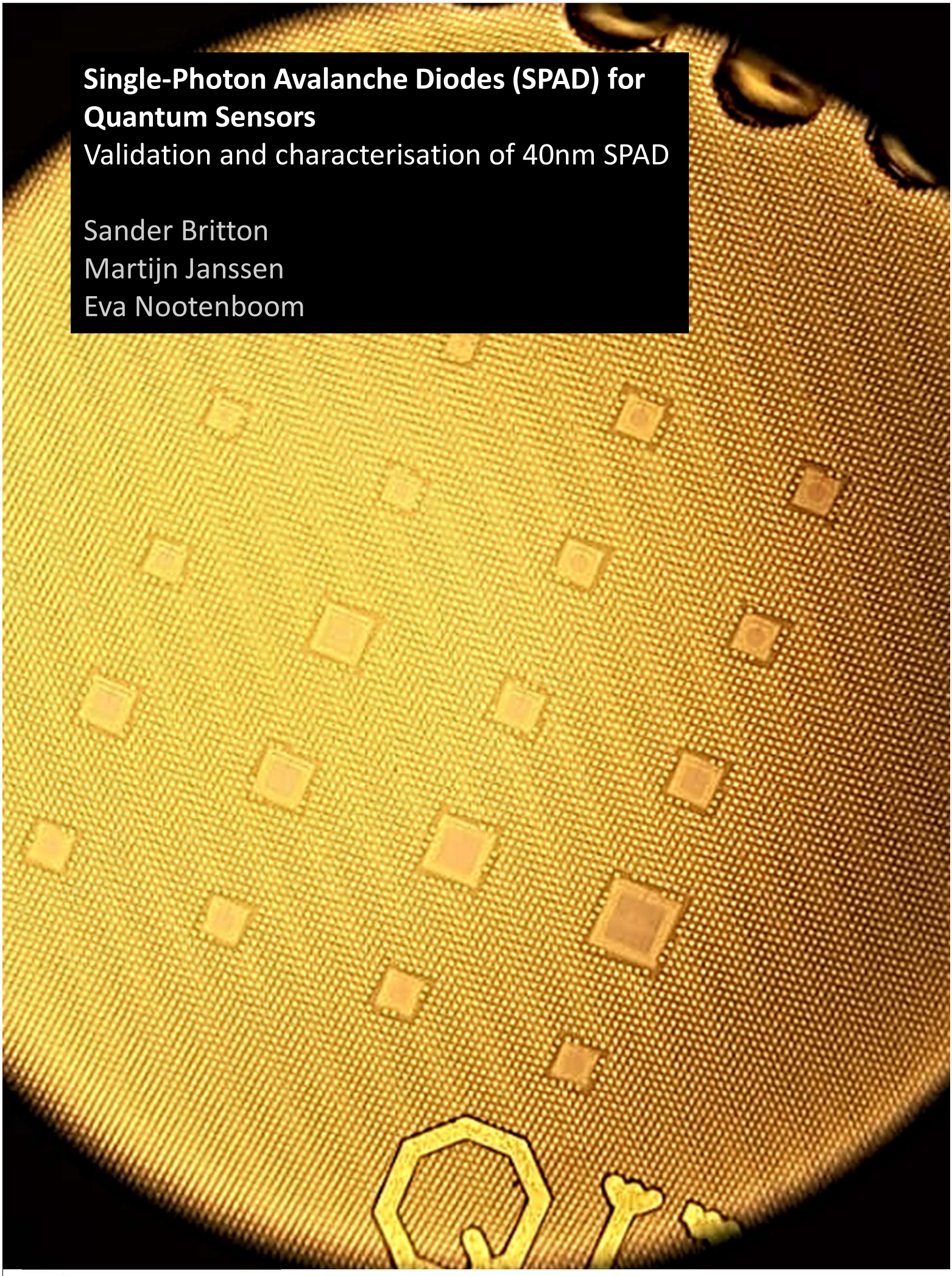


Single-Photon Avalanche Diodes (SPAD) for Quantum Sensors

Validation and characterisation of 40nm SPAD

Sander Britton
Martijn Janssen
Eva Nootenboom

TU Delft Delft University of Technology

Single-Photon Avalanche Diodes (SPAD) for Quantum Sensors

Validation and characterisation of 40nm SPAD

by

Sander Britton
Martijn Janssen
Eva Nootenboom

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Wednesday June 28, 2023 at 11:00 AM.

Student number:	Sander Britton	5110947
	Martijn Janssen	5349389
	Eva Nootenboom	5281288
Project duration:	April 24, 2023 – June 28, 2023	
Thesis committee:	Dr. R. Ishihara	TU Delft, supervisor
	I. Varveris	TU Delft, daily supervisor
	S. Wong	TU Delft
	Dr. M. Alonso-delPino	TU Delft

TU Delft

Abstract

Quantum sensing technology has emerged as a promising field with superior sensitivity and accuracy in magnetic field sensing compared to conventional technologies. This breakthrough offers significant prospects in the biomedical sector, particularly in the realm of medical imaging.

This project focuses on the implementation of quantum sensing using an array of single Nitrogen Vacancy (NV) centers combined with an array of Single Photon Avalanche Diodes (SPADs). SPADs are available in various technologies and forms. To identify the optimal SPAD for this specific application, different SPADs with diverse parameters are designed on a chip using standard 40 nm TSMC process. These parameters include active area, active area shape and metal shielding.

The research aims to validate and characterize the performance of these distinct SPADs. However, during the validation tests, it was observed that the SPADs did not meet the criteria. The presence of ESD diodes on the chip introduced a low resistance current path, thereby hindering the characterisation process of the SPADs. Solutions to fix this problem and improve the system are provided.

The documented validation methods include general chip testing, establishing the I-V curves of the SPADs and avalanche and quenching testing. In addition, the characterisation methods for junction capacitance, dark count rate (DCR), photon detection probability (PDP) and time jitter are given.

For characterizing DCR and PDP a readout system is designed. It consists of an analog-to-digital converter (ADC), FPGA and Arduino. The data is communicated continuously to a PC. The ADC is not fully tested. On the other hand, the FPGA and Arduino are tested and verified to be sufficient for the DCR measurements. The readout system is too inaccurate for PDP measurements. A recommendation on how to make the readout system sufficient for PDP is given.

Preface

This bachelor's thesis, titled *Validation and characterisation of 40nm SPAD*, was undertaken as part of the Bachelor of Electrical Engineering program at Delft University of Technology. The completion of this thesis marks the culmination of three years of dedicated effort. Engaging with one of the cutting-edge research groups provided us with a valuable opportunity to expand our knowledge and experience in the field.

Throughout this journey, we acquired a profound understanding of the research process, measurements, and design, particularly in the realm of quantum technology pertaining to Single Photon Avalanche Diodes (SPAD) and Nitrogen Vacancy (NV) centers. One of the most significant lessons we learned was to 'question everything'. We discovered that every component utilized in our experiments influenced the measurements, whether it was the lighting conditions, proximity of objects or people, or other factors. These lessons will undoubtedly serve us well in our future pursuits, whether in pursuing a master's degree or exploring other paths.

We would like to express our gratitude to our daily and head supervisors, Ioannis Varveris and Ryoichi Ishihara respectively, for their invaluable guidance throughout the project. Additionally, we extend our appreciation to Salahuddin Nur and Nikolaj Nitzsche for their assistance in brainstorming solutions to the challenges we encountered. Luc Enthoven and Niels Fakkel consistently made time for us, providing valuable help and suggestions. Furthermore, we would like to acknowledge the contributions of Job van Staveren, Zu-Yao Chang, Deb Dutta, Lukasz Pakula, and Filip Simjanoski, who facilitated our experiments and supported us in progressing to the next stages.

During this project, we had the pleasure of collaborating with Samantha van Rijs, Lars Visser, and Dagmar Westenbrink. Together, we embarked on this Bachelor Graduation Project, providing mutual support and working as a cohesive team to successfully complete the project.

This thesis was conducted within the Quantum Integration Technology (QIT) group at Delft University of Technology, and we are grateful for the opportunity to be a part of this esteemed research community.

Sander Britton
Martijn Janssen
Eva Nootenboom
Delft, June 2023

Contents

1 Introduction

This chapter will state the problem discussed in this thesis and give background information on Nitrogen Vacancy (NV) centers and Single Photon Avalanche Diodes (SPADs). Finally, an outline of the remaining thesis is provided.

1.1. Problem statement

Here the problem statement will be elaborated on from a high overview down to the specifics.

1.1.1. Quantum sensing

Quantum sensing technology offers enhanced sensitivity and accuracy compared to existing technologies[6]. One area where this technology can have a significant impact is in the biomedical sector, particularly in the field of medical imaging. This project focuses on utilizing magnetic fields for medical imaging to identify and precisely locate cancerous cells within the body.

By incorporating quantum sensing, the precision and size of the imaging machine can be greatly improved. Traditional imaging techniques, such as MRI machines, are typically large and bulky, limiting their ability to provide real-time imaging. This results in the surgeon having to rely on his own senses to distinguish healthy tissue from cancerous cells during surgery. Surgeons do not always get it right even after years of training.[16]

To make sure that all the cancerous cells are eliminated, surgeons remove additional healthy tissue around the suspected area [19], causing unnecessary harm to the patient. However, with the implementation of quantum sensing technology, the localization precision of the imaging process can be enhanced. This enables more accurate and real-time tracking of cancerous cells, reducing the need for excessive removal of healthy tissue.

Overall, quantum sensing technology has the potential to revolutionize medical imaging in the biomedical sector. By improving the precision and form factor of imaging machines, it can aid in the identification and localization of cancerous cells, ultimately leading to more successful surgeries and minimizing harm to patients.

1.1.2. NV center and SPAD pairing

The quantum sensing implementation in this project is an array of single Nitrogen Vacancy (NV) centers paired with an array of Single Photon Avalanche Diodes (SPADs). This setup should produce higher precision than systems employing NV ensembles, due to the better spatial resolution. This is thanks to knowing the exact location of the NV center and the ability to read out this specific NV center with a SPAD.

The fluorescence output of an NV center changes when a magnetic field is present. This enables the magnetic field strength to be calculated. The SPAD can detect photons and thus measure fluorescence. These paired sensors are placed next to the body to measure the magnetic field of the cells. A visualisation can be seen in Figure 1.1.

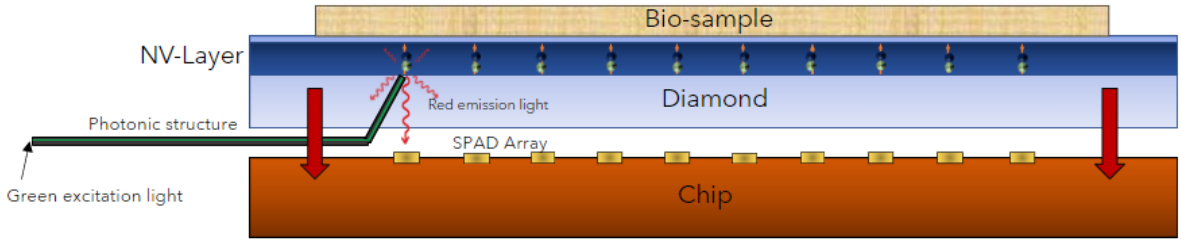


Figure 1.1: The setup for the bio-sensing implementations. The bio-sample is placed on top of a diamond with NV centers. These are aligned to the SPAD array on the chip. When the green light excites the NV center it sends out a red light. The SPAD is able to detect this photon. Image created by Ioannis Varveris.

1.1.3. SPAD characteristics

To optimise the system, the characteristics of the SPAD should be optimal for detecting NV center fluorescence. SPADs are manufactured in many different technologies and forms. To find the optimal SPAD for this use case, SPADs with different parameters are designed. The technology employed as basis for these SPADs is the TSMC 40nm process. This process is chosen because of its world class technology and form factor. SPADs built utilising the standard TSMC 40 nm process for NV center fluorescence measurement is novel. The chip created with the different SPADs is shown in Figure 1.2 and an image with active areas to scale can be seen in Figure A.2. The chip contains twenty different SPADs. The SPADs differ in active area, active area shape and metal shielding. The Table 1.1 shows these aspects per SPAD matched to Figure 1.2. The sensor will be placed next to the body, therefore characterisation should be done in the temperature range between room temperature and body temperature (37 °C).

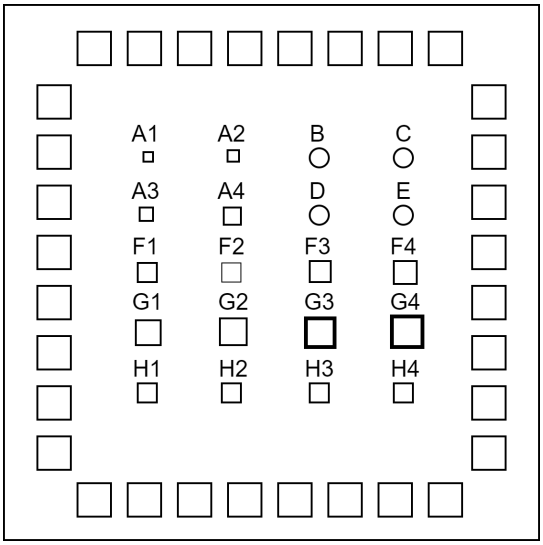


Figure 1.2: The SPADs on the chip grouped by their common cathode pin. The inner squares represent the SPADs. The outer squares the I/O pads. The areas are not to scale.

1.1.4. Project overview and steps taken

The choice of SPAD design parameters consists of the following steps: design, validation, characterisation, implementation. In this thesis validation and characterisation will be discussed. The design of the different SPADs was done by Stein Fakkel and Ioannis Varveris of Delft University of Techonology's Quantum Integration Technology group led by Ryoichi Ishihara. During the validation the general behaviour of the chip and the behaviour of the individual SPADs is verified, to verify if it matches the expected behaviour. The characterisation consists of multiple measurements to characterise the individual SPADs for different parameters. Finally, the SPADs can be compared and the best SPAD and its corresponding parameters for detecting NV center fluorescence can be chosen.

Table 1.1: Table of SPAD names with active area. *If no aspect is given, square SPAD with no extra metal is assumed.

SPAD name	Active area (μm^2)	Specialty*
A1	2x2	
A2	4x4	
A3	6x6	
A4	8x8	
B, C, D, E	12x12	circular active area
F1	10x10	
F2	10x10	extra metal shielding
F3	14x14	
F4	16x16	
G1	18x18	
G2	20x20	
G3	26x26	extra metal on top of the active area
G4	36x36	extra metal on top of the active area
H1-H4	12x12	

1.2. Theory

In this section, the theory behind NV centers and SPADs is discussed.

1.2.1. Nitrogen Vacancy Center

In this section the NV center will be discussed in more detail. First the technology behind it is explained, followed by the behaviour that makes the NV centers so useful. Lastly, the readout of the magnetic field using the NV center is discussed.

Nitrogen Vacancy Technology

The NV center is a point defect in a diamond where two adjacent carbon atoms are substituted by a nitrogen atom and a vacancy [13]. The nitrogen atom has five valence electrons instead of the 4 carbon has. Three of the electrons will form covalent bonds with the neighbouring carbon atoms, the last two are a lone pair. The vacancy has three unpaired electrons from its neighbouring carbon atoms, of which two will form quasi-covalent bonds and the last one will form a bond with one of the remaining electrons of the nitrogen atom, resulting in one electron remaining unpaired. This system presents a spin and is called the NV^0 configuration. The NV center is free to trap a free-floating electron from the conduction band, thus obtaining negative charge denoted by NV^- [11]. This NV^- configuration holds all the properties needed for the intended application. For simplicity this configuration is referred to with the term 'NV center'.

Behaviour of a Nitrogen Vacancy Center

The energy diagram of the NV center can be found in Figure 1.3. It consists of a ground state $|g\rangle$, an excited state $|e\rangle$ and a shelving state $|s\rangle$. All these states are within the band gap of the diamond. Without an external magnetic field there are two magnetic spin levels in both the ground and excited state: $|m_s = 0\rangle$ and $|m_s = \pm 1\rangle$.

Electrons can be moved from the ground state to the excited state with a green laser. The excitation process is spin-conserving, so electrons that are in $|0, g\rangle$ will go to $|0, e\rangle$ when excited, just like electrons with spin $|\pm 1\rangle$. In the excited state electrons can decay back directly to the ground state by emitting a red photon (see Figure 1.3) with wavelength 600-850nm[13]. This decay path is radiative and spin-conserving. Electrons with spin $|m = \pm 1\rangle$ can also decay to the ground state through the shelving state. This process is not spin-conserving. Electrons in $|\pm 1, e\rangle$ that decay non-radiatively predominantly, through the shelving state, end up in $|0, g\rangle$. When exciting the system with a green laser for a sufficient amount of time, most electrons will be in the $|0\rangle$ spin state. After typically a few microseconds the system is in the $|0\rangle$ state with an accuracy approaching 95%.[13] When a system is initialized this way it is possible to read-out its spin optically.[3][11][17]

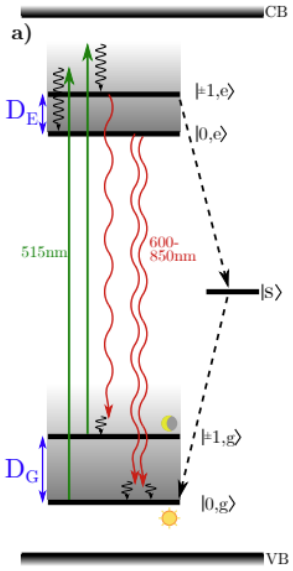


Figure 1.3: The energy diagram of the NV center. Two spin conserving decay paths and one indirect spin non-conserving decay path are shown.[13]

NV Center Readout

There are multiple measurement schemes to readout NV centers. Two of these are: pulsed ODMR (PU-ODMR) and continuous wave ODMR (CW-ODMR).[5][42] ODMR stands for Optically Detected Magnetic Resonance.

In the CW-ODMR measurement scheme the green laser, microwave (MW) and read-out all happen continuously at the same time. The green laser is used to constantly excite the electrons. At the same time the microwaves (MWs) act on the electrons. At the frequency resonant to the $|0\rangle \leftrightarrow |\pm 1\rangle$ transition the centers are excited. As discussed before in Section 1.1 this is apparent via a decrease in fluorescence, which is also constantly read out. When the fluorescence is plotted against the MW frequency, a dip is observed as seen in Figure 1.5d. When a static magnetic field is applied, the $|\pm 1\rangle$ is split into $|+1\rangle$ and $|-1\rangle$, this is called the Zeeman effect. This is observed in the ODMR spectrum by two dips instead of one as can be seen in Figure 1.4. The distance between the dips can be used to determine the magnetic field. The magnetic field in the direction along the quantization axis of the NV center can be determined with the following equation[13]:

$$B_q = \frac{h}{2g_e\mu_B}\Delta\nu$$

The PU-ODMR works mostly in the same way. The difference with CW-ODMR is that the NV center is not constantly polarised with the laser and MWs, instead they are pulsed. First the laser pulse is sent to polarise the NV to $m_s = 0$ state. Then, the MW pulse is sent to drive the centers to the $m_s = \pm 1$ state. After the MW pulse, the NV state is read. Instead of the frequency sweep, one frequency is tested per cycle (Figure 1.5e). This method is more difficult to setup, but gives more accurate results.[4][5][7] The fluorescence rates of NV centers is in the range of hundreds of kilo counts/s [5][13].

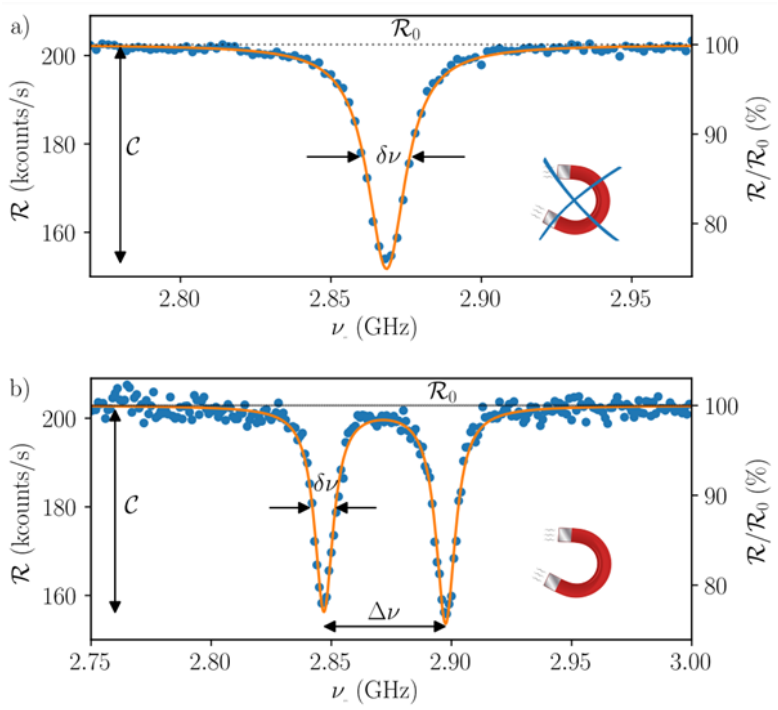


Figure 1.4: Fluorescence spectrum of a NV center where in a) there is no static magnetic field and in b) there is a static magnetic field, thus showing the splitting of the dips known as the Zeeman effect.[13]

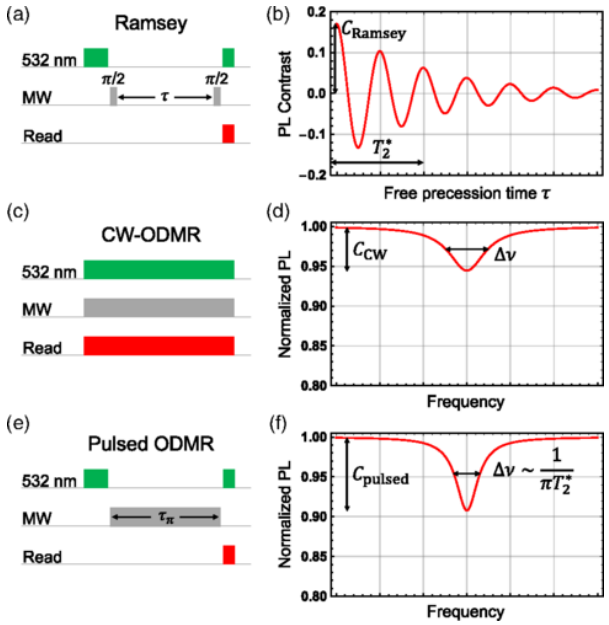


Figure 1.5: Overview of Ramsey, CW-ODMR and PU-ODMR measurement schemes. (a) and (b) are applicable to the Ramsey method, which will not be discussed in this study. (c) Schematic overview of CW-ODMR protocol. (d) CW-ODMR spectrum with contrast C_{cw} and linewidth $\Delta\nu$. (e) Schematic overview of PU-ODMR protocol with MW pulse time $t_\pi \sim T_2$. (f) PU-ODMR spectrum with contrast C_{pulsed} and linewidth $\Delta\nu$.[5]

1.2.2. SPAD Technology

In this section the SPAD will be discussed in more detail. First the working principal is explained. Next the characteristics of the SPAD are given and elaborated on.

Working principal

A SPAD is a special type of photo diode. When a SPAD is biased below it's reverse breakdown voltage, it can detect single photons[40]. In this state, biased below breakdown, negligible current flows through the SPAD, until a photon hits the active area. This hit causes a sudden increase in current called an avalanche. This avalanche takes place in nanoseconds and in a magnitude range of milliamperes. The amount of avalanches can be related to the amount of photons that hit the SPAD's active area, enabling precise photon counting capabilities. This current spike needs to be quenched (Section 1.2.3) to prevent damage to the SPAD and to rearm the device for a following photon. This involves lowering the voltage over the SPAD. Quenching is complete when the voltage is reduced below or equal to the breakdown voltage. After this, the SPAD can be rebiased to enable it to detect another photon. This process is shown in Figure 1.6 and is further elaborated on in Section 1.2.3. SPADs operating with this behaviour are called to be in Geiger mode.

Characteristics

A SPAD has several characteristics that determine its electrical behaviour and photon detection capabilities. The characteristics are the following:

Dead time: The dead time is the time that the SPAD becomes inactive after it detects a carrier [28]. This inactive period lasts until the circuit has been quenched and the bias voltage is restored [18]. This is largely determined by the quenching circuit.

Junction capacitance: The junction capacitance of a SPAD consists of the anode-to-cathode and cathode-to-substrate capacitance and it impacts both the quenching and recharge time[10]. The capacitance is dependent on the applied voltage.

Dark count rate: The dark count rate is the frequency at which detection events are triggered by non photo-generated (dark) carriers, i.e avalanches that are the result of thermally generated carriers, afterpulsing effects or crosstalk in SPAD arrays. It is dependent on excess voltage and temperature.[31]

Photon detection probability: The photon detection probability is the probability that a photon in the active region of the SPAD triggers an avalanche. This is always less then one and it depends on factors such as parasitic reflection, light's penetration depth, the excess voltage and photon wavelength.[10][31]

Time jitter: It is defined as the uncertainty in time interval between the arrival of a photon on the SPAD and when the avalanche is triggered [31]. Jitter is affected by excess voltage and SPAD technology. Jitter measured at different wavelengths is compared to each other in literature.[15][22][25][27][38] Vornicu, I.(2019)[36] found time jitter at 447nm and 640nm to be 208ps and 188ps respectively.

Afterpulsing: Afterpulsing is a correlated noise. It is caused by a trapped carrier left from a previous avalanche. This carrier is trapped in an energy level close to the energy band. When these trapped carriers are released, while the SPAD is sensitive to carriers, it could cause an undesired avalanche known as afterpulsing. [29][31].

1.2.3. Quenching circuit

As mentioned before, the general idea behind the quenching circuit is to bring the reverse-biased voltage, which is the breakdown voltage together with the excess voltage $V_{bd} + V_e$, down to the breakdown voltage, reducing the current to close to zero. Then, to re-arm for the next photon, the SPAD is rebiased back to $V_{bd} + V_e$. As can be seen in Figure 1.6.

There are two quenching circuit types: active and passive. Active quenching circuits use transistors to increase the quenching speed [39]. Passive quenching circuits use passive components like resistors to quench the avalanche. The advantage of an active circuit is faster quenching, faster recharge time and adjustable dead time. The disadvantage is that this circuit is difficult to implement on a PCB. This can be attributed to the pronounced impact of variations in gate lengths, as well as stray capacitances and resistances on the behaviour of the circuit. The advantage of a passive circuit is its simplicity. The disadvantage is the possibility of the SPAD triggering before it is fully recharged to the excess voltage. Because of this, the dead time varies widely.[37]

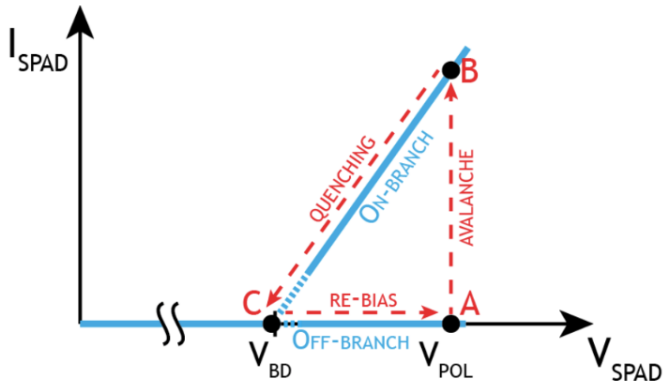


Figure 1.6: I-V characteristics of a SPAD. Before photon has reached the SPAD, the SPAD is biased above the breakdown voltage in point A. When a photon hits on the active area of the SPAD it triggers an avalanche. The current rises (B). Next the quenching circuit lowers the voltage below the breakdown voltage (C), after which the voltage is restored (A). [8]

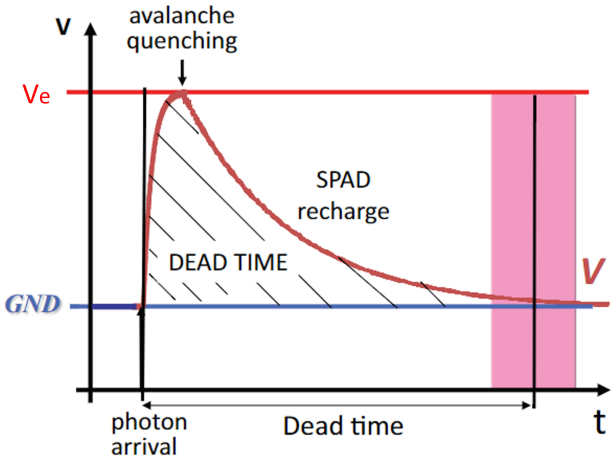


Figure 1.7: Time plot of the voltage over the ballast resistor. The voltage over the ballast resistor rises when a photon hits the SPAD due to the increase in current, this is the quenching part. After that the voltage will go down again, this is when the SPAD is recharged. [31].

The simplest quenching circuit is a passive circuit consisting of only a ballast resistor, R_b, typically in the order of $k\Omega$. The resistor is connected in series with the SPAD, limiting the avalanche current. For a quench to be successful, $\frac{V_e}{R_b}$ should be smaller than the SPAD's latch current[10]. When a SPAD is 'latched', it remains in the avalanche breakdown state even after the excess charge has been removed due to trapped charge carriers. The latch current is the minimum current required to keep the SPAD in the 'latched' state.

The diode's recovery time to return back to its initial value is the exponential law RC constant of the ballast resistor, R_b, and the SPAD's reversed-polarized capacitance, C_{SPAD}. To maximize the photon detection probability (PDP), the recovery time should be minimized which is limited by the latch current and C_{SPAD}, $t_r = R_b C_{SPAD} = \frac{V_e}{I_{latch}} C_{SPAD}$. Consequently, the photon counting frequency is highly dependent on the recovery time. Figure 1.7 shows how the voltage over the ballast resistor reaches the excess voltage and then drops down exponentially to the ground level. Then this voltage will bounce back to the excess voltage to be re-armed for the next photon [30].

Figure 1.8 is an example of a basic passive quenching circuit. The applied voltage is $V_a = V_{bd} + V_e$, where V_{bd} is the breakdown voltage and V_e the excess voltage, which reverse biases the SPAD. The product of the ballast resistance R_b, and the avalanche current should become larger than the excess voltage for a quench to become successful. Therefore, $R_b = \frac{V_e}{I_{avalanche}}$. Some variations of the passive quenching circuit contain an additional small sense resistor in series [10].

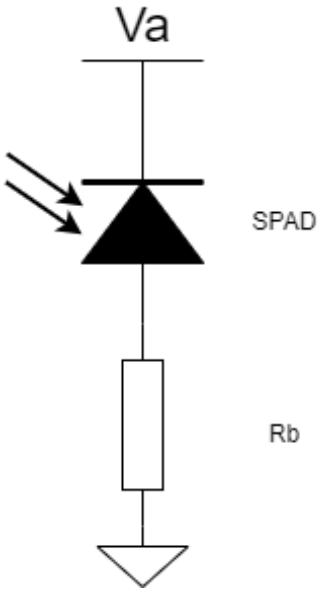


Figure 1.8: Example of a basic passive quenching circuit. The SPAD is reversed biased with V_a and the R_b value is determined by the avalanche current levels.

1.3. State-of-the-art analysis

In recent years, there has been research interest in different areas related to SPAD technology. One of these areas focuses on enhancing the pixel density and fill factor of SPADs[33]. This resulted in the development of the first SPAD camera with 1 mega pixel in 2021, utilising a pixel pitch of 2.2μm[24]. An additional area explores the integration of SPAD technology into diverse applications, such as scintillating fibers[12], spectrometers[32] and 3D range imaging[21]. Advancements in SPAD technology are also being made through the integration of SPADs into emerging fabrication technologies. This includes fully developed SPADs in 40nm technology[25] with the inclusion of microlenses and the first steps in 28nm technology [2]. Besides this, the incorporation of more efficient absorption layer material are enhancing the detection probability, like the use of a Germanium-on-Silicon achieving a peak PDP of 38% at a temperature of 125K and wavelength of 1310nm in 2019 [35].

1.4. Thesis synopsis

The subsequent sections of the thesis are organized as follows: Chapter 2 presents the program of requirements. The validation steps are outlined in Chapter 3, while Chapter 4 documents the characterisation measurements. Chapter 5 delves into the discussion of the readout circuit required for specific measurements. The findings and analysis of the measurements are presented in Chapter 6. Finally, Chapter 7 concludes the thesis by summarizing the key outcomes and providing recommendations for future research endeavors.

2

Programme of Requirements

As mentioned in Section 1.1.4, the goal of the project is to perform the validation and characterisation steps of choosing SPAD design parameters for NV center fluorescence measurement. A recommendation should be provided for the most suitable design of the SPAD that aligns with this goal, along with the corresponding optimal excess voltage.

2.1. Validation requirements

It is important to validate the chip design and fabrication. The following aspects need to be validated:

- Correct connections to SPADs.
- Proper function of the I/O ring.
- SPAD I-V behaviour matching literature.
- The quenching results match the simulation.

2.2. characterisation requirements

The following aspects need to be characterized to get a good idea of the electrical and photon detecting behaviour.

- Breakdown voltage and resistance in breakdown.
- Junction capacitance.
- Dead time: the period in which the SPAD cannot be triggered.
- Dark count: avalanche frequency without any photon hits.
- Photon Detection Probability (PDP) over the visible spectrum: probability that a sent photon triggers the SPAD.
- Timing Jitter: delay between photon hit and avalanche.

The design and simulation of the quenching circuit require the knowledge of the breakdown voltage, resistance in breakdown, and junction capacitance. An essential metric for assessing the efficiency of the SPAD is the dead time, which is predominantly determined by the characteristics of the quenching circuit. Another crucial aspect to consider is the dark count, which is an important noise effect. Both the dead time and dark count have an impact on the Photon Detection Probability (PDP), which serves as a primary indicator of the SPAD's efficiency. Additionally, it is worth noting that timing jitter, although a minor effect, plays a role as noise factor. Only room temperature conditions should be taken into account, since other temperatures are out of the scope of this project.

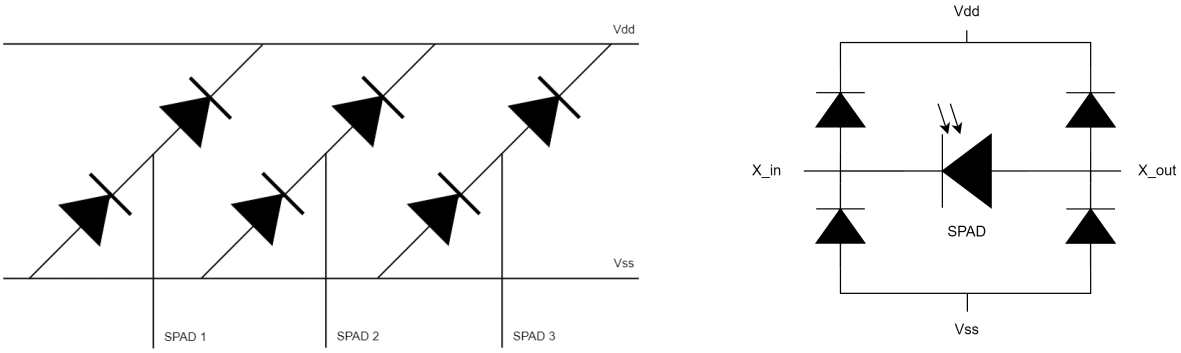
For dark count and PDP measurement, it is required to have a readout circuit. The readout circuit requirements are derived from the project description, in combination with the NV center and SPAD characteristics. These are as follows:

- Detect the spikes given by the SPAD and quenching circuit.
- 2 Mcounts/s readout capability.
- Continuous data logging to a PC.
- Operable in room temperature.

3

Validation

Before the SPADs can be characterised they need be validated to make sure that everything is working as expected. To do this, it is important to verify the design of the chip. The chip consists of 32 I/O pins and 20 SPADs with different active areas as can be seen in Figure 1.2 . The I/O ring consists of Electro Static Discharge (ESD) diodes that go from a pad to Vdd in forward bias and diodes that go from a pad to Vss in reverse bias. This connection can be seen in Figure 3.1a for three pads. These ESD diodes prevent the SPADs from electro static discharge damage and overvoltage. One SPAD and its connections to the pads and Vdd and Vss rails through the ESD diodes can be seen in Figure 3.1b. In Figure A.1 the pinout can be seen for the whole chip.



(a) Part of the I/O ring.

(b) One SPAD with all its connections.

Figure 3.1: The design of the chip.

3.1. General chip testing

To ensure the chip works correctly and the pinout matches the documentation the following items need to be tested:

- The I/O ring safety diodes from Vdd and Vss to every pin on the chip, in reverse and forward bias.
- SPADs in forward and reverse bias, to check pinout.
- Basic I-V curve of the SPADs -2.5V to 2.5V.
- Basic I-V curve from 0V to breakdown.

The ESD diodes found in the I/O ring of the chip are typically used up till 3.3V, beyond this voltage the behaviour is unsure. The expected breakdown voltage of the SPADs is around 8V (Figure 3.4). The ESD diodes are not rated for these voltages, therefore Vdd and Vss will be kept floating in upcoming tests. The pinout of the chip is not well documented, therefore extra care has to be taken into account.

To perform these and subsequent measurements, a ceramic DIP-32 package is chosen. This package enables fast development by the pins from the package and easy connection to Perfboard (Dot PCB). The disadvantages of this package are the parasitic capacitances added by the package and the socket used to connect DIP package to the PCB. These capacitances are bigger than when the chip is glued directly on a custom PCB.

Method

Testing circuit (Figure 3.2): A 100$k\Omega$ resistor is used in series with the SPAD to limit the current as to not damage the chip. The expected breakdown voltage from simulations, done with Cadence utilising a standard p-diode with the same active area dimensions, is 8.9V (see Figure 3.4). The maximum applied voltage is taken to be 10 V, just above the expected breakdown voltage. With this taken into account and utilising a 100$k\Omega$ resistor, the maximum current would be 100μA. The maximum current before burnout of the SPADs is not well defined, due to nature of their novel design. Furthermore, the TSMC guidelines[34] do not give clear indications on the maximum current for diodes. Therefore, the maximum current is based on the maximum value for resistors with the inclusion of a safety margin. Based on the maximum allowable current of a resistor of the same dimensions, in 40nm TSMC technology, an estimation of 100 μA is made. Therefore, this design should not damage the chip. A current meter is present in series to track the current.

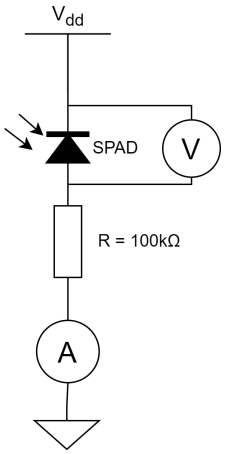


Figure 3.2: Chip testing circuit.

Testing sequence (Figure 3.3): First the safety diodes are tested in forward bias, by slowly increasing the voltage while checking the current. This is then repeated with the SPADs. This procedure is also used to check the pinout. If the current does not rise, the diode is connected in reverse bias therefore the pinout is then reversed. Hereafter the SPADs are swept in reverse bias until breakdown. If an expected I-V curve is seen the SPADs have passed the test.

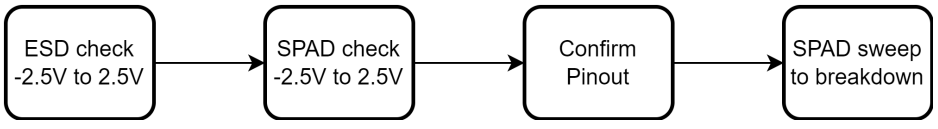


Figure 3.3: Flowchart of general chip testing sequence.

3.2. I-V curve

The I-V curve measurement is done to confirm the diode behaviour of the SPAD and find the derivable characteristics. These derivable aspects are used for the design of the quenching circuit. The internal resistance of the SPAD in reverse bias is unknown. An estimate based on literature gives a range of kilo Ohms for the reverse bias resistance.[41] The leakage currents of the SPADs are in the nano to pico Ampere range, so noise is a very prominent factor here. Shielded cables are used to limit this influence.

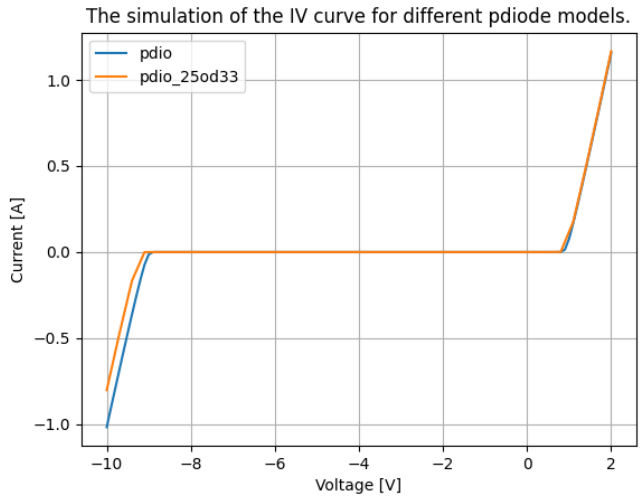
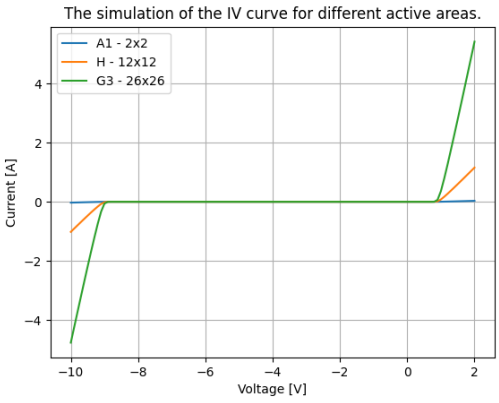


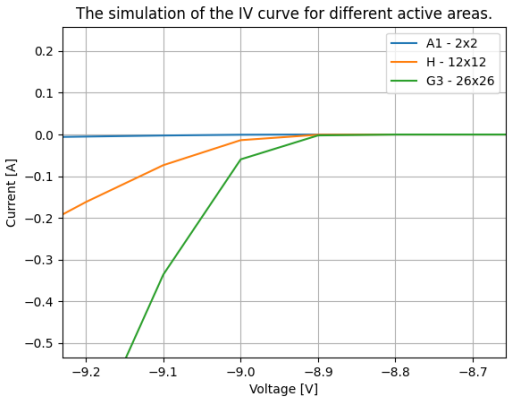
Figure 3.4: Simulation of an I-V curve using different diode models. The model ”pdio” has a breakdown voltage of -8.9V and the model ”pdio_25od33” has a breakdown voltage of -9.1V.

Three important aspects can be derived from the I-V curve: the breakdown voltage and the resistance in the neutral and breakdown region. The breakdown voltage is the voltage at the point in the I-V curve where the current starts to rapidly decrease, with respect to the leakage current [1]. Beyond this point the SPAD gets into Geiger mode and thus shows avalanche behaviour. The neutral region resistance is the resistance of the SPAD in reverse bias before breakdown. The internal breakdown resistance is the resistance of the SPAD in reverse bias after breakdown. These resistances can be found by taking the derivative of the I-V behaviour in these regions. These characteristics are important for the design of the quenching circuit.

The leakage current is in the range of nano Ampere or smaller. The breakdown voltage is hard to predict because it differs majorly between technology[15][20]. A simulation in Cadence of a standard p-diode with similar dimensions gives a breakdown of around -8.9V (see Figure 3.4), while a paper about 40nm SPADs found a breakdown voltage of -15.5V[25]. The same way the current increases rapidly in forward bias, the current past breakdown decreases rapidly when a photon hits the SPAD.[15] How rapidly the current increases/decreases depends on the resistance of the diode which is related to the active area as can be seen in Figure 3.5. The breakdown voltage however does not depend on the active area as can be seen in Figure 3.5b.



(a) Simulation of an I-V curve with different active areas.



(b) The I-V curve zoomed in around the breakdown voltage.

Figure 3.5: Simulation of an I-V curve using model ”pdio” with different active areas.

Method

To measure the I-V curve of the SPADs a voltage sweep is performed, during which the current is measured. A current limit of $50\mu A$ is set. For the measurements a Source Measure Unit (SMU), more precisely the Keithley 2636B, is directly connected to the SPAD. The sweep is done in two sets per SPAD. First from 0V to 0.8V with steps of 0.02V in forward bias, second in reverse bias from 0V to 8V with steps of 0.1V. This is done for all SPADs. Additional measurements were done with a Keysight B2912A Precision Source/Measure Unit both over SPADs and over not connected pins of the chip, to characterise the diodes of the I/O ring.

3.3. Avalanche and Quenching

The decision for a basic passive circuit is made because of its implementation simplicity, as mentioned in Section 1.2.3, while still fulfilling the requirements for the upcoming tests.[10] The configuration consists of a large ballast resistor in the range of kilo Ohms followed by the SPAD and a smaller sense resistor both in series. The sense resistor is added to convert the avalanche current to a reasonable voltage that can be fed into an analog to digital converter.

With the found values of the I-V curve a ballast resistor of 47kOhm and a sense resistor of 4.7kOhm are found utilising simulations (Figure 3.6) and formulas (Section 1.2.3). The simulations for this quenching circuit (See Figure 3.6) give an expected peak of 400mV and a dead time of 200ns. These simulations are based on two models created by Mita et al. [14][23].

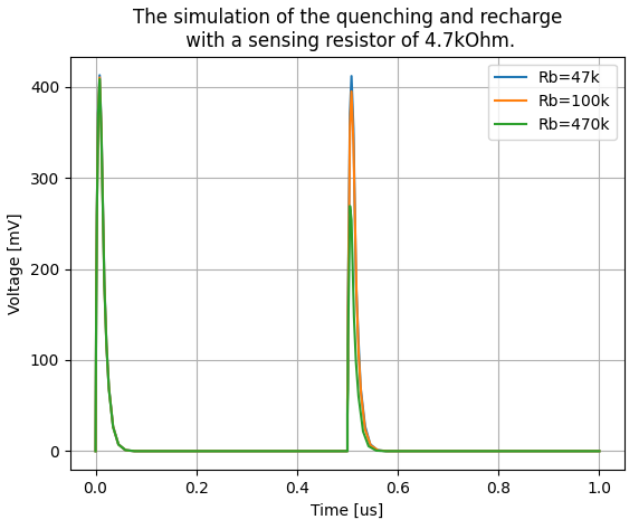


Figure 3.6: The simulation results with a sense resistor of $4.7k\Omega$ and a varying ballast resistor.

Method

The quenching of every SPAD is verified by attaching the SPAD to the quenching circuit, then the voltage over the quenching system is increased till a pulse is seen or the current limit is reached. The voltage over the ballast resistor (measured with a Fluke 177 multimeter) and the waveform of the voltage over the sense resistor (measured with a Tektronix TDS2022C oscilloscope) are measured. The measurement setup can be seen in Figure 3.7.

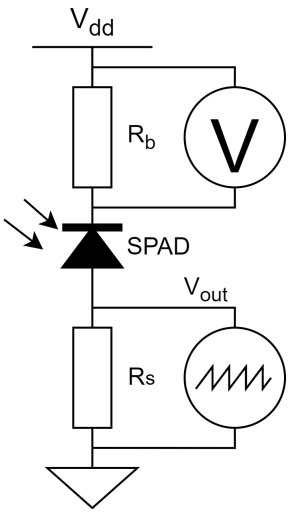


Figure 3.7: Quenching verification measurement setup.

3.4. Additional tests

If the SPAD does not show the intended behaviour the following tests are employed to find the cause.

- The chip is checked under a microscope to check for any damage or dirt. The damage can be caused by burnout or scratches.
- The sense to ballast resistor ratio is changed, in favour of the sense resistor direction. This increases the pulse amplitude measured on the output, therefore increasing the SNR (signal to noise ratio). The total resistance should not be increased too much, otherwise the recharge time will be too long.
- The system is isolated from noise effects coming from outside the system by putting it in a Faraday cage. This decreases the noise, thereby increasing the SNR.
- The connection and I-V curve between two unconnected pins is checked, to check for unwanted connections or behaviour.

4

Characterisation

To find out which SPAD is the most suited for this system, their characteristics need to be determined. This consists of multiple measurements, which are elaborated on in the following sections. They include junction capacitance, dark count rate, photon detection probability and time jitter. These characterisation measurements can only be done when the verification was successful.

4.1. Junction Capacitance

The junction capacitance of a SPAD impacts both the quenching and recharge time and it plays a crucial role in improving the accuracy of the SPAD computer model used for simulation [41]. When measuring the junction capacitance, it is necessary to account for various stray capacitances. To mitigate the effects of socket and PCB capacitances, the measurement is performed directly on the die of the SPADs.

On the die, the capacitance between pads and the capacitance between metal wires must be considered. The pad-to-pad capacitance can be determined by measuring the capacitance between the ”Not Connected” (NC) pad on the chip and an adjacent pad. However, the metal-to-metal capacitance, which arises from the adjacency of wires, is more challenging to eliminate. A reference measurement between two non-connected wires located close together can be used to account for the metal-to-metal capacitance.

Each SPAD consists of two junction capacitances: anode-to-cathode and cathode-to-substrate (as depicted in Figure 4.1). Capacitance values of diodes differ when biased in the forward or reverse. Since SPADs are operated in reverse bias, only the reverse bias capacitances are measured.

According to existing literature[41], the capacitance is expected to decrease as the DC offset increases. The values expected are in the range of single pico Farad or hundreds of femto Farad.

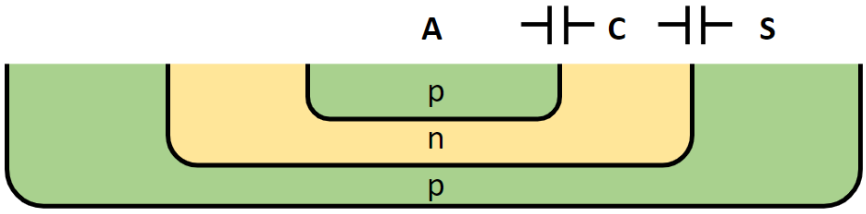


Figure 4.1: The physical model of the SPADs, showing the anode A, cathode C and substrate S. The p and n stand for p-type and n-type silicon.

Method

Once the die is set up in the probe station, the probe station needles are utilised to establish connections with the chip (see Figure 4.2 for the setup). Subsequently, the chip is moved downwards, disconnecting it from the needles. At this stage, a calibration measurement is performed with the free-floating needles. This calibration is necessary to account for the capacitance between the needles themselves and is conducted prior to each measurement.

For each SPAD, two measurements are conducted. The first measurement involves measuring the capacitance from the anode to the cathode while keeping the substrate floating. The second measurement entails measuring the capacitance from the cathode to the substrate while keeping the anode floating.

These measurements are carried out using a Keysight 4284A Precision LCR meter. Capacitance-voltage (CV) measurements are performed with a DC offset ranging from 0V to 7.5V, with 101 steps in between, at a frequency of 1MHz and an AC signal amplitude of 0.1V.

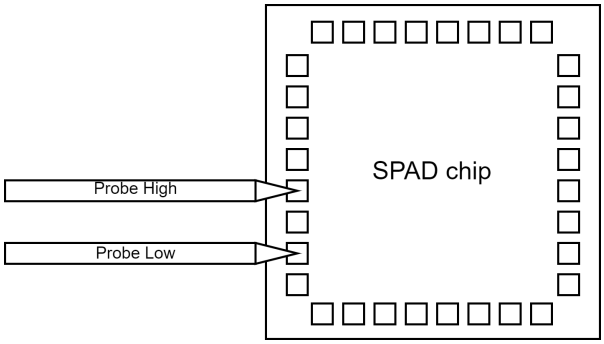


Figure 4.2: CV measurement setup using a probe station.

4.2. Dark count rate

The dark count rate (DCR) refers to the frequency at which the SPAD undergoes avalanche breakdown without any incident photons being present. This measurement is done in the context of PDP analysis, as it allows the subtraction of randomly triggered events from the genuine trigger events.

To determine the DCR, the SPAD is operated in a condition of complete darkness. The rate is then measured by recording the frequency of avalanche events occurring in the absence of any external photon stimulation. The dark count comprises various sources of noise, including avalanches triggered by thermal carriers and trapped charges that remain after quenching these. Importantly, the dark count events do not occur periodically but are randomly triggered. The DCR is influenced by the excess voltage applied and temperature. The DCR is expected to be in the kHz range and linearly dependent on the applied excess voltage[31].

Method

The measurement setup for dark count analysis utilizes the total readout system, as shown in Figure 5.1. The SPAD and its quenching and analog to digital converter (ADC) circuit are enclosed within a box in a dark room. The box contains very small openings to accommodate necessary wiring while making sure that no light comes in. The FPGA and Arduino components are positioned outside of the box. With this configuration, the counts per second are measured for each individual SPAD, enabling the quantification of the dark count rate.

4.3. Photon Detection Probability

The Photon Detection Probability (PDP) is the probability that a photon hitting the active region triggers an avalanche.[31] This probability is measured for different wave lengths. The PDP is the prime factor with which the different SPAD designs will be compared. The wavelength emitted by the NV center (600-850nm [13]) is the most important area of the PDP spectrum. Afterpulsing has to be taken into account with this measurement as the follow up pulses, which are not triggered by photons, can still be detected. One way to prevent this, is to raise the threshold of the ADC above the second largest peak.

The PDP measurement should be done in a dark room with only the laser as a light source to prevent additional triggers. The PDP peak efficiency is expected to be above 10% and be located below 600nm. The PDP is expected to drop to single digits percentages around 850 nm.[31][36] PDP is dependent on excess voltage. An increase in excess voltage leads to an increase in PDP [10][31].

Method

A variable wavelength laser is used to trigger the SPAD. This laser is attenuated and split to a reference device (LaserComponents Count-10C) and the SPAD. This reference device has a predetermined PDP. Using this, the PDP of the SPAD can be determined. The laser beam is bigger than the SPAD itself, which needs to be taken into account when calculating the PDP. However, this difference can be accounted for, since the beam size and the active area of the SPADs are known.

$$PDP = \frac{amount\ of\ detected\ photons}{total\ amount\ of\ photons}$$

The measurement setup for the PDP measurement can be found in Figure 4.3.

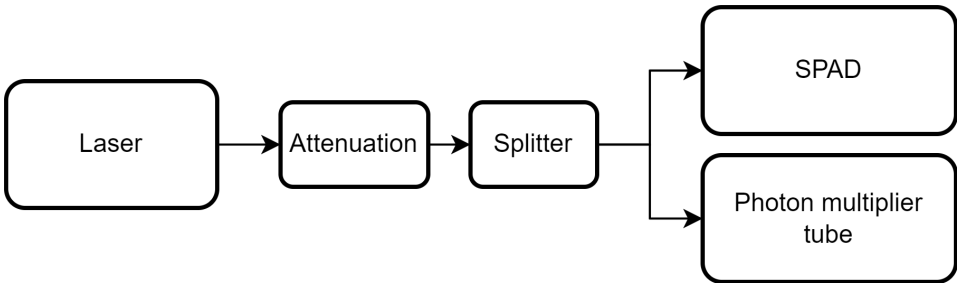


Figure 4.3: Measurement setup PDP.

4.4. Time jitter

The timing jitter is defined as the uncertainty in time between the arrival of a photon on the SPAD and when the avalanche is triggered. Jitter is affected by excess voltage and SPAD technology [31]. It is also affected by the wavelength at which it is measured [36], therefore a picosecond laser with a wavelength in the NV fluorescence range is chosen. From the acquired measurement results using the setup shown in Figure 4.5 a Gaussian curve like in Figure 4.4 should be found. As can be seen in the graph, the time jitter is the full width at half maximum (FWHM).

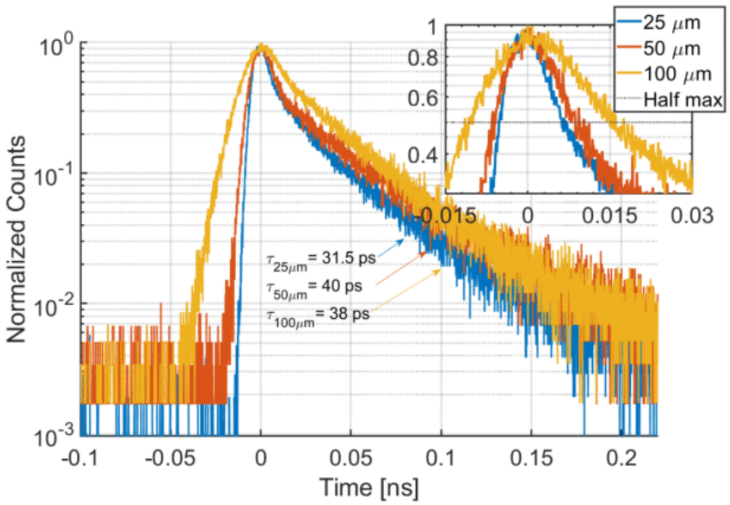


Figure 4.4: The expected outcome of the jitter measurement. [15]

Method

For the jitter a picosecond laser, high speed oscilloscope (40Gs/s →resolution of 25ps) and reference photo diode (PD) with predefined jitter are utilised. The laser first gets split to the reference PD and to the SPAD via a Neutral Density Filter (NDF) (see Figure 4.5). The oscilloscope measures the outputs of the SPAD and the PD using a trigger. Employing this setup a picosecond burst of photons is send

out and the time difference is measured. This process is repeated multiple times, hereby characterising the statistical process. It is important to take the distance between the PD, SPAD and the splitter into account; in one oscilloscope cycle (25ps) light moves 7.5mm. The probe input capacitance is also important, this delays the measurement of the input pulse. The probe capacitances therefore should be measured beforehand, and probes with similar capacitance are chosen. The SPAD jitter is then calculated with the reference jitter of the PD.

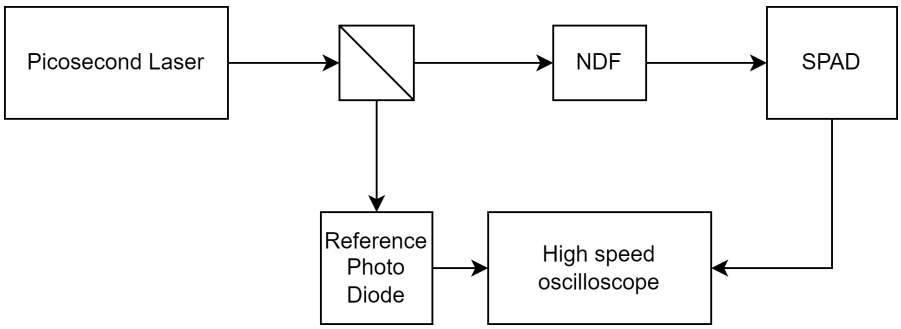


Figure 4.5: Time jitter measurement setup.

5

Readout

The purpose of the SPAD is to detect photons. When a photon hits the active area of the SPAD, a current spike will occur. This needs to be detected and processed accordingly to determine the DCR and PDP. This is the task of the readout system.

In the first section, the total system will be elaborated on. The sections after are each dedicated to a specific piece of the system, beginning with the analog to digital converter (ADC), following up with the FPGA and ending with the Arduino.

5.1. Total system

The overall goal of the readout system is to determine and communicate the counts per second continuously over a certain time period. An overview of the entire readout system is given in Figure 5.1. Several general system requirements can be defined:

- The readout system needs to detect the spikes given by the SPAD and quenching circuit.
- The readout system needs to be capable of 2 Mcounts/s.
- The readout system needs to have a minimum resolution of 1 kcounts/s.
- The readout system needs to operate at room temperature.
- The data should be continuously logged to a PC.

To be able to count the spikes given by the SPAD circuit, it is essential to first import the analogue signal to the digital domain. This is done with the ADC. This digital signal then should be counted. This data needs to be communicated to a PC, where the counts per second can be determined. For this it should be known over which time period the counts were measured.

A comparator is chosen as the ADC. This device is a simple but adequate solution. Another advantage is that they are commercially available as DIP packages, which is needed for the dot PCB on which the circuit is implemented. It also acts as an amplifier and can therefore directly be connected to the next stage when the right comparator is chosen. Another advantage is that the threshold voltage can easily be tuned to improve performance.

The counter is implemented on an FPGA. The main reason for using an FPGA is the flexibility and ease of designing and testing hardware. The FPGA can also handle quick signals, which is necessary for the rapid signal produced by the SPAD circuit and ADC.

An Arduino is used to facilitate the communication to the PC. The reason for not using direct FPGA to PC communication is because developing on FPGA takes a lot of time, mainly due to a lack of documentation and libraries. Communication would also need to be implemented from scratch, while Arduino uses a higher level design language where communication is very easy. The Arduino IDE also has a serial monitor, which makes logging data easy.

Because of its speed and ease of implementation, the Serial Peripheral Interface (SPI) protocol is used for communication between FPGA and Arduino. Arduino has existing libraries that make programming for SPI less challenging.

5.1.1. FPGA to Arduino communication

As mentioned before, the signal given by the ADC needs to be counted for pulses and communicated to a PC. The goal is to determine the count rate and therefore the counts need to be tallied over a known time period. This introduces some timing issues. The count rate needs to be determined continuously and thus the communication must be done in a smaller period of time than the time period of counting.

Another intertwined constraint is how long the time period of counting should be. The length of the counting period together with the maximum allowable count rate determines the minimum data size. The data size again is tied together with how long communication takes.

The FPGA is more precise than the Arduino and also has the ability to work in parallel, which the Arduino can not. Therefore, all the timing is done on the FPGA, which also means that it is the master in the communication. This way the FPGA can make sure that every data transmission is an amount of counts over a predetermined time period and thus it is not necessary to communicate the time. The Arduino is then set up to listen for communication and process it accordingly.

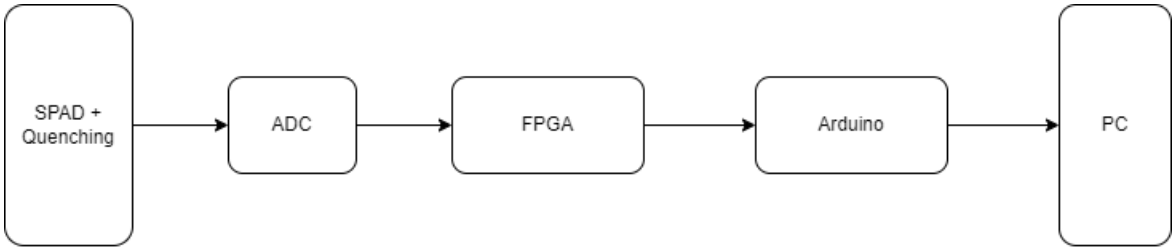


Figure 5.1: The overview of the entire readout system.

5.2. Analog to Digital Converter

A comparator is chosen as the ADC. It is a relatively simple device, but it should abide a few requirements to be adequate for the system. First, its propagation delay needs to be low enough to handle the quick voltage spikes from the SPAD. The SPAD current spikes rise last around 20 nanoseconds[10]. The propagation delay of the comparator should therefore be lower than 20 ns. Next to that, it should be able to detect the input voltages from the SPAD and quenching circuit. Roughly, the minimum SPAD current for a high input can be taken to be 0.4 milliampere[10], which results in a voltage of 1.9 V for a sense resistor of 4.7 kΩ. Also the input offset voltage needs to be negligible compared to this voltage. The input bias or leakage current should also be negligible compared to the SPAD avalanche current, which is minimally 0.1 mA.[10] Ideally, the high and low output voltages of the ADC should also be compatible with the I/O of the FPGA so no extra amplifier is needed.

Even though the requirement for a DIP package limits the amount of comparators available, the MAX903 from Maxim Integrated Products is a good fit for the project. This device has a propagation delay of 8 nanoseconds, input offset voltage of 1 mV, input bias current of 4 μA, output high voltage of 3.5 V and output low voltage of 0.3 V.[26] All these device characteristics should be adequate for the total system. The downside of this device is that it is very sensitive to parasitic capacitances and noise, which can be solved with some filtering. The proposed circuit can be seen in Figure 5.2. The threshold voltage is set by an external voltage source. The threshold is an important parameter in the system, since it decides the length of the output pulse. If placed too low, noise can pass the threshold and thus be detected as counts, which should be avoided.

Another option is a Schmitt Trigger inverter. These logic gates have the advantage of being more resilient to noisy signals, due to its negative and positive going threshold values. The downside of such a device is that it requires biasing and amplification to be compatible with the input signal. Because of this, a Schmitt Trigger inverter is harder to implement than the comparator. Therefore the preference is given to the comparator, but if the signal from the SPAD and quenching turns out to be more noisy than expected, a Schmitt Trigger inverter can be implemented.

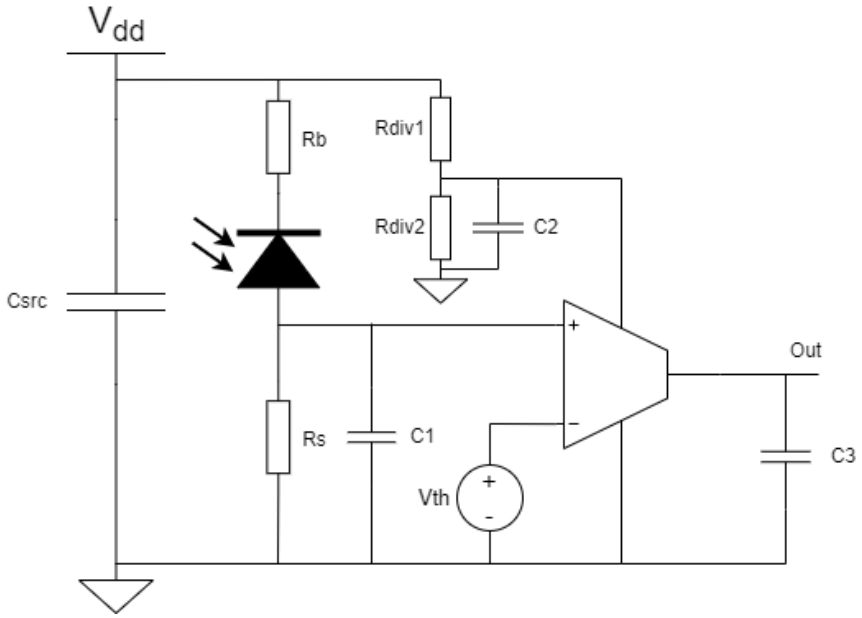


Figure 5.2: Schematic of the ADC testing setup. All capacitors are utilised to filter out noise.

5.3. FPGA

As mentioned before, the FPGA is responsible for counting the pulse signal and providing the master SPI communication. To tie these modules together, a controller is needed. A block diagram of the FPGA circuit is given in Figure 5.3. The FPGA available for this project is an Altera DE2-115 by TerasIC with the following important specs:

- 50 MHz clock.
- 3.3 V I/O pin.

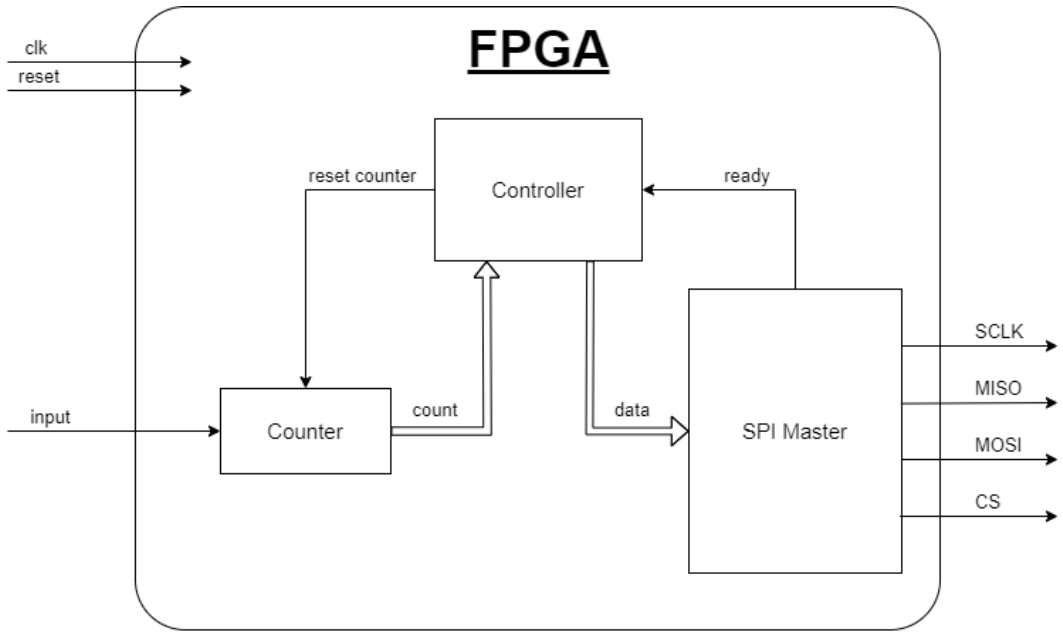


Figure 5.3: Block diagram of the FPGA inner workings. The arrows give a rough depiction of the signals that connect the modules. Note that every module is connected to the clock and reset signals. The thicker arrows represent a parallel bus.

The counter is a simple clocked module. It checks for rising edges of the input signal and increments a counter value. The counter value resets when the reset signal is set to high. The counter value is continuously fed to the controller.

The SPI master module takes care of the SPI communication. It takes in data signals and sends out the appropriate signals according to the SPI protocol. For the shared clock (SCLK) signal a frequency of 2 MHz is chosen, because this can be nicely generated with the 50 MHz FPGA clock and is a nice division of the 16 MHz of the Arduino Uno. The module also has a ready signal to notify when it is busy sending or is ready to start sending. The code for the SPI master component is taken from the GitHub of Jakub Cabal[9] according to the MIT license.

The controller takes care of the timing of the system. A simplified Finite State Machine (FSM) diagram is given in Figure 5.4. It has a clock cycle counter to keep track of time. The controller uses this to buffer and reset the counter when the predetermined counting period has passed. The buffered count is then passed through to the SPI master module. When the module has signaled that it is ready to communicate again, the controller goes back into the count state, which is reset during the communication phase, just like the timer.

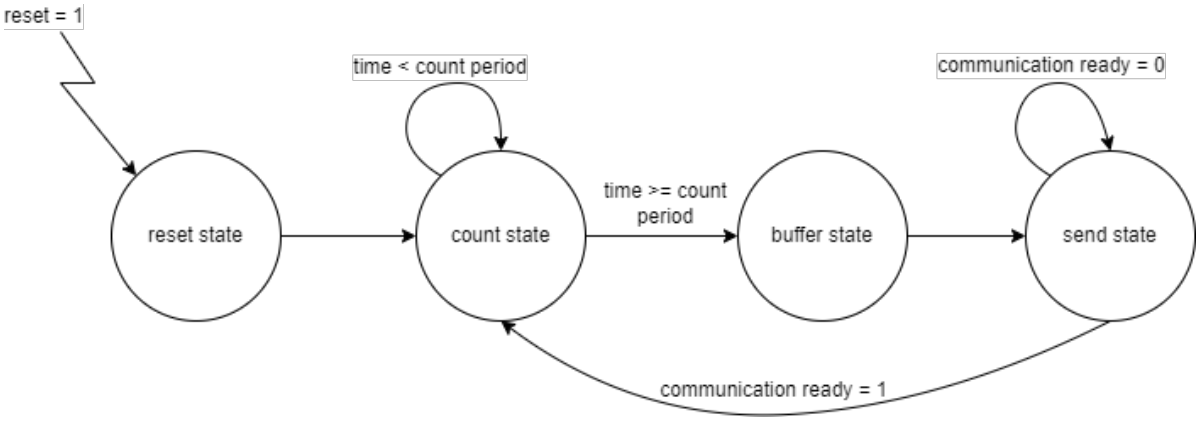


Figure 5.4: Simplified FSM of the controller unit on the FPGA.

5.4. Arduino

The main job of the Arduino is to get the data from the FPGA and log it to the PC. The Arduino used for this project was the Arduino Uno, with the following important specs:

- 16 MHz clock.
- Digital pins that are compatible with the DE2-115 FPGA I/O.
- SPI supported pins.

The Arduino is configured as a SPI slave device, awaiting data transmission from the FPGA through the Master Out Slave In (MOSI) line. An interrupt service routine can be set up so that every data packet can be logged to the serial monitor in the Arduino IDE. The Arduino is set up to receive data packets of one byte per transmission. It is possible to extend this to a 32 bit system. The FPGA then first sends a start byte of all ones. After that, each received byte is saved and when 4 bytes are received the data is processed and logged to the serial monitor.

The Arduino and FPGA are connected with male-to-female jumper wires. The ground pins of both boards are connected. The SPI output pins of the FPGA, which can be freely determined, are connected to the following Arduino SPI pins:

- Digital pin 13: SCLK (Shared Clock)
- Digital pin 12: MISO (Master Out Slave In)
- Digital pin 11: MOSI (Master In Slave Out)
- Digital pin 10: CS (Chip Select)

6

Results and Discussion

This chapter shows and discusses the results from the various tests and measurements that have been done. It will kick off with the results from the validation test and then the results from the read out.

6.1. Validation

First all the connections were checked to see if the pinout was correct. Simultaneously the ESD diodes and the SPADs were tested to see if they show diode behaviour as is described in Chapter 3. These tests showed that a current is flowing in forward bias and none in reverse bias (in the range -2.5V to 2.5V). In four cases the in- and output were switched compared to the documentation, besides that the pinout matched up. Also basic diode behaviour was seen on the ESD and SPADs (see Figure 6.1).

After it was confirmed that the safety diodes and the SPADs work from -2.5V to 2.5V the SPADs were tested in reverse bias to find their breakdown points. This turned out to be around -6V. This can be seen in Figure 6.1, where the preliminary I-V curve of SPAD H1 is displayed. Below roughly 7.0V, the resistor is the dominant factor in the circuit which results in the slower decline of the curve. A more precise I-V curve measured over a SPAD can be seen in Figure 6.2a, which has a breakdown voltage of -5.5V. As can be seen in Figure 6.2b the measured I-V curve does not match up completely with the simulation.

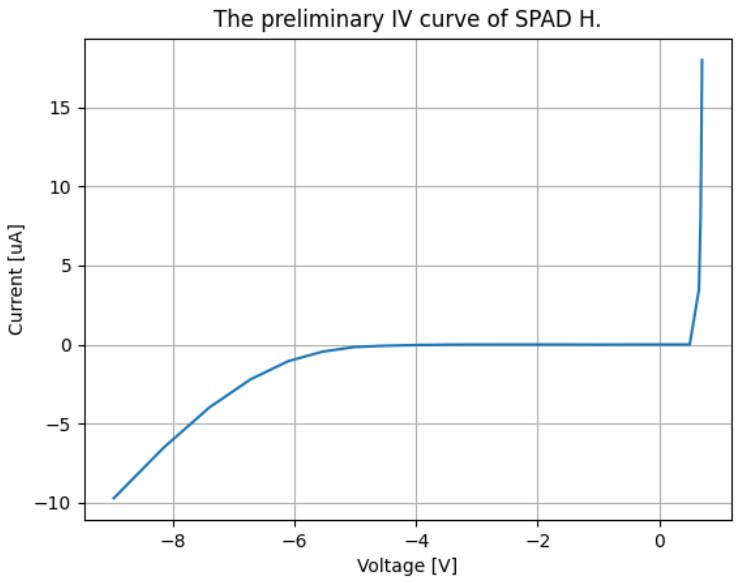
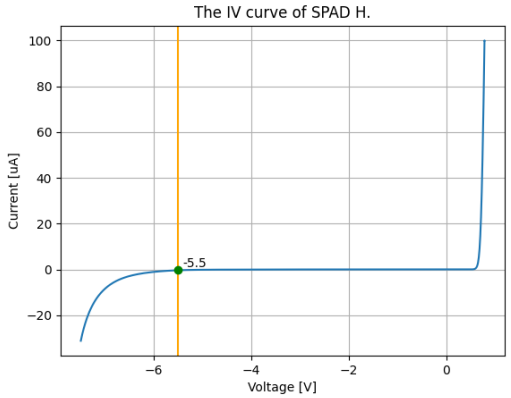


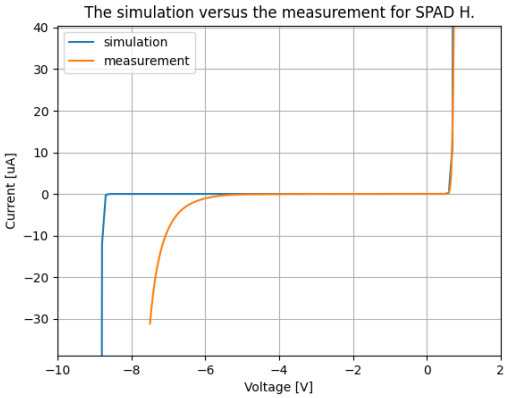
Figure 6.1: The preliminary I-V curve of SPAD H1.

6.1.1. I-V curve

Now the preliminary I-V curve is established, more precise measurements can be done. A more precise I-V curve measured over a SPAD can be seen in Figure 6.2a. The orange line indicates the breakdown voltage, which is -5.5V. This breakdown voltage is found to be the same for all measurements. As can be seen in Figure 6.2b the measured I-V curve does not match up completely with the simulation. This can be due to the difference in design of the diode utilised in the simulation and the SPAD design (Figure 3.4). A secondary explanation is the variation in breakdown voltages found in literature between different technologies (see Section 3.2).



(a) The I-V curve of SPAD H1 with a breakdown voltage of -5.5V.



(b) The I-V curve of SPAD H1, the simulation versus measurement.

Figure 6.2: The I-V curve of SPAD H1. Left is the curve showing the breakdown voltage and right is the measurement compared to the simulation.

After more careful examination it was found that a voltage sweep over two pins that are not connected via a SPAD yield the exact same I-V curve (see Section 6.1.2 for explanation). This can be seen in Figure 6.4a. This found curve is explained by the current flowing through the I/O ring and not the SPAD, see Figure 6.3.

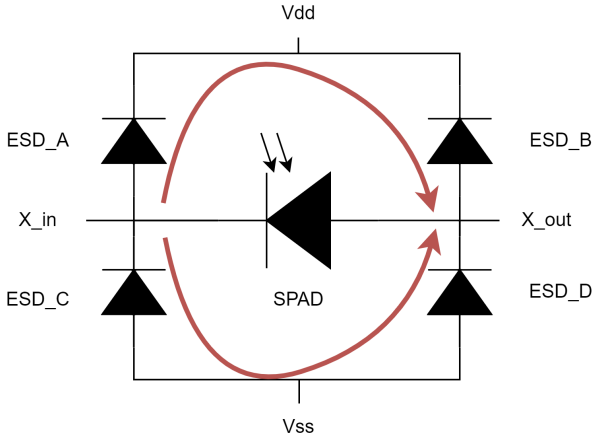
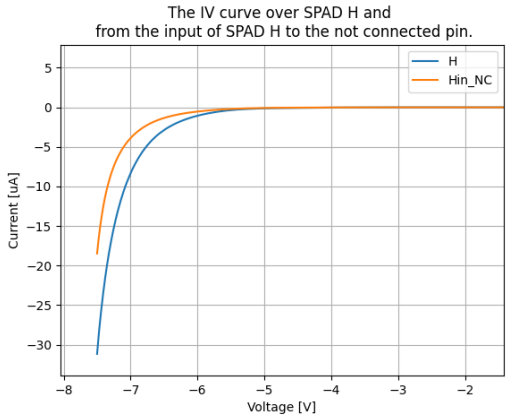


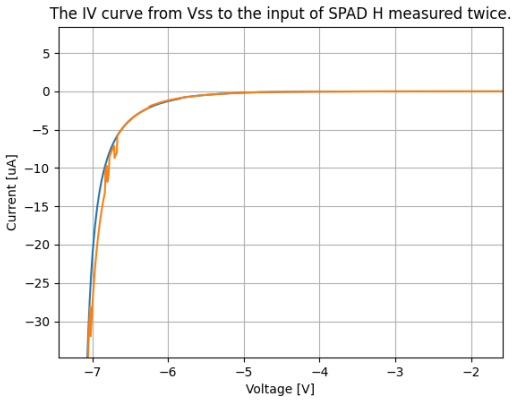
Figure 6.3: The current path taken on the chip when the voltage between X_in and X_out exceeds the breakdown voltage of the ESD diodes.

However, a distinguishable discrepancy in current exists between the measured current at the input and output of the SPAD, and the current measured between the output of the SPAD and the unconnected pin (see Figure 6.4a). This current difference amounts to approximately 4 µA at a reverse bias of 7.0V. Noteworthy differences in current are also observed when doing the same measurement twice, as depicted in Figure 6.4b. In this case, the current difference is approximately 6 µA at a reverse bias of 7.0V.

Additionally, the current difference between pads due to their placing on the chip, is evaluated by measuring various inputs and outputs, all connected to the Vdd pin. The results indicate significant variations in the I-V curve depending on the specific pin to which it is connected (see Figure 6.5). For example, the current difference between Vdd connected to the output of SPAD B or the input of SPAD D is 15 µA at -7.0V. Based on the aforementioned information, it can be deduced that the current difference between the input and output of SPAD H1, as well as the input of SPAD H1 and the unconnected pin, is attributable to both the measurement setup and the differences in wiring length across the chip.

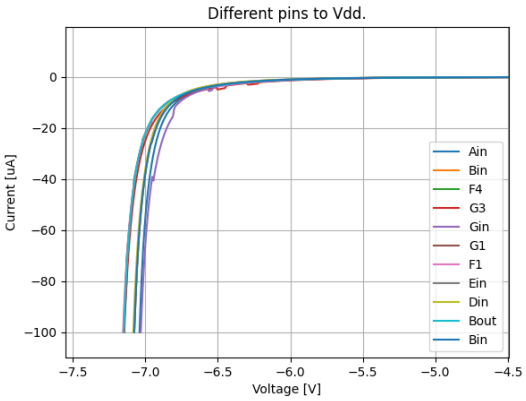


(a) The I-V curve of SPAD H1 and of the output of SPAD H1 to the not connected pin.

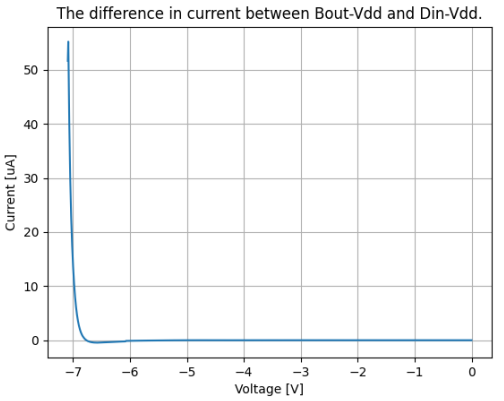


(b) The I-V curve from the Vss connected to the input of H1 measured twice.

Figure 6.4: The I-V curves relate to SPAD H1. Left shows the I-V curve of SPAD together with the curve from SPAD H1 to the not connected pin. Right shows an I-V curve that is measured twice to show the current difference due to the SMU.



(a) The spread of different in and output pins that are all connected to Vdd to see the difference in current.



(b) The difference in current measured between the Vdd and the input of SPAD D and Vdd and the output of the SPAD B.

Figure 6.5: The effect of pad placement on the chip is showed by plotting I-V curves that have one pin connected to the Vdd (Left). Right shows the current difference between two of those measurements.

6.1.2. Avalanche and Quenching

The quenching tests do not line up with the expected behaviour (see Figure 6.6). The results were often changing and did not show a predictable pattern. This test has revealed the chip does not work as intended.

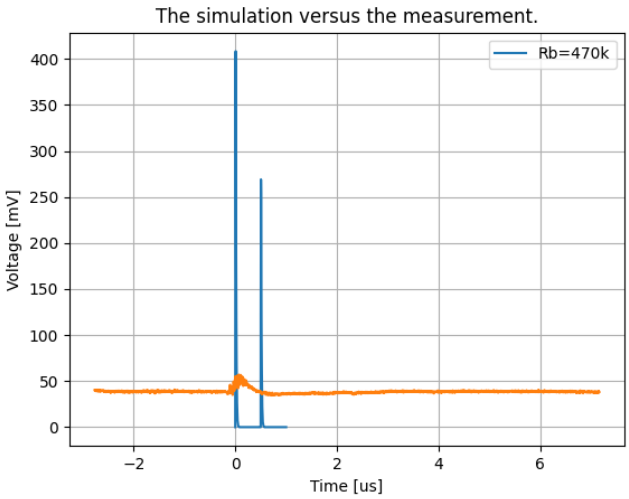


Figure 6.6: The simulation versus the measured dead time.

Additional test showed that the ESD diodes of the I/O ring are the cause of this problem (see Figure 6.3). The breakdown voltage of the ESD diodes is measured to be -5.5V (see Figure 6.5a). When 7.1V is applied from X_in to X_out in Figure 6.3, 6.4V is measured between Vdd and X_out and 0.7V between Vss and X_out. The red arrows show the path taken by the current through the I/O ring and not the SPAD (see Figure 6.3). There was no distinguishable difference found between the I-V curve over a SPAD and SPAD pin to NC (see Figure 6.4a), therefore it can be concluded that breakdown voltage of the SPADs is greater than 5.5V.

6.2. Characterisation

The validation stage of the chip was unsuccessful due to the ESD diodes having a lower breakdown voltage than the SPADs. As a result, characterisation measurements could not be performed. Potential solutions to overcome this obstacle carry high risks for the safety of the SPADs. One such solution involves expanding the range of the I-V curve to identify an additional break point associated with a SPAD. However, since the ESD diodes are past breakdown, increasing the voltage would lead to an exponential rise in current flowing through them. This current increase has the potential to cause burnout of the ESD diodes, with uncertain consequences and the potential to damage nearby components. Alternatively, scratching the chip to disconnect the ESD diodes is not a suitable approach, as the ESD diodes are situated directly beneath the I/O pads. Therefore it was impossible to do any characterisation measurements.

6.3. Readout

The readout system is tested in two parts, the ADC and the FPGA and Arduino. First the ADC results are discussed en next the FPGA and Arduino results are elaborated on.

6.3.1. Analog to Digital Converter

There is no way to test the ADC with a SPAD response, because of the I/O ring as mentioned before. Therefore a function generator is used for testing. The circuit as shown in Figure 6.7 is created. The test signal from the Tektronix AFG3021C function generator is set to be 20 ns wide pulse with a low voltage of 0 V and high voltage of 0.4 V, which is lower than expected from the SPAD and quenching circuit. For simplicity a square wave is taken. The positive input terminal and output terminal signals of the ADC are measured on the Tektronix TDS2022C oscilloscope and are given in Figure 6.8. The input signal is not as expected, because the speed of the signal is near the limit of the function generator. The threshold voltage of the comparator is set to be 100 mV, which is low enough for the comparator to be triggered. The rise time of the output seems to be higher than expected, which is likely due to the

stray capacitances in the circuit. The same can be said about the fall time. When the high output period of the test signal is increased, it is shown (Figure A.12) that the output reaches the high output voltage of roughly 3.5 V as expected. The comparator seems to be sufficient when the stray capacitances can be lowered.

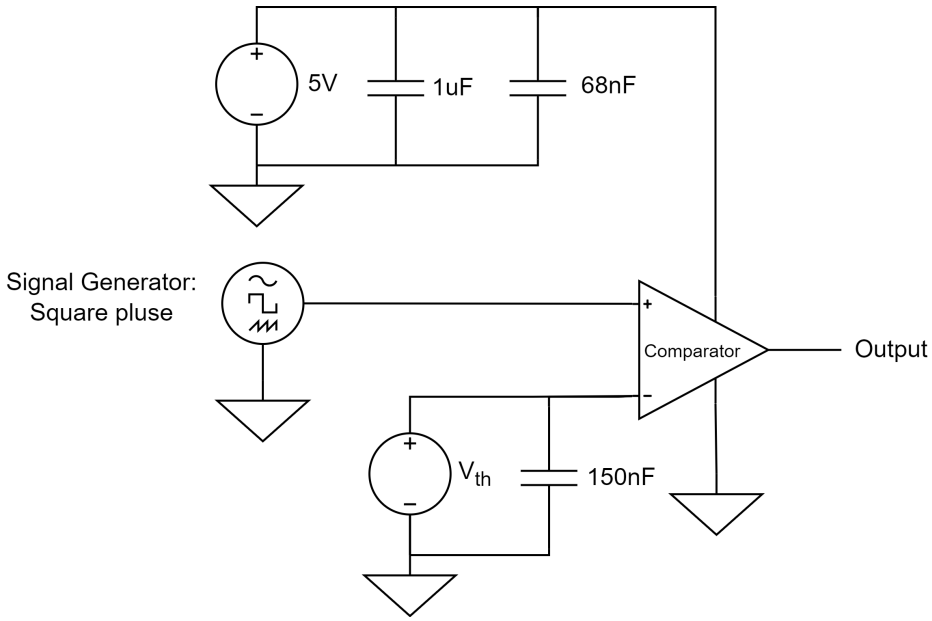


Figure 6.7: Schematic of the ADC testing setup. All capacitors are utilised to filter out noise.

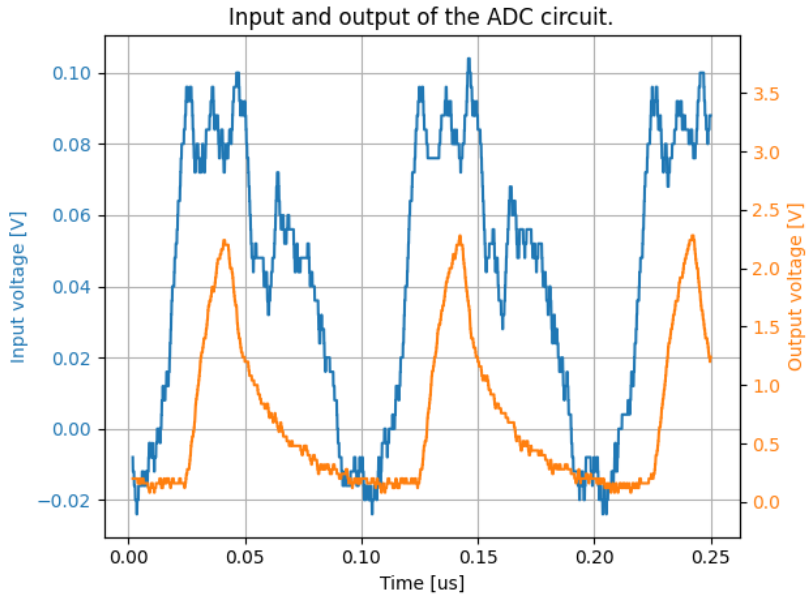


Figure 6.8: The in and output of the ADC with a test signal. The test signal has a frequency of 10MHz and a duty cycle of 20%. It ranges from 0 to 0.4 V. The threshold of the ADC is set to 0.1 V.

6.3.2. FPGA and Arduino

The system was first set up with one byte of data per transmission, which results in the counter consisting of 8 bits. Because of this, the counter can count up to 256 counts. Using the maximum count rate of 2 Mcounts/s, the maximum measure period is 128 μs. The maximum resolution for this system is roughly 7.8 kcounts/s. This can be improved with a longer measure period, but that decreases the maximum count rate. The system is also tested for a measure period of 2 ms, since the dark count is likely way lower than 2 Mcounts/s, but in the range of a few to a few tens of kcounts/s. This measure period has a maximum count rate of 128 kcounts/s and a resolution of 500 counts/s, which is better suited for dark count measurements.

To test the system, a pulse is generated on the FPGA and directly connected to the entire system. The counts measured were logged and later converted to counts per second. The mean and standard deviation were calculated and noted down. The results of these tests are given in Table 6.1. It can be seen that the standard deviation is mostly because of the limited resolution. For the 2 ms period this error is also generally low, with an exception of the 25 kcounts/s tested signal. This is likely due to the fact that a bit is wrongly communicated. When looking at the counts for this test, it can be seen that the error is almost constantly 16 counts, which directly corresponds to the fifth least significant bit. The same logic holds for the other tests with a higher error than the resolution.

With an 8 bit system, there is no way of getting a maximum count rate of 2 Mcounts/s and a minimum resolution of 1 kcounts/s. A solution to this problem is to increase the system to a 32 bit sytem. This way the counter can count up to $2^{32} \approx 4.3$Gcounts. To get the maximum resolution with a maximum count rate of 2 Mcounts/s the measure period should be approximately 2150 seconds, which is unacceptably long. With a measure period of 10 ms 2 Mcounts/s can easily be measured with a resolution of 100 count/s.

However, this system was more difficult to realize. The Arduino has little to no support for communicating words of 32 bits over SPI and thus it has to split up into 4 bytes. This method was tried out and gave good simulation results but bad results when realized. The communication was not continuous and the received data was often nonsense. Further tests are needed to assess the source of the problem.

All in all, the system is found to either be fit for the maximum count rate of 2 Mcounts/s or an accuracy of 1 kcounts/s. If the system is successfully expanded to 32 bits, both requirements can be met. However, the requirements have quite some margin. The PDP measurements can likely also be done with a lower maximum count rate and thus better resolution. The maximum count rate of 2 Mcounts/s is a rough estimation with high safety margins. When doing PDP measurements, the maximum count rate can be determined more accurately, which is likely to be a lot lower than 2 Mcounts/s. The measure period can be changed so that the maximum count rate is closer to the measured maximum count rate. If this rate is indeed lower, a better resolution is obtained. For dark count measurements the system with a measure period of 2ms is sufficient. To increase accuracy, the communication errors can be fixed. Another downside is that there is no automatic way of saving the data and processing it immediately. Right now the data is manually copied over from the Arduino IDE serial monitor to a csv file and then processed with a Python script. This can be improved upon by automating the process.

Table 6.1: Table of the readout results. For each test, the system was run for a few seconds. The data was converted to counts per second and the mean and standard deviation was taken. Each column is for a different measure period noted at the top of the column. Note that cps stands for counts per second.

Tested signal	**128 us**	**2 ms**
0.5 kcps	-	707 ±246 cps
1 kcps	-	1 ±0 kcps
2 kcps	-	2 ±0 kcps
5 kcps	-	5 ±0 kcps
10 kcps	13.9 ±3.2 kcps	-
12.5 kcps	-	12.4 ±0.49 kcps
25 kcps	2.3 ±3.6 kcps	17.2 ±0.7 kcps
50 kcps	53.9 ±2.3 kcps	50.2 ±0.4 kcps
100 kcps	-	100.2 ±2.9 kcps
119 kcps	-	119.2 ±0.2 kcps
125 kcps	128.9 ±3.9 kcps	-
250 kcps	253.9 ±3.9 kcps	-
500 kcps	503.9 ±3.9 kcps	-
794 kcps	46 ±2.5 kcps	-
1 Mcps	1.00 ±0.01 Mcps	-
1.5 Mcps	1.512 ±0.00 Mcps	-

7

Conclusion

In conclusion, the ESD diodes were found to be functioning correctly, and the pinout of the chip was verified. The SPADs exhibited expected behavior within the voltage range of -4V to 1V. However, beyond this range, the chip limitations, specifically the breakdown of the ESD diodes before the SPADs, prevented further investigation of SPAD behavior. Consequently, an I-V curve was measured, but quenching measurements and additional tests showed that the chip does not work as intended. Therefore the measured I-V curve was not the I-V curve of the SPAD. The presence of the unwanted current path caused by the ESD diodes, led to the omission of characterisation measurements. With the additional done measurements, it can be concluded that breakdown voltage of the SPADs is greater than 5.5V. It can also be concluded that the FPGA and Arduino part of the readout are suitable for dark count measurements. For PDP not all requirements are met, however it is likely that the system will still be adequate to obtain reasonable measurements. The ADC was not tested with the SPAD and quenching circuit, therefore a conclusion on the total system cannot be made. If the stray capacitances can be lowered, the system would be sufficient for the intended purpose.

Based on this research, the following recommendations are proposed:
In future designs, the ESD diodes in the I/O ring should be either removed or disconnected to prevent the issues encountered with the current circuit (Figure 6.3). However, it should be noted that eliminating the safety diodes introduces the risk of electrostatic discharge damage to the SPAD.

To minimize parasitic capacitances resulting from the socket and PCB, a custom-designed PCB can be employed. By directly gluing the chip onto the PCB, the need for a socket and package, along with their associated capacitances, can be eliminated.

The custom PCB should provide convenient connections for direct measurement of the SPADs for the I-V curve. Additionally, it should incorporate path options for the SPAD, including a standalone SPAD, a quenching circuit without ADC, and a quenching circuit with ADC. By incorporating breakout points along the path, these options can facilitate easy execution of validation and characterisation tests. Furthermore, integrating the quenching circuit and ADC on the same PCB would simplify the measurement setup. A custom PCB would also allow for a better comparator, since it does not have to be constrained to a DIP package.

It is still recommended to increase the FPGA and Arduino system to 32 bits and look into the current communication errors. The communication can also be improved by communicating directly between FPGA and PC, so sources for errors are reduced. Also the processing of the data can be automated. Ideally, the microwave frequency for the ODMR measurement can also be fed to the PC, so the ODMR spectrum can be directly plotted.

Taking into account the aforementioned recommendations, the following future work is proposed: redesign and fabricate the chip without the ESD diodes, replicate the steps performed in this thesis, including chip validation and characterisation of different SPAD designs. An additional characteristic that should be researched and measured is afterpulsing. Subsequently, improve the readout system and finally select the optimal design and couple it with the NV center to further advance the research.

A

Appendices

Here the chip documentation can be found and more graphs that could help understand the results a bit more.

A.1. Chip documentation

Figure A.1 shows the chip layout and Figure A.2 shows the chip with the active area size. Figure A.3 shows the pinout of the DIP package that the chip is wirebonded onto.

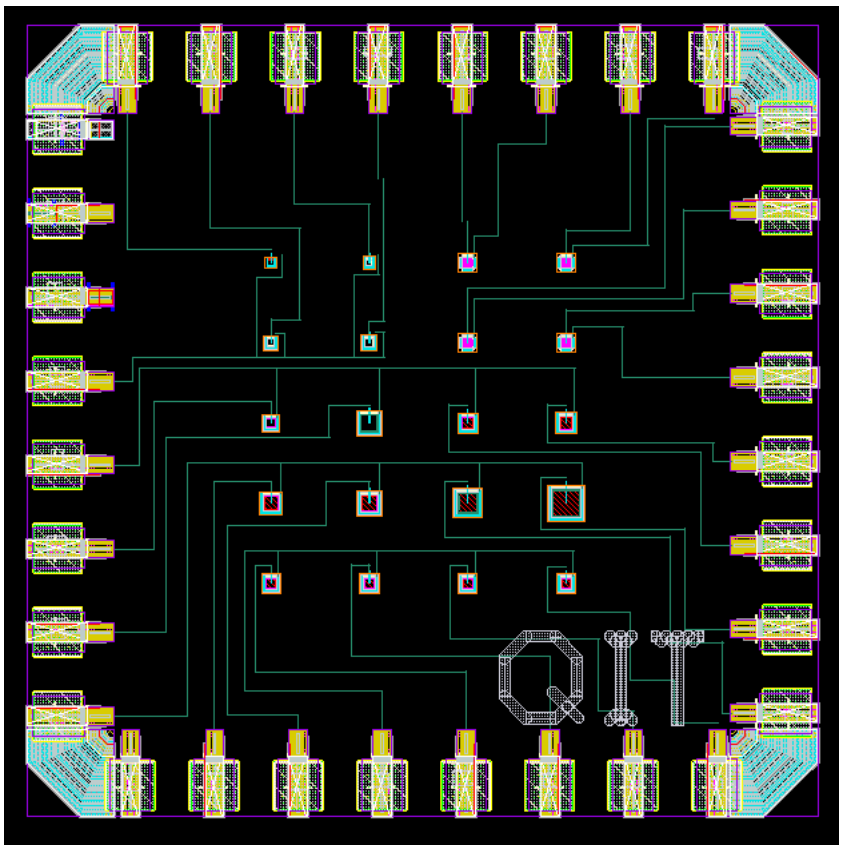


Figure A.1: The layout of the chip. The three pins in the top left are Vdd, Vss connected to the substrate and Vss starting at the top. The pin in the bottom left corner is not connected. All the other pins are in or outputs connected to SPADs.

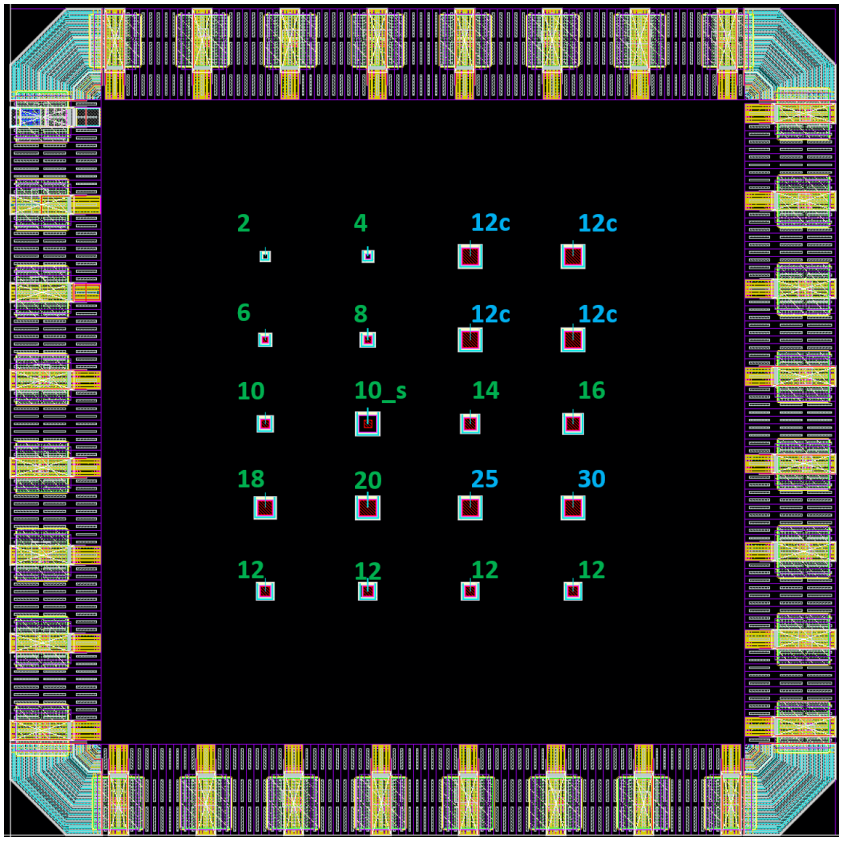


Figure A.2: The chip with the active area AxA um shown next to the SPAD. The 'c' next to some SPADs mean it is circular and the '_s' after one SPAD means it has extra metal shielding around it.

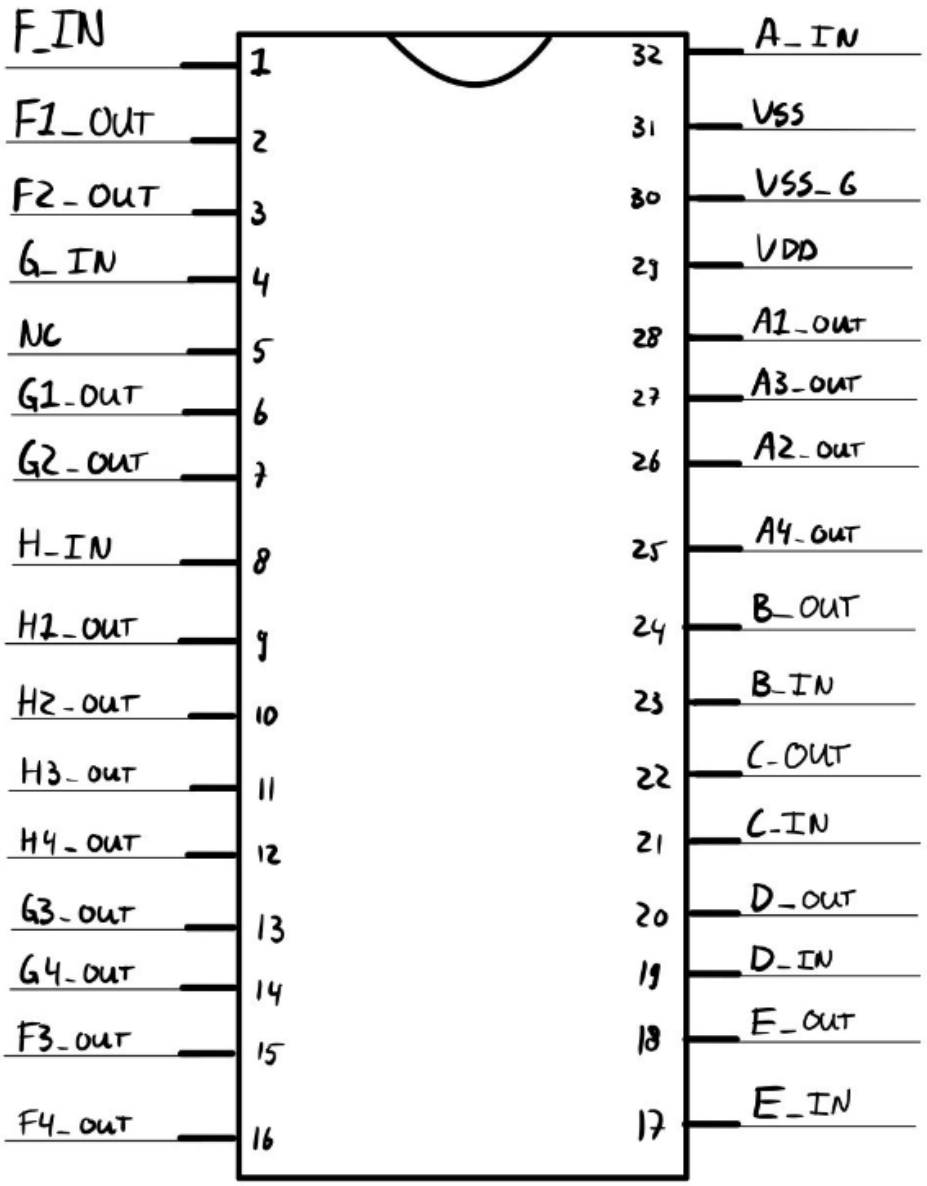


Figure A.3: The pinout of the DIP package after the chip is wirebonded to the DIP. The in-output is in reverse bias. NC = Not Connected. VSS_G is the VSS that is connected to the substrate of the chip.

A.2. Validation

Here the measurements setups are shown for various measurements as well as some additional results.

Measurement setups

Down below the measurement setups that were used for the various experiments are shown.

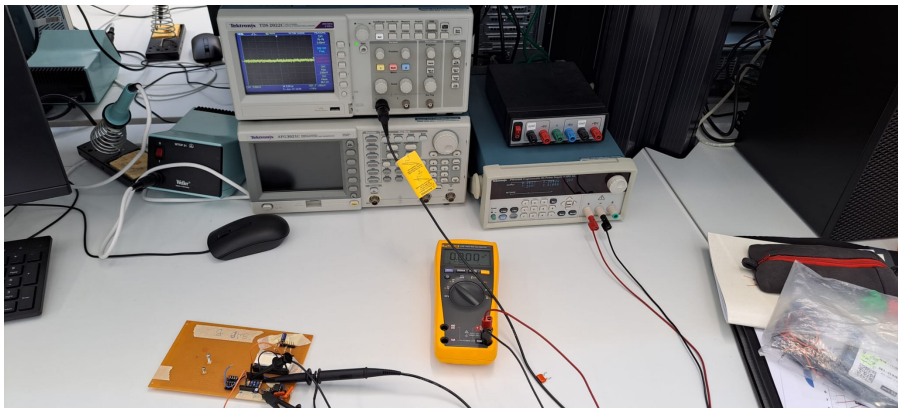


Figure A.4: The measurement setup including the power supply, the oscilloscope and the multi-meter used for the preliminary I-V curve.

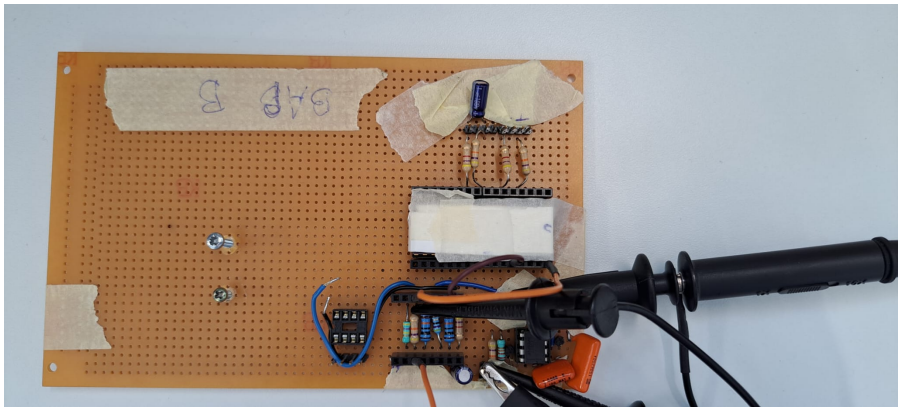


Figure A.5: This is a close-up of the PCB that is used for the measurements. The white box with headers next to it is the DIP package with the chip on top. Furthermore, there are resistors and capacitors to make the necessary measurement setups.

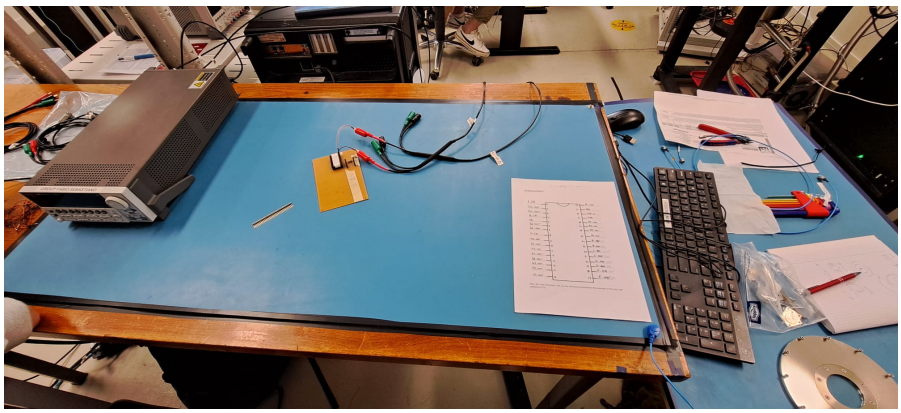


Figure A.6: The measurement setup for measuring the more elaborate I-V curves.

I-V curves

Additional I-V curves for the SPADs can be found here. Figure A.7 shows the preliminary IV curves for a select amount of SPADs. Figure A.8 shows the IV curves of all the SPADS, as can be seen these curves match each other closely. Figure A.9 shows the difference between the two different SMU's that were used. These curves are almost identical, and so the difference is negligible.

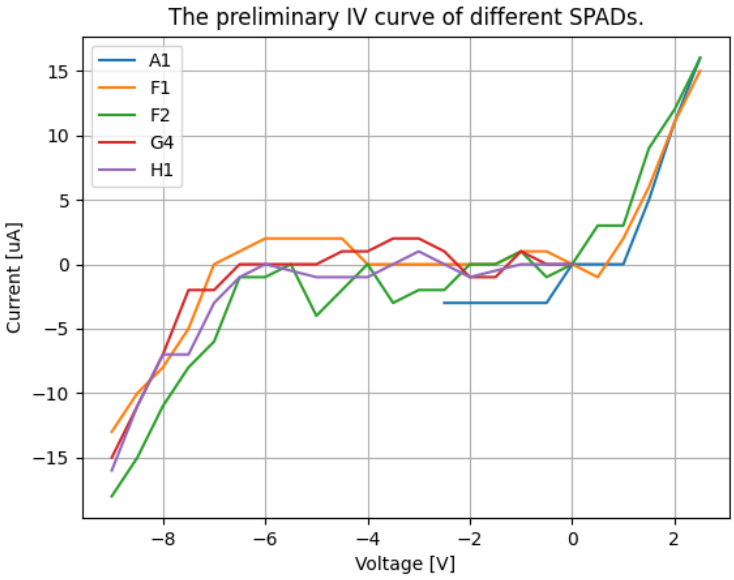


Figure A.7: The preliminary I-V curves for some SPADs.

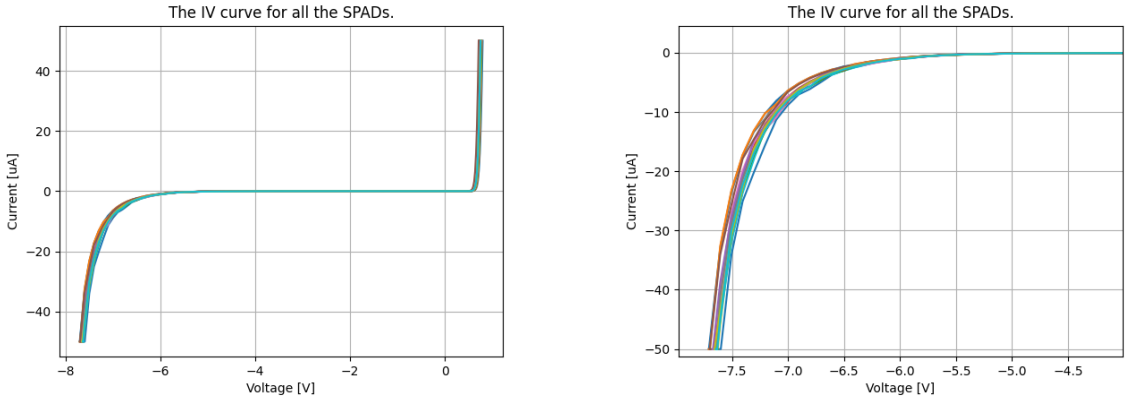


Figure A.8: The I-V curves of all the SPADs. On the left the whole graph can be seen and on the right the graph is zoomed in around the breakdown voltage. As can be seen the I-V curves match closely.

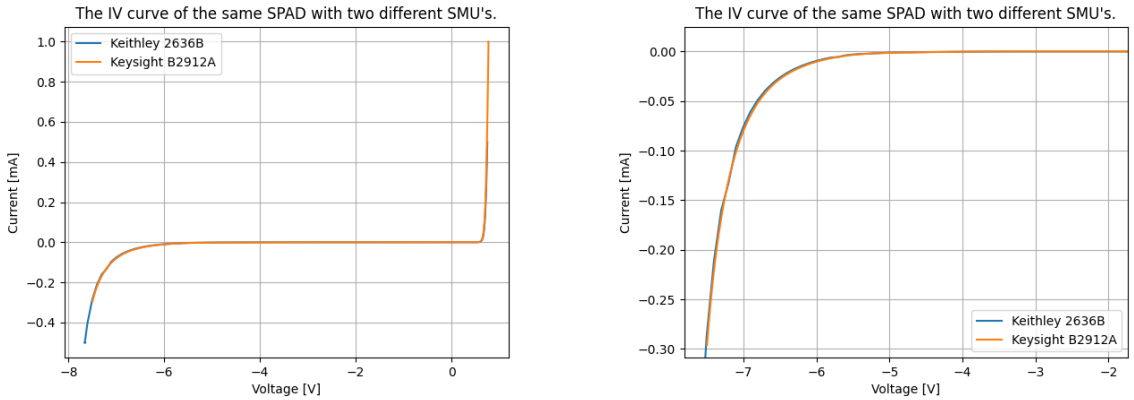


Figure A.9: The difference in I-V curves between the two different SMU's that were used. Left is the full I-V curve and right is zoomed in around the breakdown voltage.

A.3. Readout

ADC

Additional measurements for the ADC are shown here. In the figures below (Figure A.10, Figure A.11, Figure A.12 and Figure A.13) the results are shown. The blue signal is the input signal and the orange graph is the output signal.

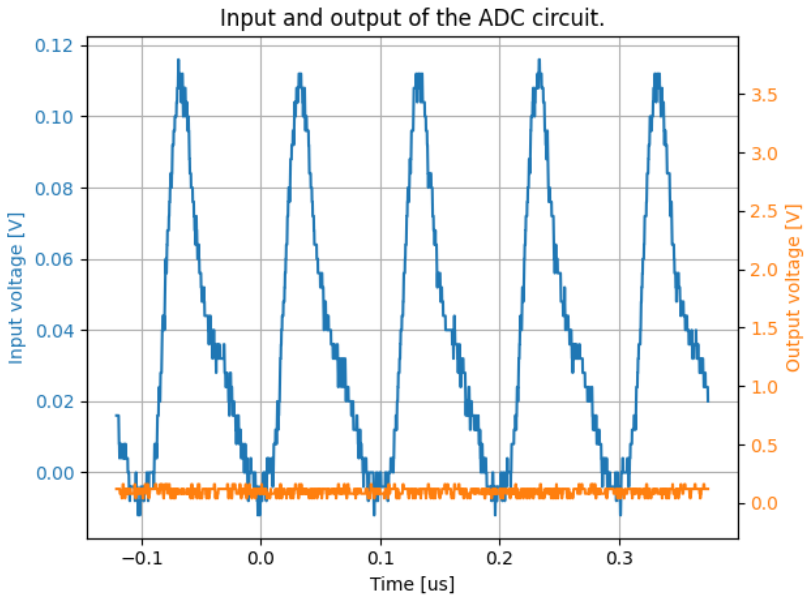


Figure A.10: The in and output of the ADC with a test signal. The test signal has a frequency of 10MHz and a duty cycle of 20%. It ranges from 0 to 0.4 V. The threshold of the ADC is set to 0.2 V.

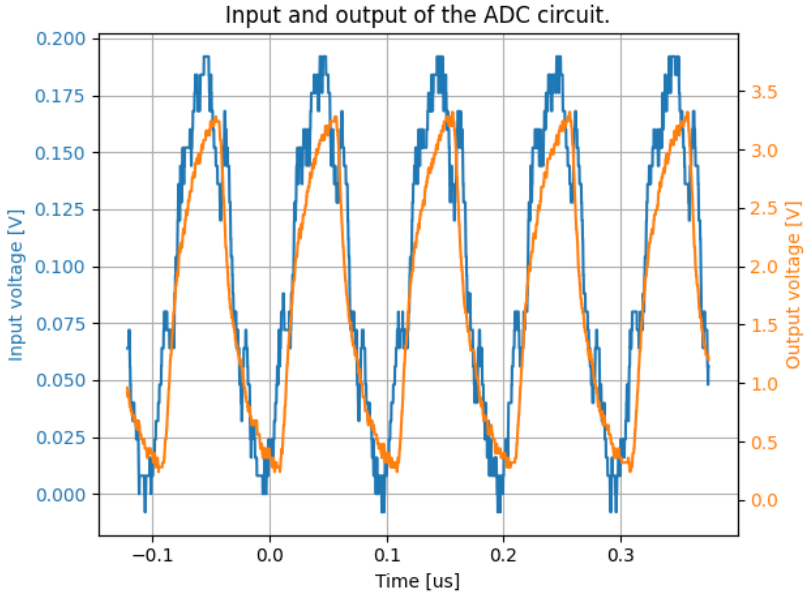


Figure A.11: The in and output of the ADC with a test signal. The test signal has a frequency of 10MHz and a duty cycle of 50%. It ranges from 0 to 0.4 V. The threshold of the ADC is set to 0.1 V.

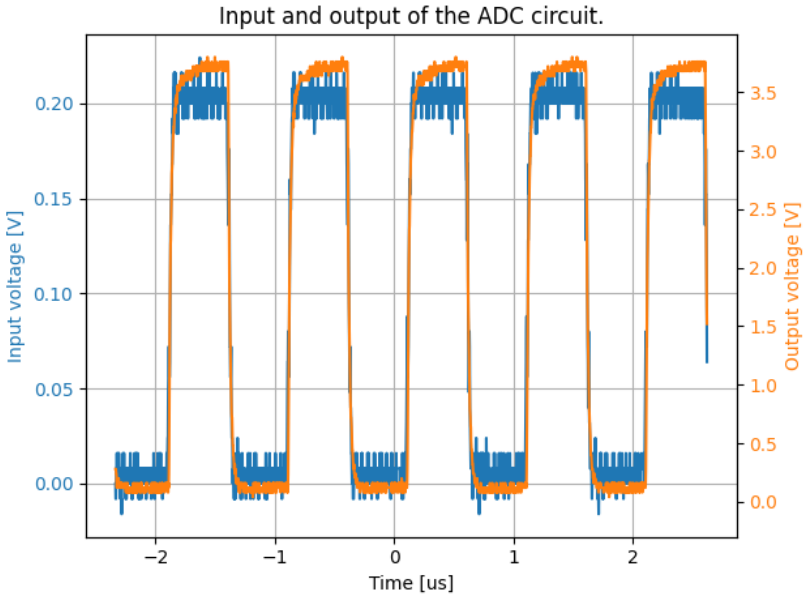


Figure A.12: The in and output of the ADC with a test signal. The test signal has a frequency of 1MHz and a duty cycle of 50%. It ranges from 0 to 0.4 V. The threshold of the ADC is set to 0.1 V.

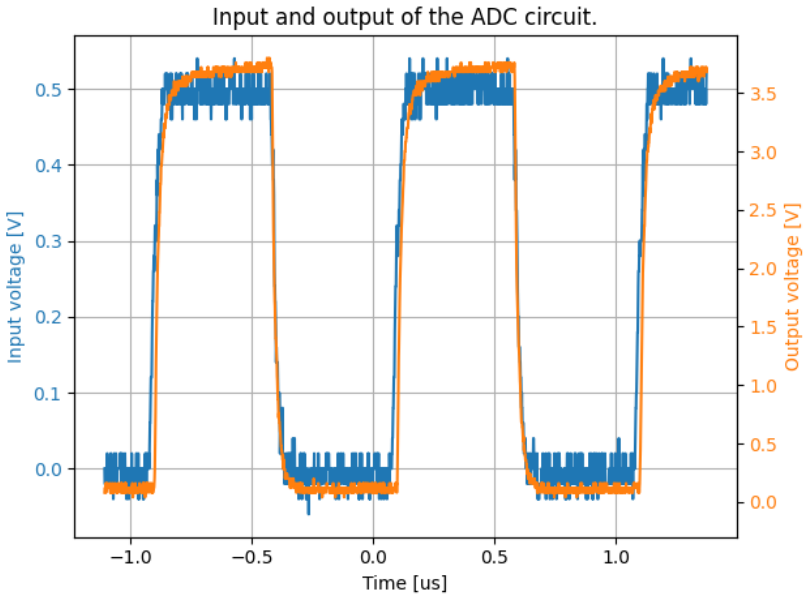


Figure A.13: The in and output of the ADC with a test signal. The test signal has a frequency of 1MHz and a duty cycle of 50%. It ranges from 0 to 1.0 V. The threshold of the ADC is set to 0.4 V.

Readout code

Here the code for the FPGA for the readout is displayed. First the file for the top level entity is given in the file `top_level_spi.vhd`. This contains the controller (shown in `controller.vhd`), input counter (shown in `input_counter.vhd`) and SPI master module (shown in `spi_master.vhd`). The test signal generator module is given in the file `input_generator.vhd`. The Arduino code for the 8 bit system is given in `readout_SPI.ino`. The data from the serial monitor of the Arduino IDE can be copied to a csv file with the filename holding the format of `[amount of bits]bits_[measure period]_[tested counts per second].csv`. This file can be read and processed by the Python script as given in `data_processing.py`.

top_level_spi.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.ALL;
3  use IEEE.MATH_REAL.ALL;
4
5
6  entity top_level_spi is
7          port(
8                  clk     : in    std_logic;
9                  reset   : in    std_logic;
10
11                 input   : in    std_logic;
12
13                 SCLK    : out   std_logic; -- SPI clock
14                 CS_N    : out   std_logic_vector(0 downto 0); -- SPI chip
                       select, active in low
15                 MOSI    : out   std_logic; -- SPI serial data from master
                       to slave
16                 MISO    : in std_logic -- SPI serial data from slave to
                       master
17         );
18 end top_level_spi;
19
20 architecture behaviour of top_level_spi is
21
22 component input_counter is
23         port(   clk             :       in      std_logic;
24                 reset           :       in      std_logic;
25
26                 input           :       in      std_logic;
27
28                 count_out       :       out     std_logic_vector(7 downto
                       0)
29         );
30 end component;
31
32 component SPI_MASTER is
33     Generic (
34         CLK_FREQ    : natural := 50e6; -- set system clock frequency in Hz
35         SCLK_FREQ   : natural := 2e6;  -- set SPI clock frequency in Hz (
               condition: SCLK_FREQ <= CLK_FREQ/10)
36         WORD_SIZE   : natural := 8;    -- size of transfer word in bits,
               must be power of two
37         SLAVE_COUNT : natural := 1     -- count of SPI slaves
38     );
39     Port (
```

```
40          CLK       : in  std_logic; -- system clock
41          RST       : in  std_logic; -- high active synchronous reset
42          -- SPI MASTER INTERFACE
43          SCLK      : out std_logic; -- SPI clock
44          CS_N      : out std_logic_vector(SLAVE_COUNT-1 downto 0); -- SPI
                chip select, active in low
45          MOSI      : out std_logic; -- SPI serial data from master to slave
46          MISO      : in  std_logic; -- SPI serial data from slave to master
47          -- INPUT USER INTERFACE
48          DIN       : in  std_logic_vector(WORD_SIZE-1 downto 0); -- data for
                 transmission to SPI slave
49          DIN_ADDR : in  std_logic_vector(natural(ceil(log2(real(SLAVE_COUNT
                ))))-1 downto 0); -- SPI slave address
50          DIN_LAST : in  std_logic; -- when DIN_LAST = 1, last data word,
                after transmit will be asserted CS_N
51          DIN_VLD  : in  std_logic; -- when DIN_VLD = 1, data for
                transmission are valid
52          DIN_RDY  : out std_logic; -- when DIN_RDY = 1, SPI master is ready
                 to accept valid data for transmission
53          -- OUTPUT USER INTERFACE
54          DOUT      : out std_logic_vector(WORD_SIZE-1 downto 0); -- received
                 data from SPI slave
55          DOUT_VLD : out std_logic  -- when DOUT_VLD = 1, received data are
                valid
56      );
57  end component;
58
59  component controller is
60          Generic (
61                  WORD_SIZE : natural := 8; -- size of transfer word in bits
                        , must be power of two
62                          SLAVE_COUNT : natural := 1      -- count of SPI
                                slaves
63          );
64          port(   clk             :       in      std_logic;
65                  reset           :       in      std_logic;
66
67                  count           :       in      std_logic_vector(7 downto
                        0);
68
69                  reset_counters  :       out     std_logic;
70
71                  DIN       : buffer  std_logic_vector(WORD_SIZE-1 downto 0);
                         -- data for transmission to SPI slave
72          DIN_ADDR : out  std_logic_vector(natural(ceil(log2(real(
                SLAVE_COUNT))))-1 downto 0); -- SPI slave address
73          DIN_LAST : out  std_logic; -- when DIN_LAST = 1, last data word,
                after transmit will be asserted CS_N
74          DIN_VLD  : out  std_logic; -- when DIN_VLD = 1, data for
                transmission are valid
75                  DIN_RDY  : in std_logic -- when DIN_RDY = 1, SPI master is
                         ready to accept valid data for transmission
76          );
77  end component;
78
79
```

```
80 constant CLK_FREQ    : natural := 50e6; -- set system clock frequency in
       Hz
81 constant SCLK_FREQ   : natural := 2e6;  -- set SPI clock frequency in Hz (
       condition: SCLK_FREQ <= CLK_FREQ/10)
82 constant WORD_SIZE   : natural := 8;    -- size of transfer word in bits,
       must be power of two
83 constant SLAVE_COUNT : natural := 1;     -- count of SPI slaves
84
85 signal count    : std_logic_vector(7 downto 0);
86 signal reset_counters_in, reset_counters_controller     :       std_logic;
87 signal DIN_ADDR :  std_logic_vector(natural(ceil(log2(real(SLAVE_COUNT))))
       -1 downto 0); -- SPI slave address
88 signal DIN, DOUT        : std_logic_vector(WORD_SIZE-1 downto 0);
89 signal DIN_VLD, DIN_RDY, DOUT_VLD, DIN_LAST     : std_logic;
90
91 begin
92
93 IC: input_counter port map(clk, reset_counters_in, input, count);
94 CTR: controller generic map(WORD_SIZE, SLAVE_COUNT) port map(clk, reset,
       count, reset_counters_controller, DIN, DIN_ADDR, DIN_LAST, DIN_VLD,
       DIN_RDY);
95 SPI: SPI_MASTER generic map(CLK_FREQ, SCLK_FREQ, WORD_SIZE, SLAVE_COUNT)
       port map(clk, reset, SCLK, CS_N, MOSI, MISO, DIN, DIN_ADDR, DIN_LAST,
       DIN_VLD, DIN_RDY, DOUT, DOUT_VLD);
96
97 reset_counters_in <= reset or reset_counters_controller;
98
99 end architecture behaviour;
```

controller.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.ALL;
3  use IEEE.numeric_std.all;
4  use IEEE.MATH_REAL.ALL;
5
6  entity controller is
7          Generic (
8                  WORD_SIZE : natural := 8; -- size of transfer word in bits
                       , must be power of two
9                          SLAVE_COUNT : natural := 1     -- count of SPI
                               slaves
10         );
11         port(   clk              :       in      std_logic;
12                 reset            :       in      std_logic;
13
14                 count            :       in      std_logic_vector(7 downto
                       0);
15
16                 reset_counters   :       out     std_logic;
17
18                 DIN      : buffer  std_logic_vector(WORD_SIZE-1 downto 0);
                           -- data for transmission to SPI slave
19         DIN_ADDR : out  std_logic_vector(natural(ceil(log2(real(
               SLAVE_COUNT))))-1 downto 0); -- SPI slave address
20         DIN_LAST : out  std_logic; -- when DIN_LAST = 1, last data word,
               after transmit will be asserted CS_N
21         DIN_VLD  : out  std_logic; -- when DIN_VLD = 1, data for
               transmission are valid
22                 DIN_RDY  : in std_logic -- when DIN_RDY = 1, SPI master is
                       ready to accept valid data for transmission
23         );
24 end controller;
25
26 architecture behaviour of controller is
27
28 type controller_state is (reset_state, count_state, buffer_state,
       send_and_reset_state);
29
30 signal state, new_state : controller_state;
31
32 constant freq           : integer       := 50; -- frequency in MHz
33 constant s              : integer       := 1000000 * freq; --clock cycles
       in one second
34 constant ms             : integer       := 1000 * freq; -- clock cycles in
        one milli second
35 constant us             : integer       := 1 * freq;-- clock cycles in one
        micro second
36
37 signal clk_count, new_clk_count          : unsigned(31 downto 0);
38 signal reset_clk_count                   : std_logic;
39
40 begin
41
42         process (clk)
43         begin
```

```
44              if (rising_edge(clk)) then
45                      if (reset = '1') then
46                              state <= reset_state;
47                              clk_count <= to_unsigned(0, 32);
48                      else
49                              state <= new_state;
50                              if (reset_clk_count = '0') then
51                                      clk_count <= new_clk_count;
52                              else
53                                      clk_count <= to_unsigned(0, 32);
54                              end if;
55                      end if;
56              end if;
57      end process;
58
59      process (state, clk_count, DIN_RDY)
60
61      variable new_DIN        : std_logic_vector(WORD_SIZE-1 downto 0);
62
63      begin
64              case state is
65                      when reset_state =>
66                              new_DIN := (others => '0');
67                              DIN_VLD <= '0';
68                              reset_counters <= '1';
69                              reset_clk_count <= '1';
70                              new_state <= count_state;
71                      when count_state =>
72                              new_DIN := DIN;
73                              DIN_VLD <= '0';
74                              reset_counters <= '0';
75                              reset_clk_count <= '0';
76                              if (clk_count > 128 * us - 2) then --
                                   minus two to take care of the offset
                                   caused by the FSM
77                                      new_state <= buffer_state;
78                              else
79                                      new_state <= count_state;
80                              end if;
81                      when buffer_state =>
82                              new_DIN := count;
83                              DIN_VLD <= '0';
84                              reset_counters <= '0';
85                              reset_clk_count <= '1';
86                              new_state <= send_and_reset_state;
87                      when send_and_reset_state =>
88                              new_DIN := DIN;
89                              DIN_VLD <= '1';
90                              reset_counters <= '1';
91                              reset_clk_count <= '1';
92                              if (DIN_RDY = '1') then
93                                      new_state <= count_state;
94                              else
95                                      new_state <= send_and_reset_state;
96                              end if;
97              end case;
```

```
98                  DIN <= new_DIN;
99          end process;
100
101         process (clk_count)
102         begin
103                 new_clk_count <= clk_count + 1;
104         end process;
105
106         DIN_ADDR <= (others => '0');
107         DIN_LAST <= '1';
108
109 end architecture behaviour;
```

input_counter.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.ALL;
3  use IEEE.numeric_std.all;
4
5  entity input_counter is
6          port(   clk             :       in      std_logic;
7                  reset           :       in      std_logic;
8
9                  input           :       in std_logic;
10
11                 count_out               :       out     std_logic_vector(7
                       downto 0)
12         );
13 end input_counter;
14
15 architecture behaviour of input_counter is
16
17         signal count, new_count :       unsigned(7 downto 0);
18         signal input_high                       : std_logic;
19
20 begin
21
22         process (clk)
23         begin
24                 if (rising_edge(clk)) then
25                         if (reset = '1') then
26                                 count <= (others => '0');
27                                 input_high <= '0';
28                         else
29                                 if (input = '1') then
30                                         if (input_high = '0') then
31                                                 count <= new_count;
32                                                 input_high <= '1';
33                                         else
34                                                 count <= count;
35                                                 input_high <= input_high;
36                                         end if;
37                                 else
38                                         count <= count;
39                                         input_high <= '0';
40                                 end if;
41                         end if;
42                 end if;
43         end process;
44
45         process (count)
46         begin
47                 new_count <= count + 1;
48         end process;
49
50         count_out <= std_logic_vector (count);
51
52 end architecture behaviour;
```

spi_master.vhd

```
1  ----------------------------------------------------------------------------------
2  -- PROJECT: SPI MASTER AND SLAVE FOR FPGA
3  ----------------------------------------------------------------------------------
4  -- AUTHORS: Jakub Cabal <>
5  -- LICENSE: The MIT License, please read LICENSE file
6  -- WEBSITE: https://github.com/jakubcabal/spi-fpga
7  ----------------------------------------------------------------------------------
8
9  library IEEE;
10 use IEEE.STD_LOGIC_1164.ALL;
11 use IEEE.NUMERIC_STD.ALL;
12 use IEEE.MATH_REAL.ALL;
13
14 -- THE SPI MASTER MODULE SUPPORT ONLY SPI MODE 0 (CPOL=0, CPHA=0)!!!
15
16 entity SPI_MASTER is
17     Generic (
18         CLK_FREQ    : natural := 50e6; -- set system clock frequency in Hz
19         SCLK_FREQ   : natural := 2e6;  -- set SPI clock frequency in Hz (
               condition: SCLK_FREQ <= CLK_FREQ/10)
20         WORD_SIZE   : natural := 8;    -- size of transfer word in bits,
               must be power of two
21         SLAVE_COUNT : natural := 1     -- count of SPI slaves
22     );
23     Port (
24         CLK      : in  std_logic; -- system clock
25         RST      : in  std_logic; -- high active synchronous reset
26         -- SPI MASTER INTERFACE
27         SCLK     : out std_logic; -- SPI clock
28         CS_N     : out std_logic_vector(SLAVE_COUNT-1 downto 0); -- SPI
               chip select, active in low
29         MOSI     : out std_logic; -- SPI serial data from master to slave
30         MISO     : in  std_logic; -- SPI serial data from slave to master
31         -- INPUT USER INTERFACE
32         DIN      : in  std_logic_vector(WORD_SIZE-1 downto 0); -- data for
               transmission to SPI slave
33         DIN_ADDR : in  std_logic_vector(natural(ceil(log2(real(SLAVE_COUNT
               ))))-1 downto 0); -- SPI slave address
34         DIN_LAST : in  std_logic; -- when DIN_LAST = 1, last data word,
               after transmit will be asserted CS_N
35         DIN_VLD  : in  std_logic; -- when DIN_VLD = 1, data for
               transmission are valid
36         DIN_RDY  : out std_logic; -- when DIN_RDY = 1, SPI master is ready
               to accept valid data for transmission
37         -- OUTPUT USER INTERFACE
38         DOUT     : out std_logic_vector(WORD_SIZE-1 downto 0); -- received
               data from SPI slave
39         DOUT_VLD : out std_logic  -- when DOUT_VLD = 1, received data are
               valid
40     );
41 end entity;
42
```

```
43  architecture RTL of SPI_MASTER is
44
45      constant DIVIDER_VALUE : natural := (CLK_FREQ/SCLK_FREQ)/2;
46      constant WIDTH_CLK_CNT : natural := natural(ceil(log2(real(
            DIVIDER_VALUE))));
47      constant WIDTH_ADDR    : natural := natural(ceil(log2(real(SLAVE_COUNT
            ))));
48      constant BIT_CNT_WIDTH : natural := natural(ceil(log2(real(WORD_SIZE))
            ));
49
50      type state_t is (idle, first_edge, second_edge, transmit_end,
            transmit_gap);
51
52      signal addr_reg        : unsigned(WIDTH_ADDR-1 downto 0);
53      signal sys_clk_cnt     : unsigned(WIDTH_CLK_CNT-1 downto 0);
54      signal sys_clk_cnt_max : std_logic;
55      signal spi_clk         : std_logic;
56      signal spi_clk_rst     : std_logic;
57      signal din_last_reg_n  : std_logic;
58      signal first_edge_en   : std_logic;
59      signal second_edge_en  : std_logic;
60      signal chip_select_n   : std_logic;
61      signal load_data       : std_logic;
62      signal miso_reg        : std_logic;
63      signal shreg           : std_logic_vector(WORD_SIZE-1 downto 0);
64      signal bit_cnt         : unsigned(BIT_CNT_WIDTH-1 downto 0);
65      signal bit_cnt_max     : std_logic;
66      signal rx_data_vld     : std_logic;
67      signal master_ready    : std_logic;
68      signal present_state   : state_t;
69      signal next_state      : state_t;
70
71  begin
72
73      ASSERT (DIVIDER_VALUE >= 5) REPORT "condition: SCLK_FREQ <= CLK_FREQ
            /10" SEVERITY ERROR;
74
75      load_data <= master_ready and DIN_VLD;
76      DIN_RDY   <= master_ready;
77
78      --
            ------------------------------------------------------------------------

79      --  SYSTEM CLOCK COUNTER
80      --
            ------------------------------------------------------------------------

81
82      sys_clk_cnt_max <= '1' when (to_integer(sys_clk_cnt) = DIVIDER_VALUE
            -1) else '0';
83
84      sys_clk_cnt_reg_p : process (CLK)
85      begin
86          if (rising_edge(CLK)) then
87              if (RST = '1' or sys_clk_cnt_max = '1') then
88                  sys_clk_cnt <= (others => '0');
```

```
89              else
90                  sys_clk_cnt <= sys_clk_cnt + 1;
91              end if;
92          end if;
93      end process;
94
95      --
           ------------------------------------------------------------------------
96      --  SPI CLOCK GENERATOR AND REGISTER
97      --
           ------------------------------------------------------------------------
98
99      spi_clk_gen_p : process (CLK)
100     begin
101         if (rising_edge(CLK)) then
102             if (RST = '1' or spi_clk_rst = '1') then
103                 spi_clk <= '0';
104             elsif (sys_clk_cnt_max = '1') then
105                 spi_clk <= not spi_clk;
106             end if;
107         end if;
108     end process;
109
110     SCLK <= spi_clk;
111
112     --
           ------------------------------------------------------------------------
113     --  BIT COUNTER
114     --
           ------------------------------------------------------------------------
115
116     bit_cnt_max <= '1' when (bit_cnt = WORD_SIZE-1) else '0';
117
118     bit_cnt_p : process (CLK)
119     begin
120         if (rising_edge(CLK)) then
121             if (RST = '1' or spi_clk_rst = '1') then
122                 bit_cnt <= (others => '0');
123             elsif (second_edge_en = '1') then
124                 bit_cnt <= bit_cnt + 1;
125             end if;
126         end if;
127     end process;
128
129     --
           ------------------------------------------------------------------------
130     --  SPI MASTER ADDRESSING
131     --
           ------------------------------------------------------------------------
132
```

```
133     addr_reg_p : process (CLK)
134     begin
135         if (rising_edge(CLK)) then
136             if (RST = '1') then
137                 addr_reg <= (others => '0');
138             elsif (load_data = '1') then
139                 addr_reg <= unsigned(DIN_ADDR);
140             end if;
141         end if;
142     end process;
143
144     one_slave_g: if (SLAVE_COUNT = 1) generate
145         CS_N(0) <= chip_select_n;
146     end generate;
147
148     more_slaves_g: if (SLAVE_COUNT > 1) generate
149         cs_n_g : for i in 0 to SLAVE_COUNT-1 generate
150             cs_n_p : process (addr_reg, chip_select_n)
151             begin
152                 if (addr_reg = i) then
153                     CS_N(i) <= chip_select_n;
154                 else
155                     CS_N(i) <= '1';
156                 end if;
157             end process;
158         end generate;
159     end generate;
160
161     --
        ------------------------------------------------------------------------

162     --  DIN LAST RESISTER
163     --
        ------------------------------------------------------------------------

164
165     din_last_reg_n_p : process (CLK)
166     begin
167         if (rising_edge(CLK)) then
168             if (RST = '1') then
169                 din_last_reg_n <= '0';
170             elsif (load_data = '1') then
171                 din_last_reg_n <= not DIN_LAST;
172             end if;
173         end if;
174     end process;
175
176     --
        ------------------------------------------------------------------------

177     --  MISO SAMPLE REGISTER
178     --
        ------------------------------------------------------------------------

179
180     miso_reg_p : process (CLK)
```

```
181     begin
182         if (rising_edge(CLK)) then
183             if (first_edge_en = '1') then
184                 miso_reg <= MISO;
185             end if;
186         end if;
187     end process;
188
189     --
            ------------------------------------------------------------------------

190     --  DATA SHIFT REGISTER
191     --
            ------------------------------------------------------------------------

192
193     shreg_p : process (CLK)
194     begin
195         if (rising_edge(CLK)) then
196             if (load_data = '1') then
197                 shreg <= DIN;
198             elsif (second_edge_en = '1') then
199                 shreg <= shreg(WORD_SIZE-2 downto 0) & miso_reg;
200             end if;
201         end if;
202     end process;
203
204     DOUT <= shreg;
205     MOSI <= shreg(WORD_SIZE-1);
206
207     --
            ------------------------------------------------------------------------

208     --  DATA OUT VALID RESISTER
209     --
            ------------------------------------------------------------------------

210
211     dout_vld_reg_p : process (CLK)
212     begin
213         if (rising_edge(CLK)) then
214             if (RST = '1') then
215                 DOUT_VLD <= '0';
216             else
217                 DOUT_VLD <= rx_data_vld;
218             end if;
219         end if;
220     end process;
221
222     --
            ------------------------------------------------------------------------

223     --  SPI MASTER FSM
224     --
            ------------------------------------------------------------------------
```

```
225
226     -- PRESENT STATE REGISTER
227     fsm_present_state_p : process (CLK)
228     begin
229         if (rising_edge(CLK)) then
230             if (RST = '1') then
231                 present_state <= idle;
232             else
233                 present_state <= next_state;
234             end if;
235         end if;
236     end process;
237
238     -- NEXT STATE LOGIC
239     fsm_next_state_p : process (present_state, DIN_VLD, sys_clk_cnt_max,
240                                 bit_cnt_max)
241     begin
242         case present_state is
243             when idle =>
244                 if (DIN_VLD = '1') then
245                     next_state <= first_edge;
246                 else
247                     next_state <= idle;
248                 end if;
249
250             when first_edge =>
251                 if (sys_clk_cnt_max = '1') then
252                     next_state <= second_edge;
253                 else
254                     next_state <= first_edge;
255                 end if;
256
257             when second_edge =>
258                 if (sys_clk_cnt_max = '1') then
259                     if (bit_cnt_max = '1') then
260                         next_state <= transmit_end;
261                     else
262                         next_state <= first_edge;
263                     end if;
264                 else
265                     next_state <= second_edge;
266                 end if;
267
268             when transmit_end =>
269                 if (sys_clk_cnt_max = '1') then
270                     next_state <= transmit_gap;
271                 else
272                     next_state <= transmit_end;
273                 end if;
274
275             when transmit_gap =>
276                 if (sys_clk_cnt_max = '1') then
277                     next_state <= idle;
278                 else
279                     next_state <= transmit_gap;
280                 end if;
```

```
281
282             when others =>
283                 next_state <= idle;
284         end case;
285     end process;
286
287     -- OUTPUTS LOGIC
288     fsm_outputs_p : process (present_state, din_last_reg_n,
            sys_clk_cnt_max)
289     begin
290         case present_state is
291             when idle =>
292                 master_ready   <= '1';
293                 chip_select_n  <= not din_last_reg_n;
294                 spi_clk_rst    <= '1';
295                 first_edge_en  <= '0';
296                 second_edge_en <= '0';
297                 rx_data_vld    <= '0';
298
299             when first_edge =>
300                 master_ready   <= '0';
301                 chip_select_n  <= '0';
302                 spi_clk_rst    <= '0';
303                 first_edge_en  <= sys_clk_cnt_max;
304                 second_edge_en <= '0';
305                 rx_data_vld    <= '0';
306
307             when second_edge =>
308                 master_ready   <= '0';
309                 chip_select_n  <= '0';
310                 spi_clk_rst    <= '0';
311                 first_edge_en  <= '0';
312                 second_edge_en <= sys_clk_cnt_max;
313                 rx_data_vld    <= '0';
314
315             when transmit_end =>
316                 master_ready   <= '0';
317                 chip_select_n  <= '0';
318                 spi_clk_rst    <= '1';
319                 first_edge_en  <= '0';
320                 second_edge_en <= '0';
321                 rx_data_vld    <= sys_clk_cnt_max;
322
323             when transmit_gap =>
324                 master_ready   <= '0';
325                 chip_select_n  <= not din_last_reg_n;
326                 spi_clk_rst    <= '1';
327                 first_edge_en  <= '0';
328                 second_edge_en <= '0';
329                 rx_data_vld    <= '0';
330
331             when others =>
332                 master_ready   <= '0';
333                 chip_select_n  <= not din_last_reg_n;
334                 spi_clk_rst    <= '1';
335                 first_edge_en  <= '0';
```

```
336                 second_edge_en <= '0';
337                 rx_data_vld    <= '0';
338             end case;
339         end process;
340
341 end architecture;
```

input_generator.vhd

```
1  library IEEE;
2  use IEEE.std_logic_1164.ALL;
3
4  entity input_generator is
5          port(   clk             :       in      std_logic;
6                  reset           :       in      std_logic;
7
8                  output          :       buffer  std_logic
9          );
10 end input_generator;
11
12 architecture behaviour of input_generator is
13
14 constant freq           : integer       := 50; -- frequency in MHz
15 constant s              : integer       := 1000000 * freq; --clock cycles
       in one second
16 constant ms             : integer       := 1000 * freq; -- clock cycles in
       one milli second
17 constant us             : integer       := 1 * freq;-- clock cycles in one
       micro second
18
19
20 begin
21
22         process (clk)
23
24         variable clk_count : integer := 0;
25
26         begin
27                 if (rising_edge(clk)) then
28                         if (reset = '1') then
29                                 output <= '0';
30                         else
31                                 if (clk_count < (100 * us)) then
32                                         output <= '0';
33                                         clk_count := clk_count + 1;
34                                 elsif (clk_count < (200 * us - 1)) then
35                                         output <= '1';
36                                         clk_count := clk_count + 1;
37                                 else
38                                         output <= '0';
39                                         clk_count := 0;
40                                 end if;
41                         end if;
42                 end if;
43         end process;
44
45 end architecture behaviour;
```

readout_SPI.ino

```
1  #include<SPI.h>
2
3  volatile boolean received;
4  volatile byte Slavereceived;
5
6  void setup()
7  {
8    Serial.begin(115200);
9
10   pinMode(MISO,OUTPUT);                    //Sets MISO as OUTPUT
11
12   SPCR |= _BV(SPE);                        //Turn on SPI in Slave Mode
13
14   received = false;
15
16   SPI.attachInterrupt();                   //Interuupt ON is set for SPI
         commnucation
17
18   Serial.println("Setup complete");
19 }
20
21
22 ISR (SPI_STC_vect)                         //Inerrrput routine function
23 {
24   Slavereceived = SPDR;          // Value received from master if store in
         variable slavereceived
25
26   received = true;                         //Sets received as True
27 }
28
29
30 void loop(){
31   if(received){
32       Serial.println(Slavereceived);
33       received = false;
34     }
35 }
```

data_processing.py

```
import csv
import os
import numpy as np
import matplotlib.pyplot as plt

prefix_conversions = {"u": 10**-6, "m": 10**-3, "k": 10**3, "M": 10**6}


filename = "8bit_2ms_119kcps.csv"

mean_and_std = True
barplot = False


def main():
    counts = read_csv_file(filename)

    n_bits, measure_period, measure_period_multiplier, tested_cps =
        read_filename(filename)

    cps = convert_counts_to_cps(counts, measure_period,
        measure_period_multiplier)

    mean, std = get_mean_and_std(cps)

    print(f"Mean: {mean} +/- {std}")

    if barplot:
        make_barplot(cps)


def read_csv_file(filename):
    with open(filename) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        counts = []
        first = True
        for count in csv_reader:
            if first:
                first = False
                continue

            counts.append(int(count[0]))
    return counts

def read_filename(filename: str):
    filename = filename.split(".csv")[0]
    infos = filename.split("_")

    n_bits = int(infos[0].strip("bit"))
    measure_period = int(infos[1].strip("ums"))
    measure_period_multiplier = prefix_conversions[infos[1][-2]]
    tested_cps = float(infos[2].strip("Mkcps")) * prefix_conversions[infos
        [2][-4]]

    return (n_bits, measure_period, measure_period_multiplier, tested_cps)
```

```
53
54 def convert_counts_to_cps(counts, measure_period,
       measure_period_multiplier):
55     cps = []
56     for count in counts:
57         # print(f"{count}, {measure_period}, {measure_period_multiplier}")
58         cps.append(round(count / measure_period /
               measure_period_multiplier))
59     return cps
60
61 def get_mean_and_std(vals):
62     mean = np.mean(vals)
63     std = np.std(vals)
64
65     return (mean, std)
66
67 def make_barplot(vals):
68     unique_vals = np.unique(vals)
69     occurences = []
70
71     for unique_val in unique_vals:
72         occurences.append(np.count_nonzero(vals == unique_val))
73
74     print(unique_vals)
75     print(occurences)
76
77     plt.bar(unique_vals, occurences)
78     plt.title("Barplot of the received values")
79     plt.xlabel('Value')
80     plt.ylabel("Occurences")
81     plt.show()
82
83
84 if __name__ == "__main__":
85     main()
```

Bibliography

[1] He Qiuyang et al. “An accurate simulation model for single-photon avalanche diodes including important statistical effects”. In: *Journal of Semiconductors* 34.10 (Oct. 2013), p. 104007. DOI: 10.1088/1674-4926/34/10/104007. URL: https://dx.doi.org/10.1088/1674-4926/34/10/104007.

[2] T. Chaves de Albuquerque et al. “Integration of SPAD in 28nm FDSOI CMOS technology”. In: *2018 48th European Solid-State Device Research Conference (ESSDERC)*. 2018, pp. 82–85. DOI: 10.1109/ESSDERC.2018.8486852.

[3] David D. Awschalom et al. “Quantum technologies with optically interfaced solid-state spins”. In: *Nature Photonics* 12 (2018), pp. 516–527. DOI: 10.1038/s41566-018-0232-2.

[4] J.F. Barry et al. “Optical magnetic detection of single-neuron action potentials using quantum defects in diamond”. In: (Oct. 2016). DOI: https://doi.org/10.1073/pnas.1601513113. URL: https://www.pnas.org/doi/full/10.1073/pnas.1601513113.

[5] John F. Barry et al. “Sensitivity optimization for NV-diamond magnetometry”. In: *Rev. Mod. Phys.* 92 (1 Mar. 2020), p. 015004. DOI: 10.1103/RevModPhys.92.015004. URL: https://link.aps.org/doi/10.1103/RevModPhys.92.015004.

[6] Kai Bongs, Simon Bennett, and Anke Lohmann. “Quantum sensors will start a revolution — if we deploy them right”. In: *Nature* (May 2023). URL: https://www.nature.com/articles/d41586-023-01663-0.

[7] Emilie Bourgeois, Michal Gulka, and Milos Nesladek. “Photoelectric Detection and Quantum Readout of Nitrogen-Vacancy Center Spin States in Diamond”. In: *Advanced Optical Materials* 8.12 (2020), p. 1902132. DOI: https://doi.org/10.1002/adom.201902132. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/adom.201902132. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/adom.201902132.

[8] Danilo Bronzi et al. “Large-area CMOS SPADs with very low dark counting rate”. In: *Proceedings of SPIE - The International Society for Optical Engineering* 8631 (Feb. 2013). DOI: 10.1117/12.2004209.

[9] Jakub Cabal. *spi-fpga*. Version 1.1. Apr. 2021. URL: https://github.com/jakubcabal/spi-fpga.

[10] S Cova et al. “Avalanche photodiodes and quenching circuits for single-photon detection”. In: *Applied optics* 35 (Apr. 1996), pp. 1956–76. DOI: 10.1364/AO.35.001956.

[11] Marcus W. Doherty et al. “The nitrogen-vacancy colour centre in diamond”. In: *Physics Reports* 528 (1 Feb. 2013), pp. 1–45. URL: https://doi.org/10.1016/j.physrep.2013.02.001.

[12] Peter Fischer, Robert K. Zimmermann, and Benedict Maisano. “CMOS SPAD sensor chip for the readout of scintillating fibers”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1040 (2022), p. 167033. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2022.167033. URL: https://www.sciencedirect.com/science/article/pii/S0168900222004582.

[13] “Fundamentals of magnetic field measurement with NV centers in diamond”. In: *Qnami* (Dec. 2020). URL: https://qnami.ch/wp-content/uploads/2020/12/2020-12-07-Qnami-TN1-The-NV-center-1.pdf.

[14] G. Giustolisi, R. Mita, and G. Palumbo. “Behavioral modeling of statistical phenomena of single-photon avalanche diodes”. In: *International Journal of Circuit Theory and Applications* 40.7 (2012), pp. 661–679. DOI: https://doi.org/10.1002/cta.748. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cta.748. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cta.748.

[15] Francesco Gramuglia et al. “A Low-Noise CMOS SPAD Pixel With 12.1 Ps SPTR and 3 Ns Dead Time”. In: *IEEE Journal of Selected Topics in Quantum Electronics* 28.2: Optical Detectors (2022), pp. 1–9. DOI: 10.1109/JSTQE.2021.3088216.

[16] NIH News in Health. *Technologies Enhance Tumor Surgery. Helping Surgeons Spot and Remove Cancer*. Accessed: 15-06-2023. URL: https://newsinhealth.nih.gov/2016/02/technologies-enhance-tumor-surgery.

[17] David A. Hopper, Henry J. Shulevitz, and Lee C. Bassett. “Spin Readout Techniques of the Nitrogen-Vacancy Center in Diamond”. In: *Micromachines* 9.9 (2018). ISSN: 2072-666X. DOI: 10.3390/mi9090437. URL: https://www.mdpi.com/2072-666X/9/9/437.

[18] Shenjie Huang et al. “Optimal Photon Counting Receiver for Sub-Dead-Time Signal Transmission”. In: *Journal of Lightwave Technology* 38.18 (2020), pp. 5225–5235. DOI: 10.1109/JLT.2020.3000723.

[19] National Cancer Institute. *Surgery to Treat Cancer*. Accessed: 15-06-2023. URL: https://www.cancer.gov/about-cancer/treatment/types/surgery#:~:text=Once%5C%20you%5C%20are%5C%20under%5C%20anesthesia,other%5C%20tissues%5C%20near%5C%20the%5C%20tumor.

[20] Khalil Jradi, Denis Pellion, and Dominique Ginhac. “Design, Characterization and Analysis of a 0.35 µm CMOS SPAD”. In: *Sensors* 14.12 (2014), pp. 22773–22784. ISSN: 1424-8220. DOI: 10.3390/s141222773. URL: https://www.mdpi.com/1424-8220/14/12/22773.

[21] Juha Kostamovaara et al. “Solid-State Pulsed Time-of-Flight 3-D Range Imaging Using CMOS SPAD Focal Plane Array Receiver and Block-Based Illumination Techniques”. In: *IEEE Photonics Journal* 14.2 (2022), pp. 1–11. DOI: 10.1109/JPHOT.2022.3153487.

[22] M.J. Lee et al. “High-Performance Back-Illuminated Three-Dimensional Stacked Single-Photon Avalanche Diode Implemented in 45-nm CMOS Technology”. In: (2018). DOI: 10.1109/JSTQE.2018.2827669.

[23] R. Mita, G. Palumbo, and P.G. Fallica. “Accurate model for single-photon avalanche diodes”. In: *IET Circuits, Devices Systems* 2.2 (2008), pp. 207–212. DOI: 10.1049/iet-cds:20070180.

[24] K. Morimoto. “Megapixel SPAD cameras for time-resolved applications”. In: (2021). DOI: https://doi.org/10.5075/epfl-thesis-8773. URL: https://infoscience.epfl.ch/record/283481.

[25] S. Pellegrini et al. “Industrialised SPAD in 40 nm technology”. In: *2017 IEEE International Electron Devices Meeting (IEDM)*. 2017, pp. 16.5.1–16.5.4. DOI: 10.1109/IEDM.2017.8268404.

[26] Maxim Integrated Products. *High-Speed, Low-Power Voltage Comparators, MAX903 Datasheet*. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX900-MAX903.pdf.

[27] Justin A. Richardson et al. “Scaleable Single-Photon Avalanche Diode Structures in Nanometer CMOS Technology”. In: *IEEE Transactions on Electron Devices* 58.7 (2011), pp. 2028–2035. DOI: 10.1109/TED.2011.2141138.

[28] Elham Sarbazi, Majid Safari, and Harald Haas. “The Impact of Long Dead Time on the Photocount Distribution of SPAD Receivers”. In: *2018 IEEE Global Communications Conference (GLOBECOM)*. 2018, pp. 1–6. DOI: 10.1109/GLOCOM.2018.8647814.

[29] M. Stipčević, H. Skenderović, and D. Gracin. “Characterization of a novel avalanche photodiode for single photon detection in VIS-NIR range”. In: *Opt. Express* 18.16 (Aug. 2010), pp. 17448–17459. DOI: 10.1364/OE.18.017448. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-18-16-17448.

[30] Mario Stipčević. “Active quenching circuit for single-photon detection with Geiger mode avalanche photodiodes”. In: *Appl. Opt.* 48.9 (Mar. 2009), pp. 1705–1714. DOI: 10.1364/AO.48.001705. URL: https://opg.optica.org/ao/abstract.cfm?URI=ao-48-9-1705.

[31] P. Sun. “Flexible CMOS Single-Photon Avalanche Diode Image Sensor Technology”. In: (2016). DOI: https://doi.org/10.4233/uuid:11717f7d-51c9-471b-8f9e-ee1a70e7f032. URL: https://repository.tudelft.nl/islandora/object/uuid:11717f7d-51c9-471b-8f9e-ee1a70e7f032?collection=research.

[32] Tuomo Talala et al. “Time-Resolved Raman Spectrometer With High Fluorescence Rejection Based on a CMOS SPAD Line Sensor and a 573-nm Pulsed Laser”. In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–10. DOI: 10.1109/TIM.2021.3054679.

[33] G. Torilla et al. “DCR and crosstalk characterization of a bi-layered 24 × 72 CMOS SPAD array for charged particle detection”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1046 (2023), p. 167693. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2022.167693. URL: https://www.sciencedirect.com/science/article/pii/S0168900222009858.

[34] TSMC. *TSMC 45/40nm CMOS Design rule*. 2015.

[35] P. Vines et al. “High performance planar germanium-on-silicon single-photon avalanche diode detectors”. In: *Nature Communications* (2019). DOI: 10.1038/s41467-019-08830-w.

[36] I. Vornicu et al. “Low-Noise and High-Efficiency Near-IR SPADs in 110nm CIS Technology”. In: *ESSDERC 2019 - 49th European Solid-State Device Research Conference (ESSDERC)*. 2019, pp. 250–253. DOI: 10.1109/ESSDERC.2019.8901757.

[37] Ion Vornicu et al. “Compact CMOS active quenching/recharge circuit for SPAD arrays”. In: *International Journal of Circuit Theory and Applications* 44 (July 2015). DOI: 10.1002/cta.2113.

[38] Ion Vornicu et al. “Design of High-Efficiency SPADs for LiDAR Applications in 110nm CIS Technology”. In: *IEEE Sensors Journal* 21.4 (2021), pp. 4776–4785. DOI: 10.1109/JSEN.2020.3032106.

[39] F. Zappa et al. “Monolithic active-quenching and active-reset circuit for single-photon avalanche detectors”. In: *IEEE Journal of Solid-State Circuits* 38.7 (2003), pp. 1298–1301. DOI: 10.1109/JSSC.2003.813291.

[40] F. Zappa et al. “Principles and features of single-photon avalanche diode arrays”. In: *Sensors and Actuators A: Physical* 140.1 (2007), pp. 103–112. ISSN: 0924-4247. DOI: https://doi.org/10.1016/j.sna.2007.06.021. URL: https://www.sciencedirect.com/science/article/pii/S0924424707004967.

[41] F. Zappa et al. “SPICE modeling of single photon avalanche diodes”. In: *Sensors and Actuators A: Physical* 153.2 (2009), pp. 197–204. ISSN: 0924-4247. DOI: https://doi.org/10.1016/j.sna.2009.05.007. URL: https://www.sciencedirect.com/science/article/pii/S0924424709002581.

[42] Jixing Zhang et al. *Diamond Nitrogen-Vacancy Center Magnetometry: Advances and Challenges*. 2020. DOI: 10.48550/arXiv.2010.10231. arXiv: 2010.10231.