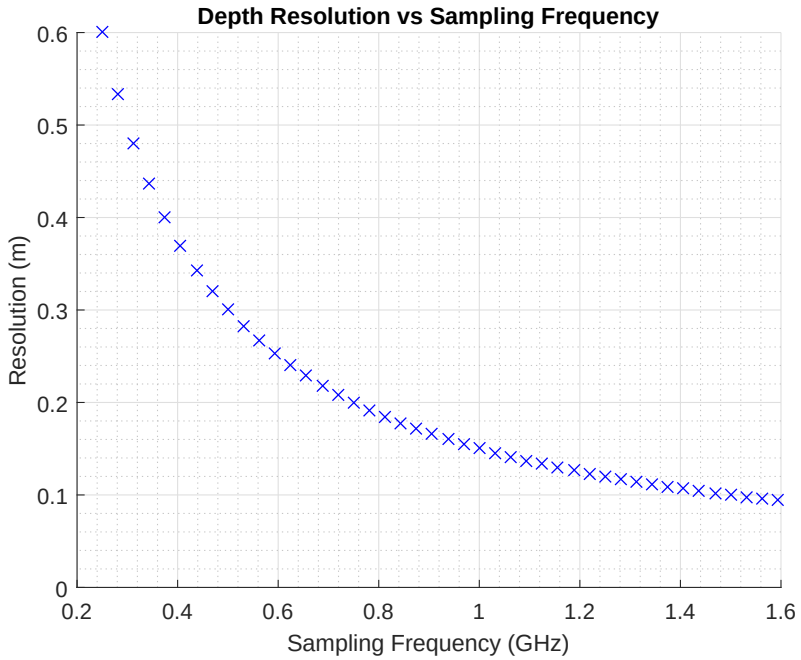

Figure A.5: Depth resolution depending on the sampling frequency of the ADCs.

Appendix B

Computation Model

This appendix has content related to the averager system described in chapter 4, which were not necessary to include throughout the text.

B.1 Full Averager Time Diagram

The time diagram in Figure B.1 illustrates the time behavior of the averager from the inputs of the AVG subsystem all the way to the connections to the memory banks. The memory manager performs the synchronization in accessing each memory bank and reading and writing samples from the AVG block, as represented.

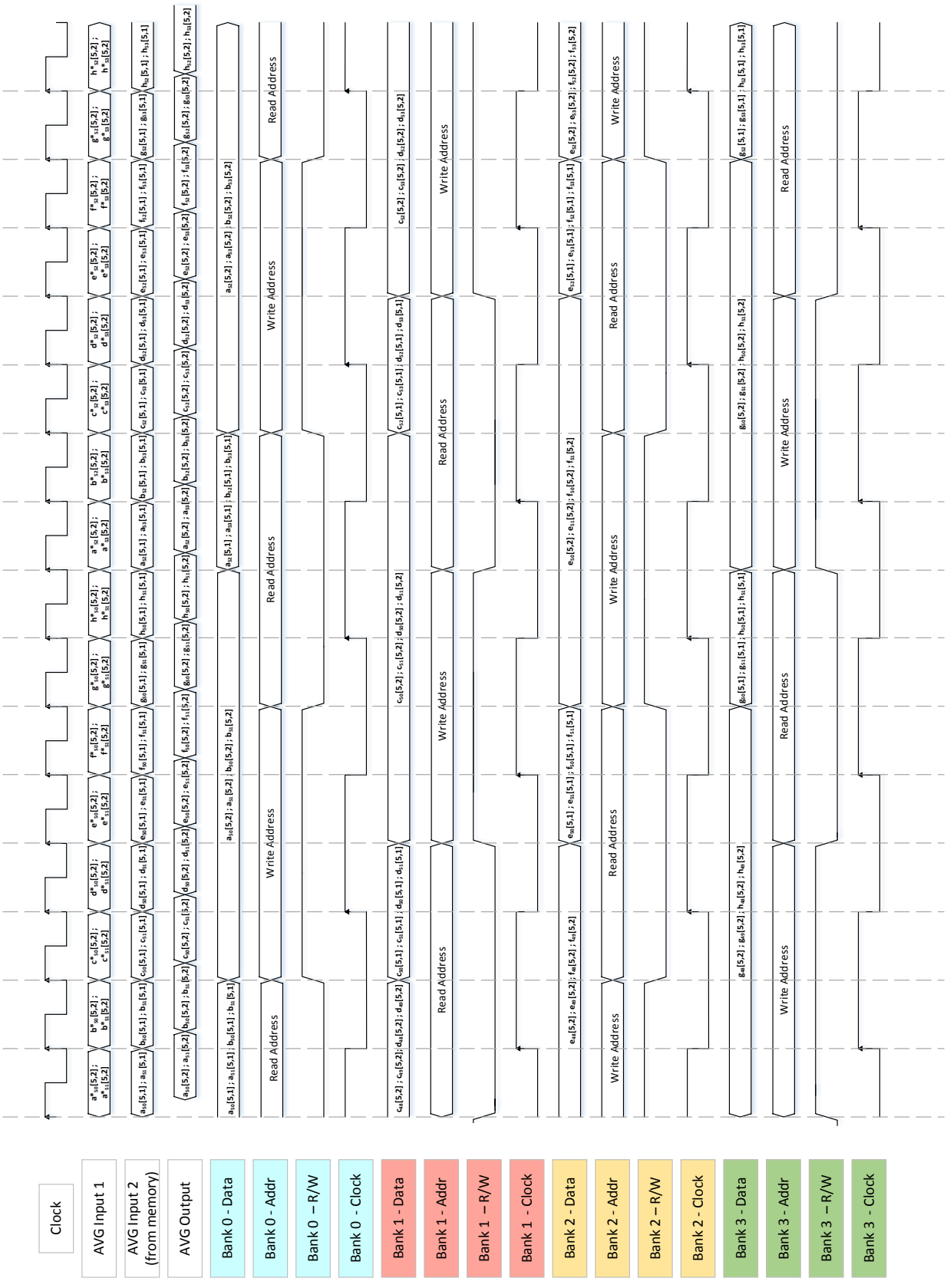

Figure B.1: Averager time diagram – averaging and memory accessing in mode 2.

Appendix C

Number of RIF Components Required for Desired FPS

The following plots show the number of RIF components necessary to achieve FPS output values in a range of 15 to 60 FPS. Each one of the images shows this correlation for a system with different LVDS data transfer rates and bits per sample generated by the ADCs, considering a grid with 32 lines $\times$120 columns $\times$2000 samples $= 7.68 \times 10^6$ total samples in a grid. The analysis that led to the creation of these images was carried out when attempting to understand the best way of using the SRIF and RIF components to interface the LiDAR sensor and the microcontroller.

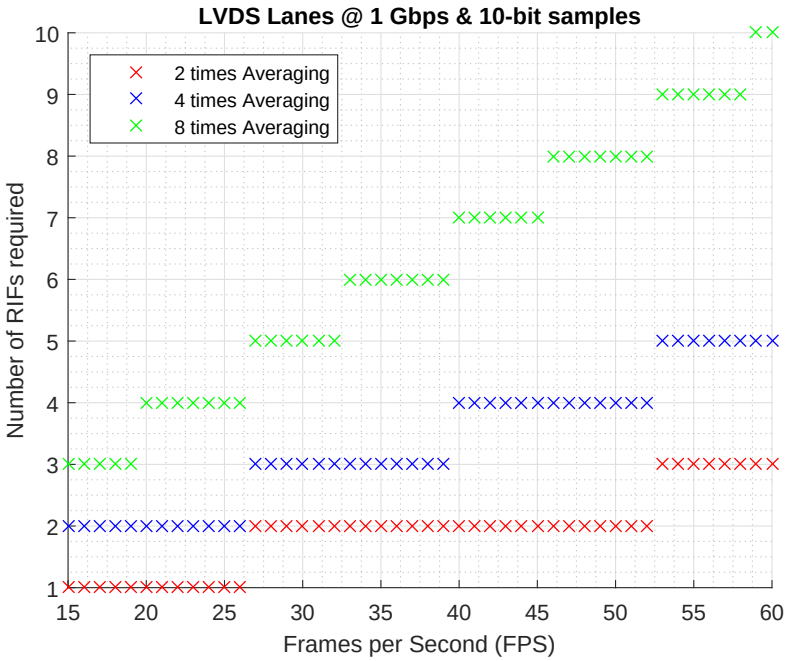

Figure C.1: Number of RIFs required to achieve different FPS output rates at 1 Gbps LVDS speed and 10 bits per sample.

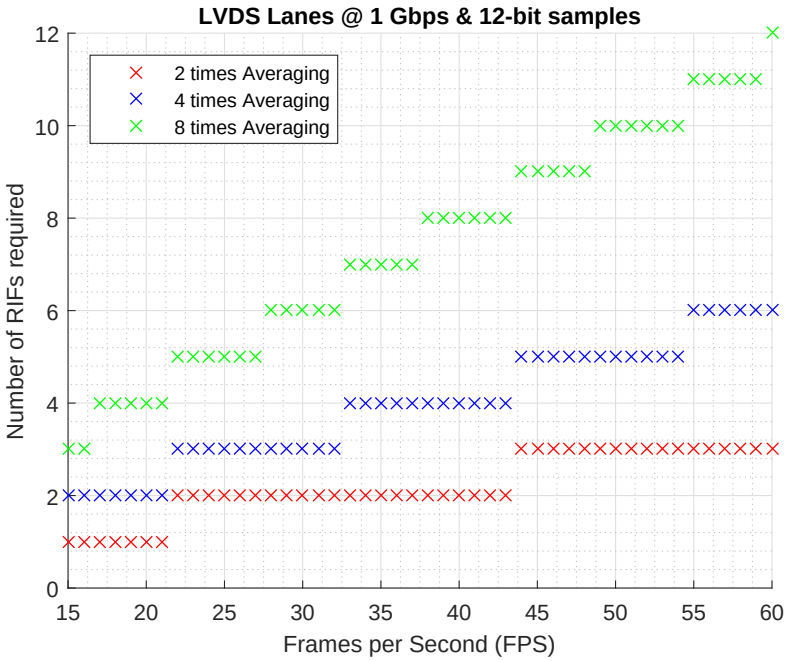

Figure C.2: Number of RIFs required to achieve different FPS output rates at 1 Gbps LVDS speed and 12 bits per sample.

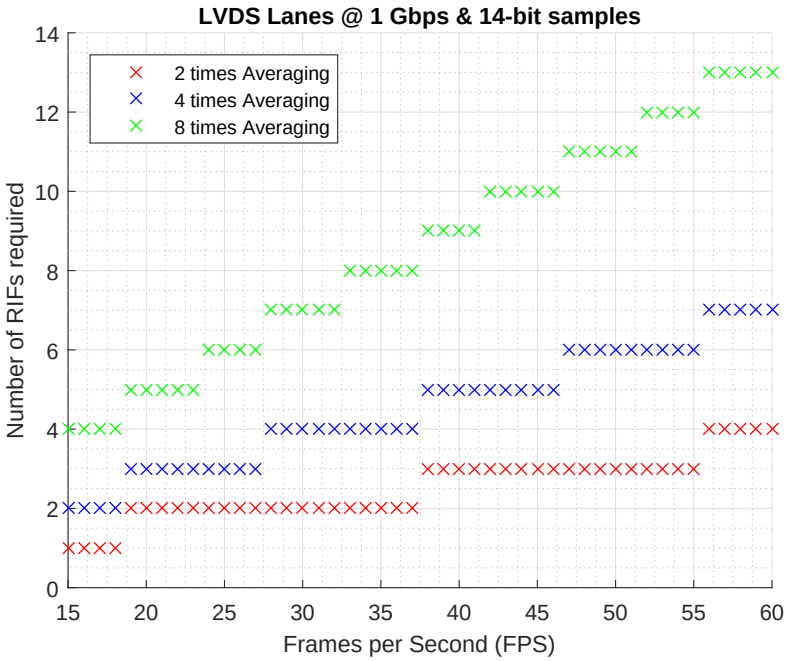

Figure C.3: Number of RIFs required to achieve different FPS output rates at 1 Gbps LVDS speed and 14 bits per sample.

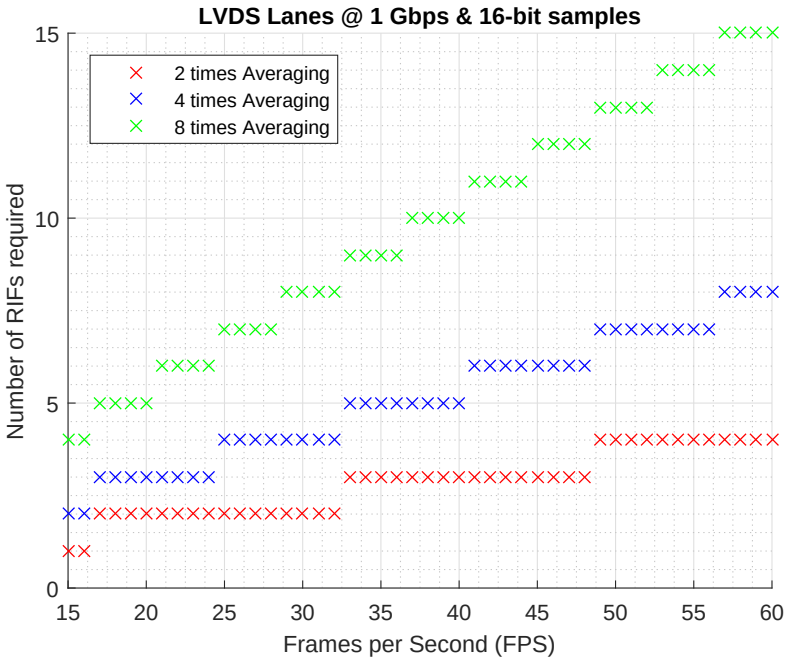

Figure C.4: Number of RIFs required to achieve different FPS output rates at 1 Gbps LVDS speed and 16 bits per sample.

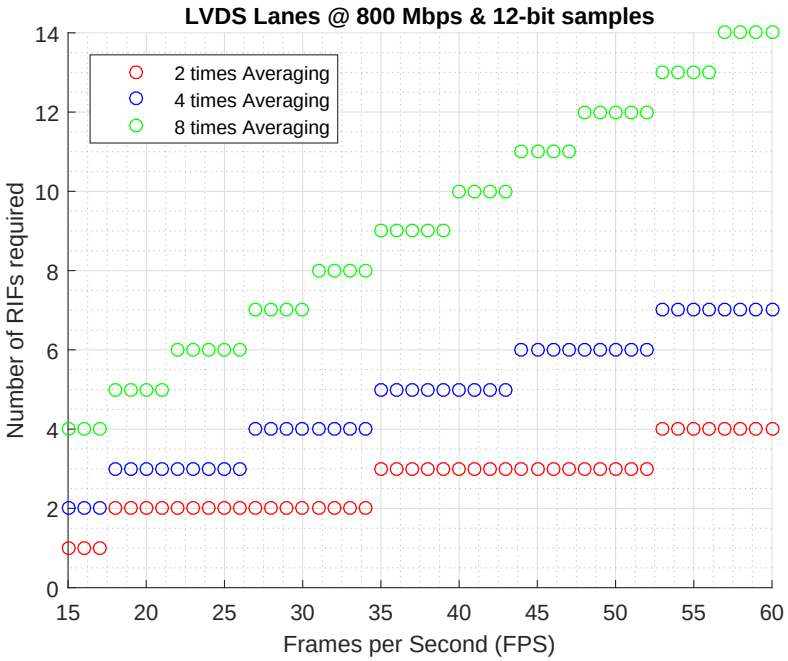

Figure C.5: Number of RIFs required to achieve different FPS output rates at 800 Mbps LVDS speed and 12 bits per sample.

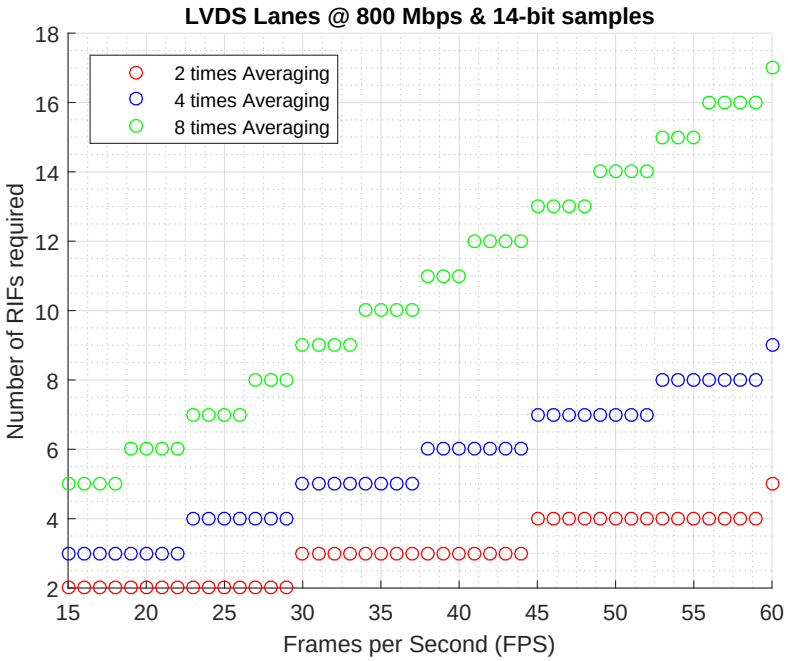

Figure C.6: Number of RIFs required to achieve different FPS output rates at 800 Mbps LVDS speed and 14 bits per sample.

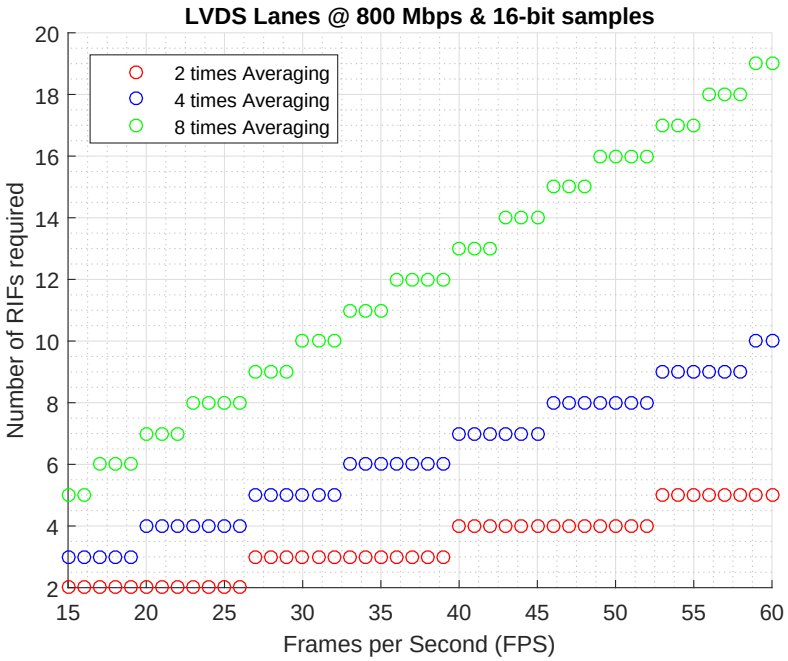

Figure C.7: Number of RIFs required to achieve different FPS output rates at 800 Mbps LVDS speed and 16 bits per sample.

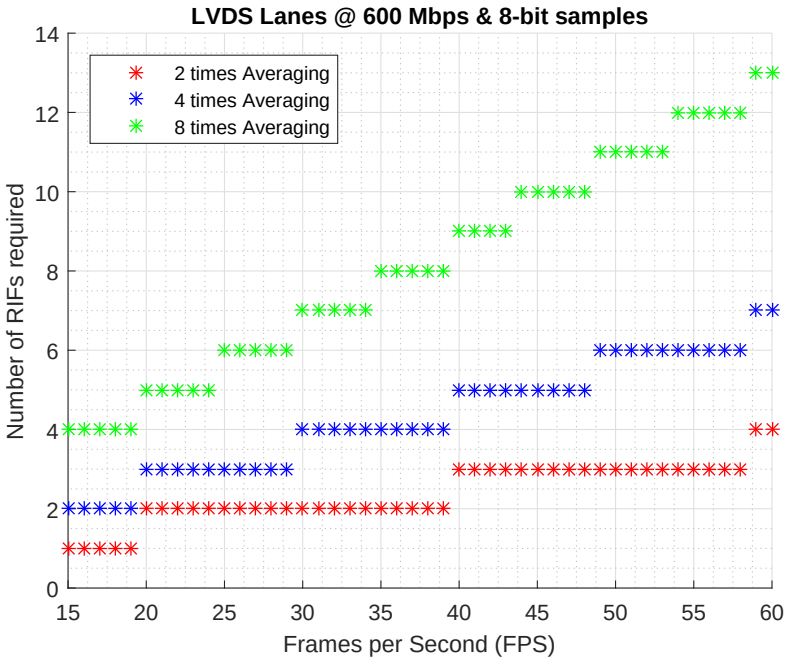

Figure C.8: Number of RIFs required to achieve different FPS output rates at 600 Mbps LVDS speed and 8 bits per sample.

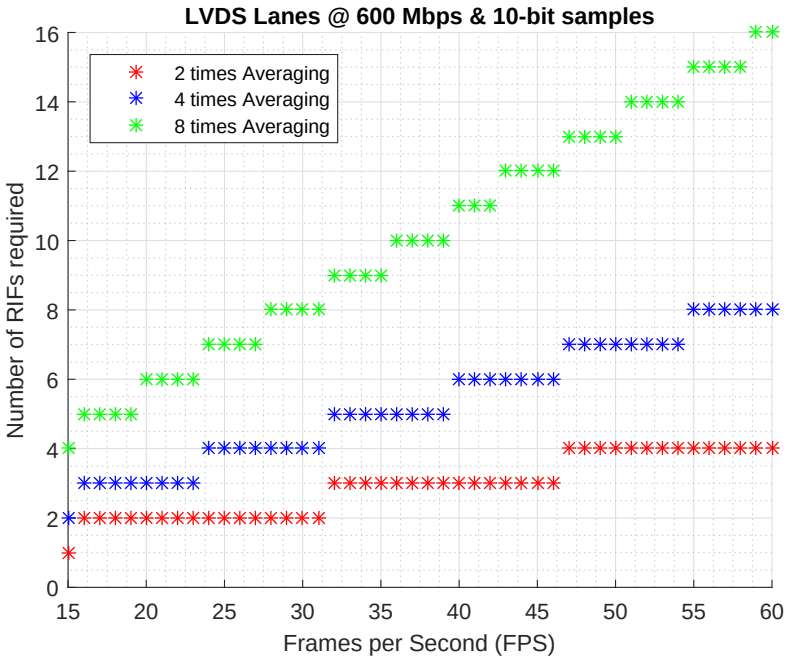

Figure C.9: Number of RIFs required to achieve different FPS output rates at 600 Mbps LVDS speed and 10 bits per sample.

Appendix D

Achievable FPS Rate Depending on Grid Size

This appendix contains some of the results originating from the performance analysis of the developed system described in section 4.2.2. All the plots were created simulating a system with a maximum of 4 RIF components for interfacing and assuming an LVDS connection speed of 1 Gbps. In the following Figures, the achievable FPS output values were plotted for different sizes of the grid (in samples per frame) and with the system performing averaging by 2, 4 and 8.

It is important to note that the best achievable FPS rate, for a grid with 7.68×10^6 and averaging by 8, is in line with the prediction of number of RIFs required to achieve a certain FPS output rate in Figures C.1 – C.4.

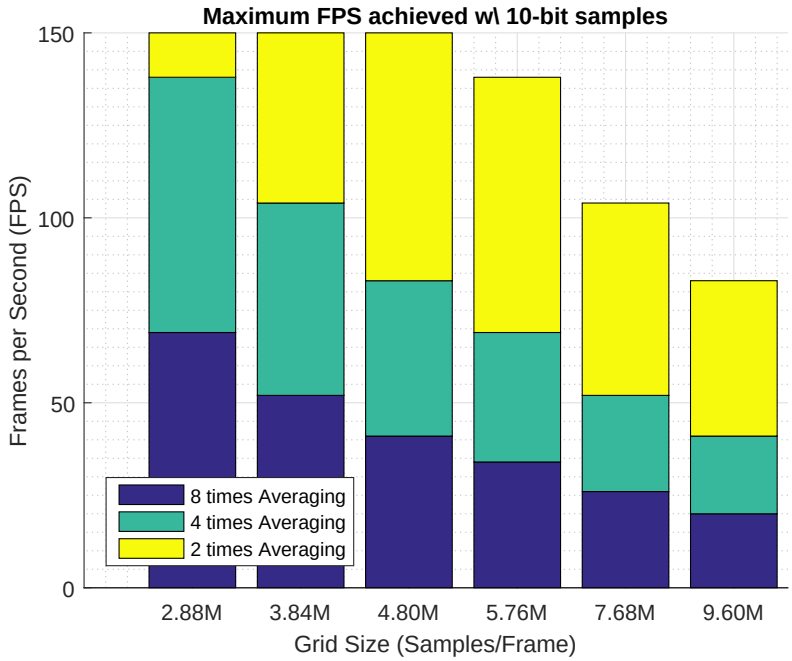

Figure D.1: Achievable FPS output rate for different grid dimensions with 10 bits per sample.

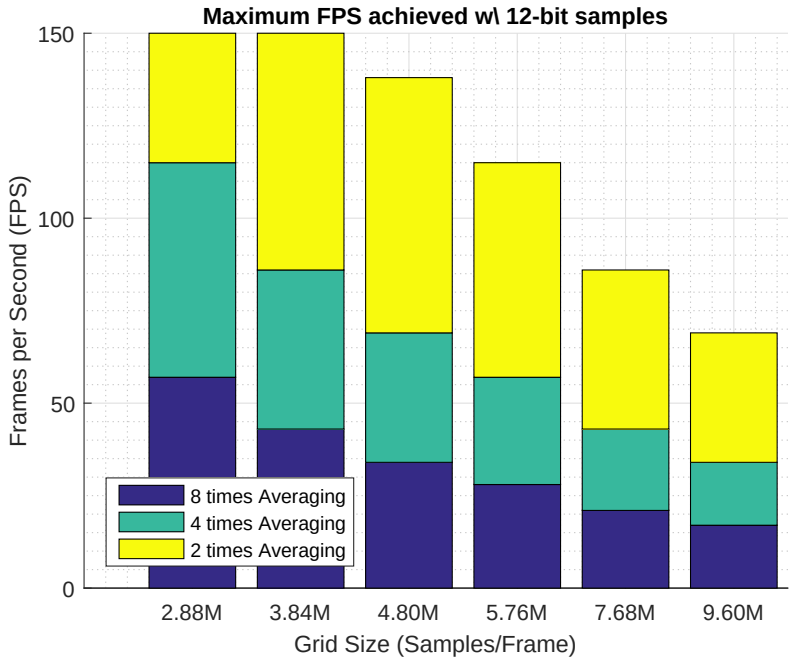

Figure D.2: Achievable FPS output rate for different grid dimensions with 12 bits per sample.

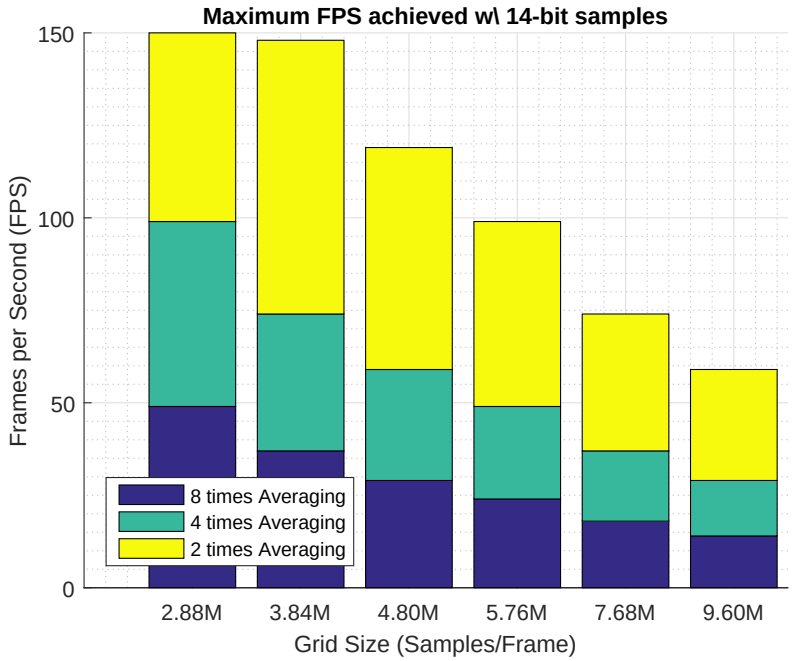

Figure D.3: Achievable FPS output rate for different grid dimensions with 14 bits per sample.

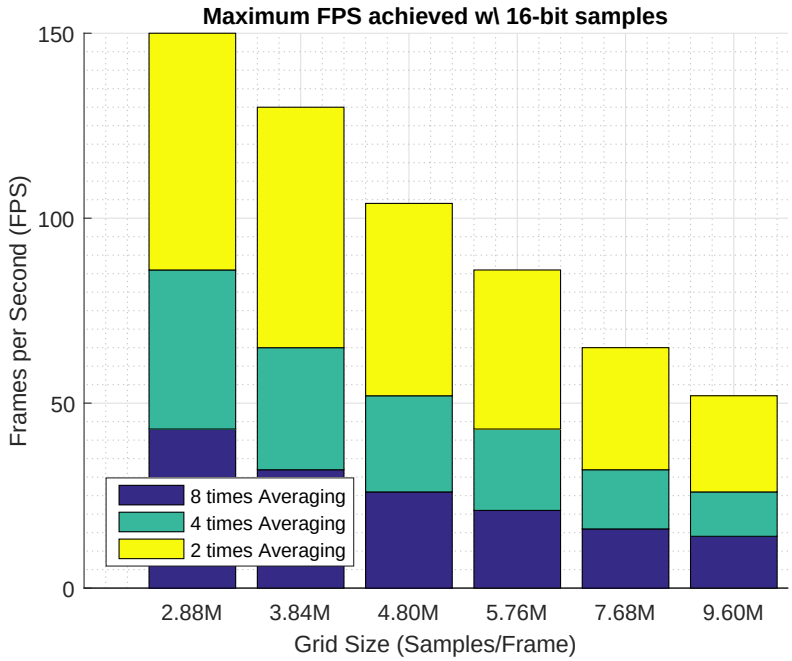

Figure D.4: Achievable FPS output rate for different grid dimensions with 10 bits per sample.

Appendix E

Signal Averaging Comparison

Throughout the analysis performed in chapter 5, many signals were generated, simulating the data sampled by an ADC, with different SNR values. The following plots consist in the result of applying signal averaging by 2, 4, 8 and 16 on each of these set of acquired signals with an SNR of -3, 0, 3, 6 and 9 dB.

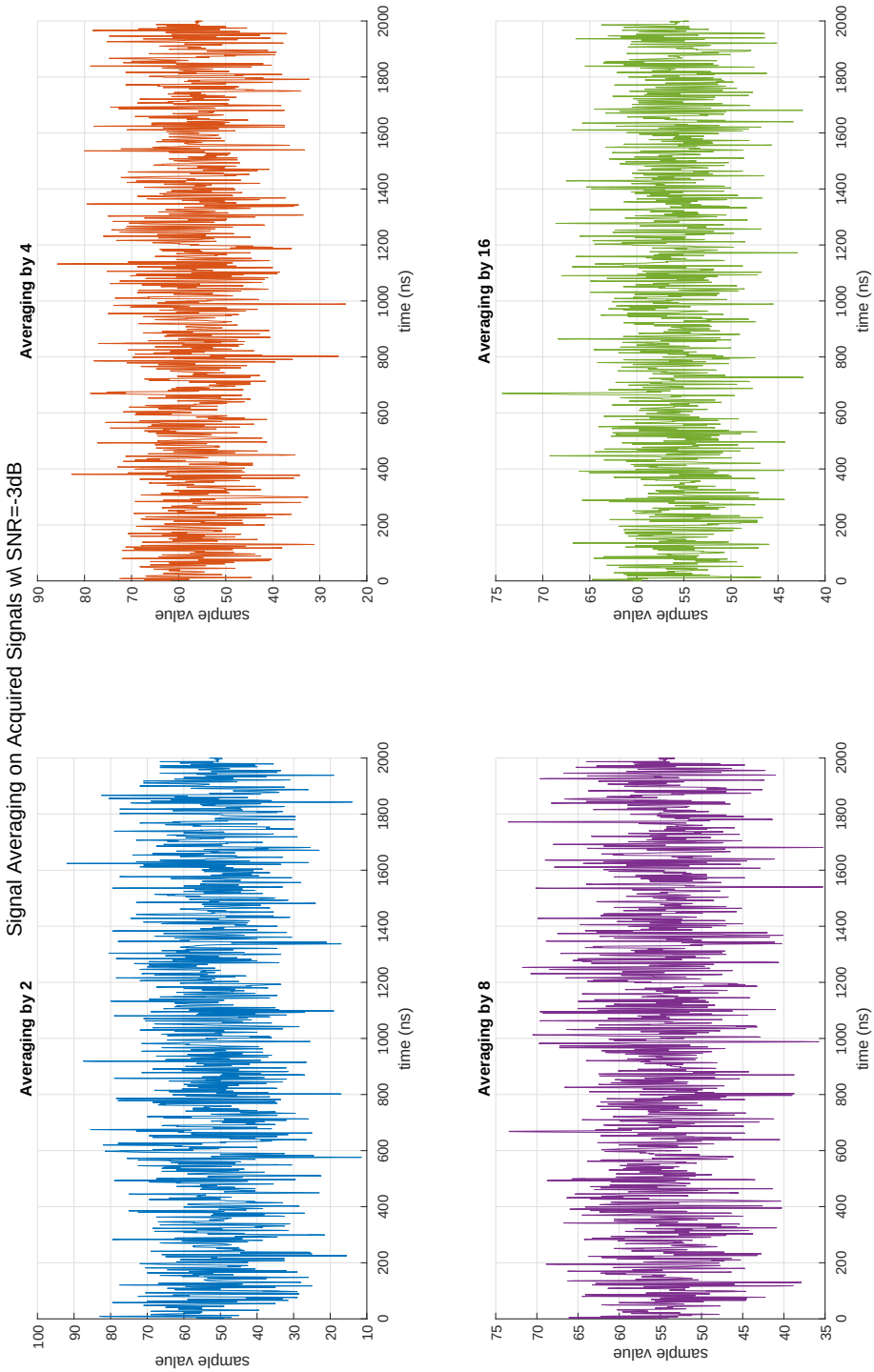

Figure E.1: Signal averaging by different amounts on acquired signals with $SNR = -3\,\mathrm{dB}$.

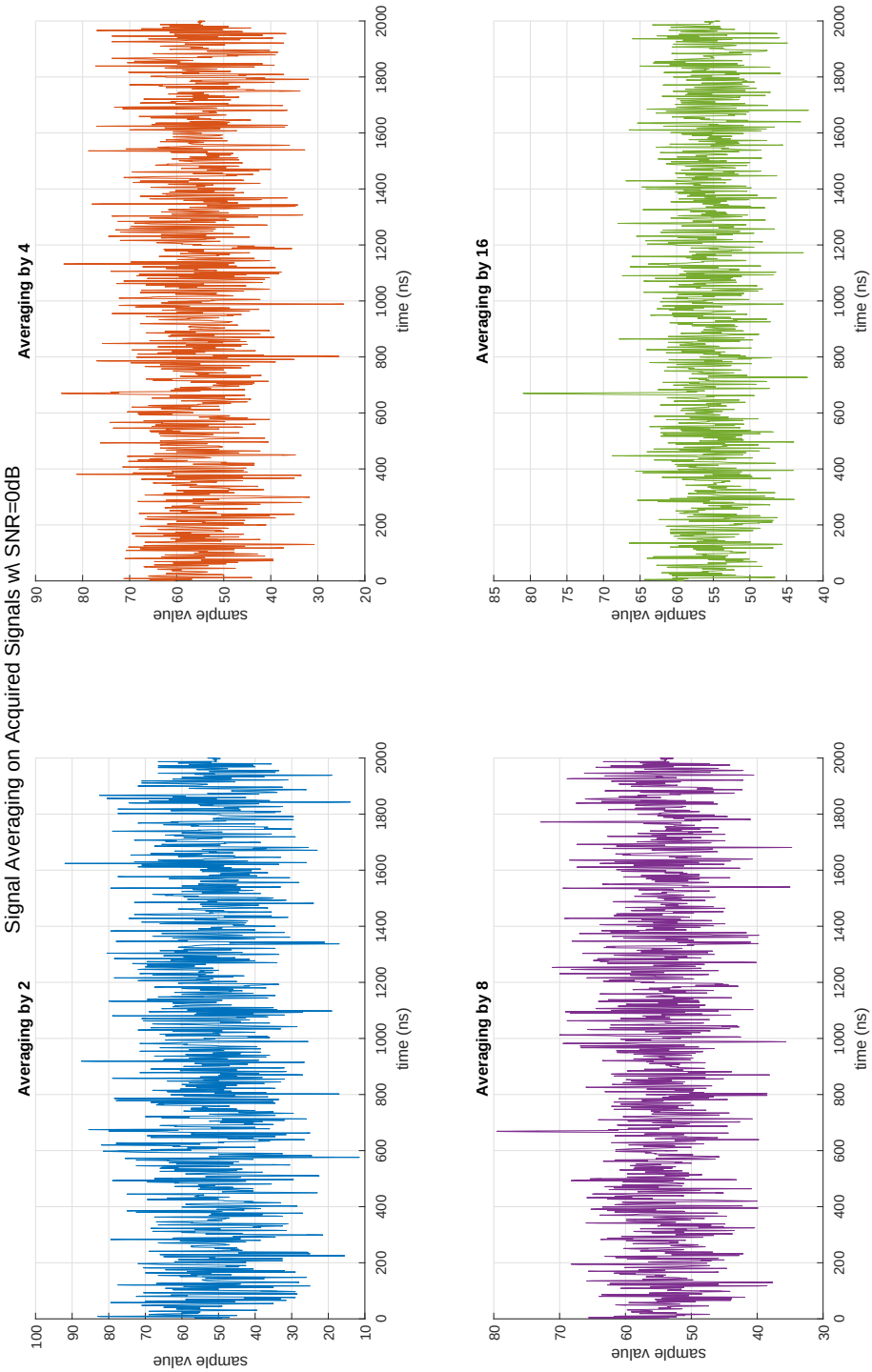

Figure E.2: Signal averaging by different amounts on acquired signals with $SNR = 0$dB.

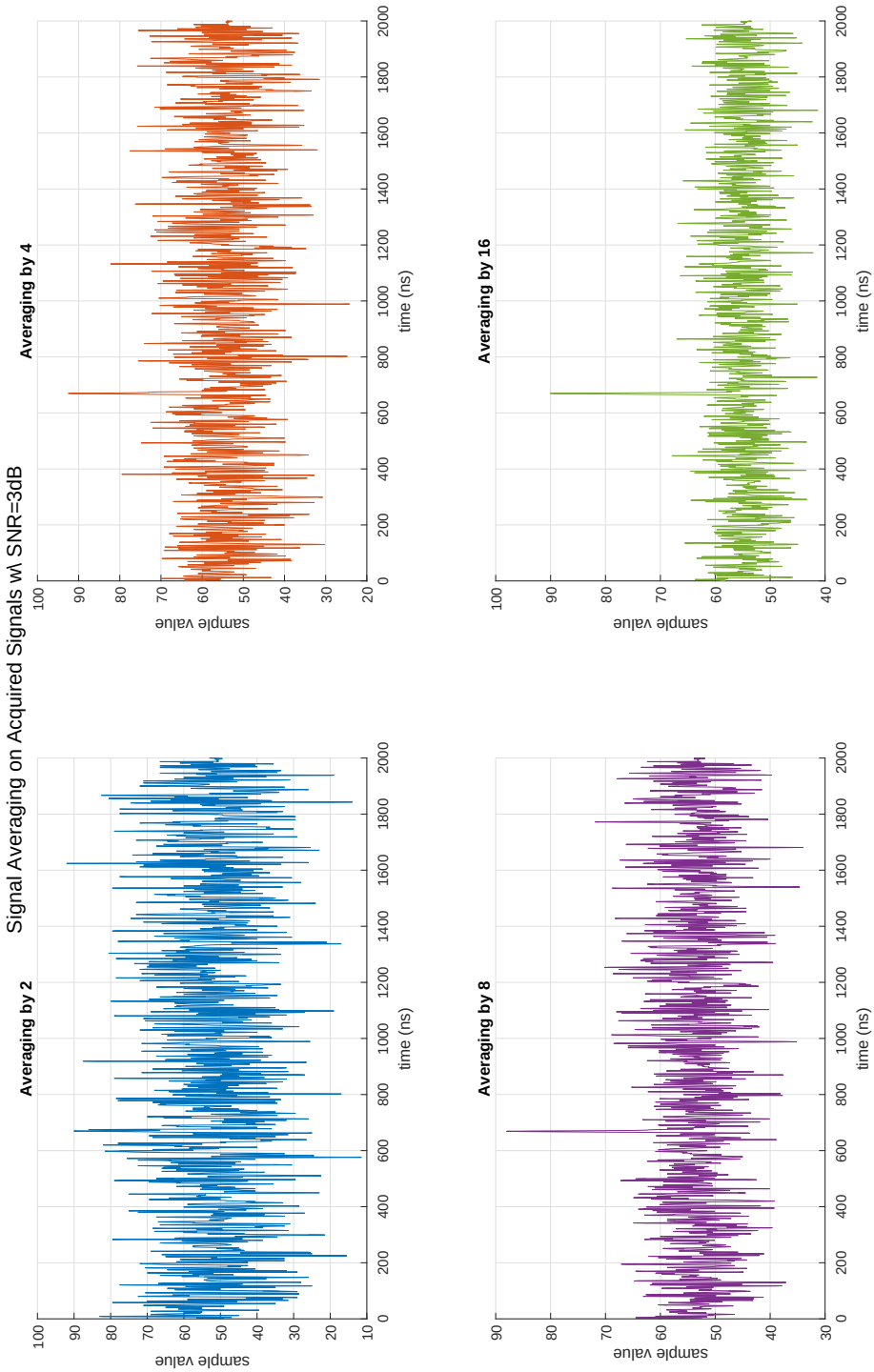

Figure E.3: Signal averaging by different amounts on acquired signals with $SNR = 3\,\text{dB}$.

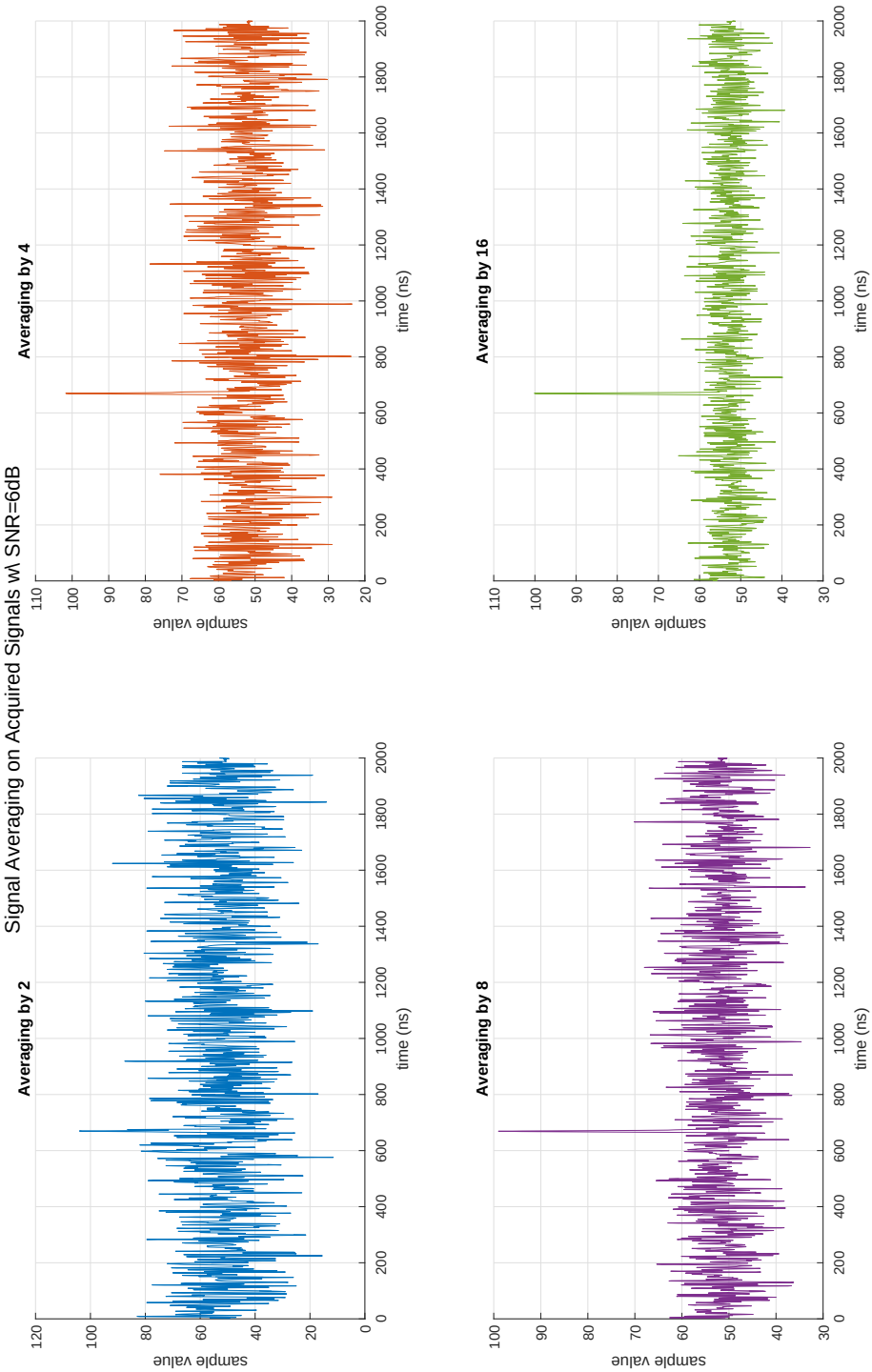

Figure E.4: Signal averaging by different amounts on acquired signals with $SNR = 6\,\mathrm{dB}$.

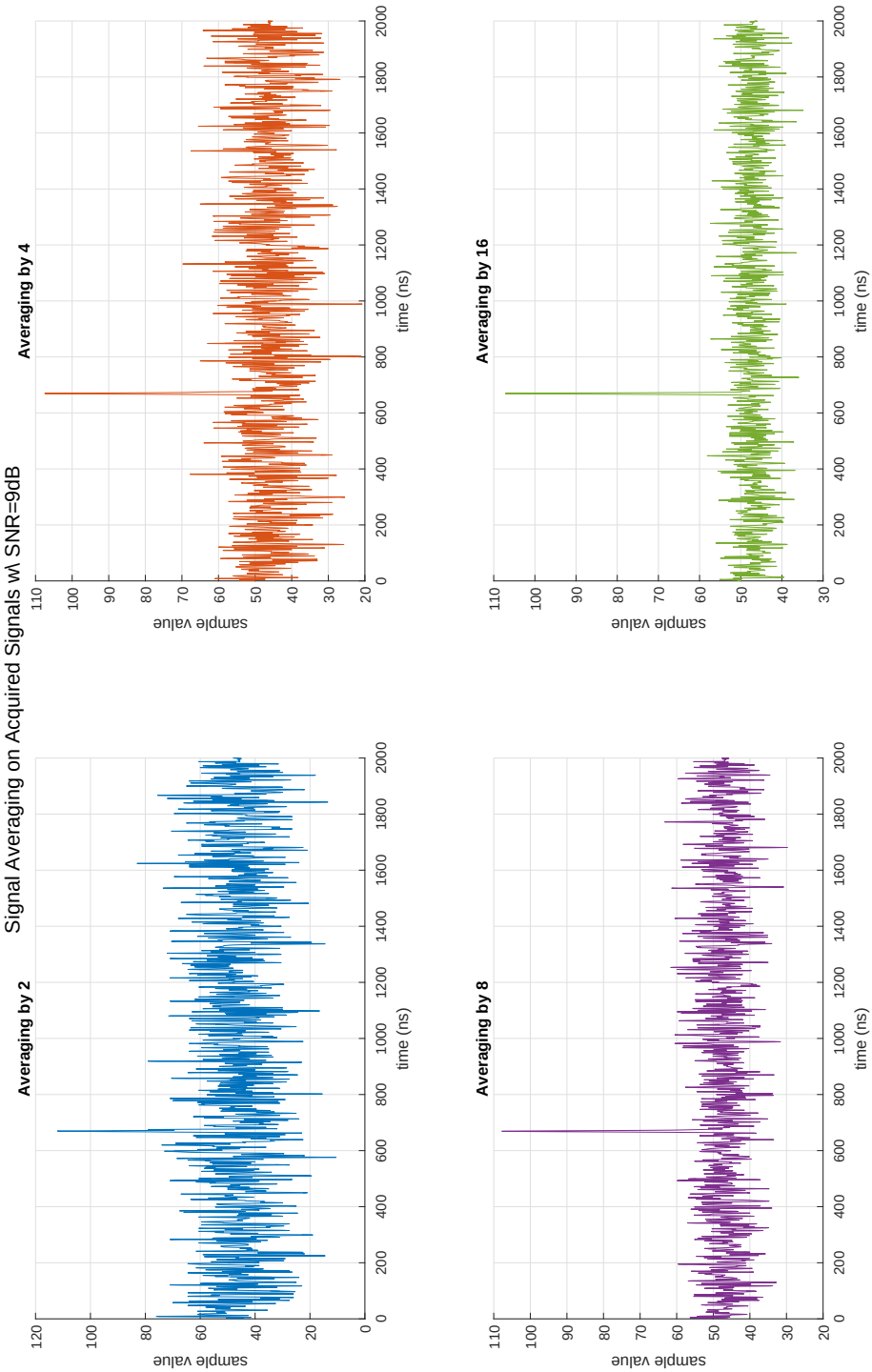

Figure E.5: Signal averaging by different amounts on acquired signals with $SNR = 9\,\text{dB}$.

References

[1] Statista - number of cars sold worldwide from 1990 to 2018. Available at https://www.statista.com/statistics/200002/international-car-sales-since-1990/.

[2] German Trade and Invest. The automotive industry in germany. Available at https://www.gtai.de/GTAI/Content/EN/Invest/_SharedDocs/Downloads/GTAI/Industry-overviews/industry-overview-automotive-industry-en.pdf.

[3] Ching-Yao Chan. Advancements, prospects, and impacts of automated driving systems. *International Journal of Transportation Science and Technology*, 6(3):208 – 16, Sept. 2017. automated driving systems;ADS products;transportation systems;vehicle automation;personal mobility services;.

[4] McKinsey and Company. A road map to the future of the auto industry. Available at https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/a-road-map-to-the-future-for-the-auto-industry.

[5] U. Scheunert, P. Lindner, E. Richter, T. Tatschke, D. Schestauber, E. Fuchs, and G. Wanielik. Early and multi level fusion for reliable automotive safety systems. pages 196 – 201, Piscataway, NJ, USA, 2007//. multi level fusion;reliable automotive safety systems;ProFusion2 project;sensor data fusion;driver assistance systems;integrated project PReVENT;vehicles detection;early fusion processing;.

[6] U. Scheunert, P. Lindner, H. Cramer, T. Tatschke, A. Polychronopoulos, and G. Wanielik. Multi level processing methodology for automotive applications. pages 6 pp. –, Piscataway, NJ, USA, 2006//. multilevel processing;automotive application;sensorial source;automatic environmental perception system;ProFusion2 project;fusion technique algorithm enhancement;vehicle environment;data structure;information handling;data fusion process;sensor data fusion;information fusion system;.

[7] Lili Huang and M. Barth. Tightly-coupled lidar and computer vision integration for vehicle detection. pages 604 – 9, Piscataway, NJ, USA, 2009//. vehicle detection;tightly-coupled LIDAR;computer vision sensors;driver assistance systems;autonomous driving applications;vehicle position estimation;regions of interest;classifier error correction;Adaboost classifier;traffic surveillance;roadway navigation tasks;.

[8] M. Kutila, P. Pyykonen, W. Ritter, O. Sawade, and B. Schaufele. Automotive lidar sensor development scenarios for harsh weather conditions. pages 265 – 70, Piscataway, NJ, USA, 2016//. in-vehicle system performance estimations;harsh weather conditions;automotive laser scanners;automotive LIDAR sensor development;.

[9] Wikipedia. Robo-taxi. Available at https://en.wikipedia.org/wiki/Robo-Taxi.

[10] Waymo. Waymo. Available at https://waymo.com/.

[11] EE Times. Lidar tech today, lidar vendors tomorrow. Available at https://www.eetimes.com/document.asp?doc_id=1333254&print=yes.

[12] Statista. Global lidar market size in 2016 and 2024. Available at https://www.statista.com/statistics/814633/global-lidar-market-size/.

[13] Statista. Global lidar system sensor sales in 2014 and 2022. Available at https://www.statista.com/statistics/430086/automotive-sales-of-automotive-lidar-systems-worldwide/.

[14] Automated driving. Available at https://www.smmt.co.uk/wp-content/uploads/sites/2/automated_driving.pdf, accessed last on 2018-02-03.

[15] P. Lindner and G. Wanielik. 3d lidar processing for vehicle safety and environment recognition. pages 6 pp. –, Piscataway, NJ, USA, 2009//. 3D LIDAR processing;vehicle safety;environment recognition;adaptive cruise control;lane change assist;intelligent vehicle safety systems;grayscale cameras;multi sensor data fusion;multilayer laser scanner;3-dimensional occupancy grid;.

[16] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto. A lidar and vision-based approach for pedestrian and vehicle detection and tracking. pages 1044 – 9, Piscataway, NJ, USA, 2007//. vehicle detection;vehicle tracking;sensorial-cooperative architecture;intelligent vehicles;in-vehicle Lidar;monocular vision;object classification method;Bayesian-sum decision rule;.

[17] Feihu Zhang, D. Clarke, and A. Knoll. Vehicle detection based on lidar and camera fusion. pages 1620 – 5, Piscataway, NJ, USA, 2014//. vehicle detection;LiDAR;camera fusion;advanced driver assistance systems;ADAS;range information;sensor fusion;hypothesis generation phase;hypothesis verification phase;stereo camera;false alarm rates;.

[18] Mordor Inteligence. Lidar market - segmented by component, product, end-user industry, and geography - growth, trends and forecasts (2018 - 2023). Available at https://www.mordorintelligence.com/industry-reports/lidar-market.

[19] SPI Lasers. Spi lasers - what is a continuous wave laser? Available at http://www.spilasers.com/industrial-fiber-lasers/redpower/what-is-a-continuous-wave-cw-laser/.

[20] Hamamatsu - mems mirrors. Available at https://www.hamamatsu.com/resources/pdf/ssd/mems_mirror_koth9003e.pdf, accessed last on 2018-02-03.

[21] Innoluce. Innoluce - choice for 1d over 2d. Available at http://www.innoluce.com/.