Development of a Multichannel TCSPC System in a Spartan 6 FPGA

LinoSPAD - Fluorescence Lifetime Imaging for Fluorescence Guided Surgery

Harald Homulle 4032578

TU Delft
Faculty of EEMCS
Master in Electrical Engineering
Micro Electronics

EPFL
Quantum Architecture Group

June 2014
Development of a Multichannel TCSPC System in a Spartan 6 FPGA (25 June 2014)

This thesis is made using \LaTeX{} in Nimbus Roman 11 pt.
Preface

The finalization of the master is done with a master project of one year. This master project was part of a multidisciplinary project for cancer research named “Advanced sensing system with single-photon sensitivity and picosecond time resolution in a compact probe for fluorescence guided surgery” and financed by the Swiss National Foundation (SNF). Part of it was done at EPFL, Lausanne, Switzerland. Work on the development of a medical probe for Fluorescence Lifetime Imaging was carried out.

The clinical application would be in image guided surgical oncology, using Near InfraRed (NIR) fluorescent markers to help the surgeon to identify tumoural tissue and metastatic lymph nodes from normal tissues with a much higher level of certainty thanks to the differences in fluorescence lifetime. The lifetime between different cell types can be large, i.e. different tissues can be visually separated.

The camera system to be developed comprises four parts: the hardware, the firmware/software, optimized optics and an illumination system (currently a pulsed laser). The design of the prototype hardware: the PCBs and the camera chip was done by Samuel Burri. Francois Powolny worked on the last two parts, building of the system and selecting the right optics.

The work I did through this master thesis was dealing with the systems firmware / software implementation. I developed the firmware for the FPGA and software for the display and analysis of the fluorescence signals, furthermore I took part in a preliminary study of ICG in vivo. Finally complete tests of both FPGA firmware and software were done.

Harald Homulle
Delft & Lausanne, June 2014

Acknowledgements

I would like to thank the following people for making this project possible and for the support during my time in Switzerland and Delft.

Francois Powolny, my daily supervisor, who helped me a lot in understanding the projects purpose, the measurements and the different systems. Besides this thanks for all the time in meetings, discussions and the feedback on my work, which were very useful.

Edoardo Charbon, the groups professor, for making the project possible and giving me the opportunity for an intern ship in Switzerland.

Samuel Burri, for the help on the firmware and of course the design of the boards and chip without all would not have been possible.

Claudio Bruschini, the project manager.

Lucio Carrara, Fastree 3D, for the discussions on the firmware and the implementation of different pieces of VHDL.

Shingo Mandai, for borrowing his SPAD chip so I have at least some SPAD results to show.
Summary

For the master project work was carried out for the development of a fluorescence lifetime imaging probe for fluorescence guided surgery. For this project a prototype was designed. The work on the prototype was divided into three main parts, hardware, firmware / software, and system / optics. In this thesis the firmware / software of the system are described. An overview of the system is given and the performance is evaluated.

The systems hardware consists of a mainboard with a Spartan 6 FPGA and a Cypress FX3 USB3 controller. Secondly a daughterboard was designed which houses the LinoSPAD chip, a line of 256 SPADs. The outputs of the SPADs are directly connected to the FPGA.

The FPGAs firmware was developed. The system had to incorporate a Time Correlated Single Photon Counting structure for the 256 channels. TCSPC calculates the timing of individual photons and stores this information in a histogram. The Time to Digital Converters were implemented in the carrychain structure on the FPGA. A clock of 100 MHz is used, therefore 512 stages (128 Carry4 blocks) form the delayline of the TDCs. The TDC range is > 10 ns to fill the complete clock period.

The histograms were stored in the FPGA RAM memory blocks. The final firmware has 8 TDCs, each having 32 channels. Those channels are connected to one TDC. Each channel has its own RAM memory in which the histogram is stored. The memories can be read-out in less than 2 ms over USB3.

The performance of the TDCs was assessed. The resolution is 20-21 ps and the non linearities are DNL 4 LSB and INL 7.5 LSB. The results were confirmed both in simulation and in the physical design. Crosstalk from SPADs was removed by not storing timing information when multiple SPADs fire in one clock period.

Secondly the systems software was developed. The software has a user-friendly GUI in which the main aspects of the fluorescence signals can be plotted. Both the raw TCSPC histograms and the extracted information as intensity and lifetime can be shown.

In order to extract the fluorescence lifetime, two algorithms were studied and compared: Wiener filter and Centre of Mass method (CoM).

The Wiener filter is based on deconvolution of the measured signal with the Instrument Response Function of the system (IRF). On the deconvoluted exponential, a fit is made to extract the lifetime. The CoM uses the centre of mass of the fluorescence histogram and the centre of mass of the IRF to make an estimation of the lifetime.

Both algorithms show similar results in simulation with errors up to 30 ps over a lifetime range from 100 ps up to 1 ns. These errors are obtained for a histogram with 2500 photon counts, noise and systems non linearities being present. However the Centre of Mass method is up to 100× faster compared to the Wiener filter due to its lower complexity.

The FluoCAM camera system, already developed in a previous project, was calibrated. It was also used to perform biological experiments in-vitro and in-vivo. Findings of fluorescence lifetimes are comparable to literature for well-known fluorescence compounds. Furthermore first tests in vivo (mice) reveal similar lifetimes over several days.

The LinoSPAD system could not be tested for fluorescence as the LinoSPAD chip is not working. Instead a fluorescence emulator was designed in a second FPGA to test the LinoSPAD FPGA system.
The emulator mimics fluorescence exponential signals, however due to FPGA limitations the derived exponentials are not analysable to extract lifetimes.

Besides an emulator, some experiments were done with a single SPAD. The systems resolution and non-linearities were confirmed using SPAD noise density tests. The noise rate of the SPAD was found in the same order as expected (100 Hz).

Finally with the data collected from FluoCAM, the performance achievable with the LinoSPAD camera system was estimated.

Main system requirements have been fulfilled during the project. However the system has to be tested with the working LinoSPAD chip to confirm the usability of the complete system. The first step towards a fluorescence probe has been set.
# Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>i</td>
</tr>
<tr>
<td>Summary</td>
<td>iii</td>
</tr>
<tr>
<td>Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Abbreviations</td>
<td>viii</td>
</tr>
</tbody>
</table>

## 1 Introduction

1.1 Photoluminescence

1.1.1 Fluorescence

1.1.2 Phosphorescence

1.1.3 Fluorophores

1.2 IndoCyanine Green

1.3 Fluorescence lifetime measurement techniques

1.3.1 Frequency domain

1.3.2 Time domain

1.4 Single Photon Avalanche Diodes

1.4.1 FluoCAM

1.4.2 LinoSPAD

1.5 Camera systems

1.5.1 Microscopy: Lifa / Microtime 200

1.5.2 Spectroscopy: Picosecond / PicoMaster / Easylife TCSPC

1.5.3 (Pre-)Clinical: IVIS / FLARE / SPY / PDE / Fluostick / LinoSPAD

## I Firmware development

2 Time to Digital Converters

2.1 TDCs overview

2.1.1 Time Counter

2.1.2 Delayline

2.1.3 Oscillator / Pulse Shrinking

2.1.4 Comparison

2.2 TDCs for the LinoSPAD system

3 Time Correlated Single Photon Counting - Spartan 6 implementation

3.1 Components

3.1.1 Input filter

3.1.2 TDC

## II List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDC</td>
<td>Time to Digital Converter</td>
</tr>
<tr>
<td>LinoSPAD</td>
<td>Linear Image Sensor Single Photon Avalanche Diode</td>
</tr>
<tr>
<td>IVIS</td>
<td>In Vivo Imaging System</td>
</tr>
<tr>
<td>FLARE</td>
<td>Fluorescence Lifetime Analysis and Real-time Evaluation System</td>
</tr>
<tr>
<td>SPY</td>
<td>Single Photon Yields</td>
</tr>
<tr>
<td>PDE</td>
<td>Photostability Evaluation System</td>
</tr>
<tr>
<td>Fluostick</td>
<td>Fluorescence Lifetime Imaging System</td>
</tr>
<tr>
<td>Lifa</td>
<td>Linear Image Sensor</td>
</tr>
<tr>
<td>Microtime 200</td>
<td>Microtime Imaging System 200</td>
</tr>
<tr>
<td>PicoMaster</td>
<td>Picoscope Master Bridge</td>
</tr>
<tr>
<td>Easylife TCSPC</td>
<td>Easylife Time-Correlated Single Photon Counting System</td>
</tr>
<tr>
<td>Spartan 6</td>
<td>FPGA Technology</td>
</tr>
</tbody>
</table>
7.2.1 Point setup ........................................... 67
7.2.2 Wide field setup .................................... 68
7.3 LinoSPAD optics ....................................... 68
  7.3.1 Line optics ......................................... 69
  7.3.2 Scanning ........................................... 69

8 Measurements FluoCAM .................................. 71
  8.1 Camera performance .................................. 71
    8.1.1 Noise (DCR) ........................................ 71
    8.1.2 Hot & Dead ......................................... 71
    8.1.3 Resolution ......................................... 72
    8.1.4 Non linearities .................................... 73
    8.1.5 Efficiency ......................................... 73
  8.2 Lifetime calibration & Comparison with literature ...... 74
    8.2.1 Water ............................................... 74
    8.2.2 Milk ................................................ 75
    8.2.3 Blood ............................................... 77
  8.3 In vivo .................................................. 78
    8.3.1 ICG-RGD ............................................ 79
    8.3.2 IVIS & FluoCAM .................................... 79

9 Measurements LinoSPAD .................................. 83
  9.1 Emulator ................................................ 83
    9.1.1 Principle ........................................... 83
    9.1.2 Implementation ..................................... 84
    9.1.3 Setup ............................................... 86
    9.1.4 Results ............................................ 87
  9.2 OneSPAD ................................................ 88
    9.2.1 Setup ............................................... 88
    9.2.2 SPAD pulse verification ............................. 88
    9.2.3 LinoSPAD system with OneSPAD ..................... 89
  9.3 LinoSPAD expectations .................................. 91

10 Conclusion .............................................. 93
  10.1 Evaluation of the LinoSPAD prototype ................... 93
  10.2 Further Work ......................................... 96

Bibliography ................................................. 97

List of Figures ............................................. 101

List of Tables .............................................. 104

A Design Brief ........................................... 105

B Schematic & Layout ....................................... 107

C Detailed lifetime analysis study .......................... 109
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCD</td>
<td>Charge Coupled Device</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal Oxide Semiconductor</td>
</tr>
<tr>
<td>CT</td>
<td>Computed Tomography</td>
</tr>
<tr>
<td>DCR</td>
<td>Dark Count Rate</td>
</tr>
<tr>
<td>DNL</td>
<td>Differential Non Linearity</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>FWHM</td>
<td>Full Width at Half Maximum</td>
</tr>
<tr>
<td>FX3</td>
<td>Cypress EZ-USB FX3</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>ICG</td>
<td>IndoCyanine Green</td>
</tr>
<tr>
<td>IFFT</td>
<td>Inverse Fast Fourier Transform</td>
</tr>
<tr>
<td>INL</td>
<td>Integral Non Linearity</td>
</tr>
<tr>
<td>IRF</td>
<td>Instruments Response Function</td>
</tr>
<tr>
<td>ISE</td>
<td>Integrated Software Environment</td>
</tr>
<tr>
<td>LED</td>
<td>Laser Emitting Diode</td>
</tr>
<tr>
<td>LFSR</td>
<td>Linear Feedback Shift Register</td>
</tr>
<tr>
<td>LSB</td>
<td>Least Significant Bit</td>
</tr>
<tr>
<td>LUT</td>
<td>Lookup Table</td>
</tr>
<tr>
<td>MRI</td>
<td>Magnetic Resonance Imaging</td>
</tr>
<tr>
<td>MSB</td>
<td>Most Significant Bit</td>
</tr>
<tr>
<td>NA</td>
<td>Not Available</td>
</tr>
<tr>
<td>PET</td>
<td>Positron Emission Tomography</td>
</tr>
<tr>
<td>PMT</td>
<td>Photomultiplier Tube</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>ROI</td>
<td>Region of Interest</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>SPAD</td>
<td>Single Photon Avalanche Diode</td>
</tr>
<tr>
<td>TCSPC</td>
<td>Time Correlated Single Photon Counting</td>
</tr>
<tr>
<td>TDC</td>
<td>Time to Digital Converter</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>VHSIC</td>
<td>Very High Speed Integrated Circuit</td>
</tr>
<tr>
<td>Xilinx Spartan 6</td>
<td>xc6slx100-3fgg676</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

According to the Dutch Center of Statistics (CBS) the main cause of death in the Netherlands is cancer (as of 2011) [1]. Therefore a lot of research is done to improve the life expectancy and quality of life of cancer patients, in particular in the domains of anti-cancer drugs, treatment regimens, medical imaging technology and operation techniques.

Techniques used to determine the cancer type and location are based on either chemical research on the human fluids or by imaging the patient [2]. Chemical analysis can normally determine the type of cancer, however exact locations have to be found by imaging afterwards; before surgery [3].

There are many ways of imaging (non-invasive). CT and MRI are commonly used, both produce per slice information of the body. Whereas CT provides more detailed images of the hard tissue, such as bones and muscles; MRI provides images of soft tissues, as heart and brain etc. However with these images it is not always possible to differentiate cancer tissue from healthy tissues. Therefore to detect the cancer, a (radioactive) contrast marker has to be applied.

Techniques as PET rely on chemical changes inside the patients body. Fluorodeoxyglucose (FDG), a sort of radioactive sugar, is used as marker, as the cancer has a high sugar uptake. The radioactivity in the body, where the tumour is located, is therefore much higher then in the surrounding tissue. This contrast is measured with PET.

Following a PET, CT or MRI scan, tumour can be removed using minimal-invasive techniques like endoscopy or laparoscopy, or by open surgery. The objective is to remove it completely (radically) with sufficient tumour-free margin.

However the clinical discrimination between tumour and normal tissue, which is still mostly based on visual inspection and palpation, is difficult during the operation.

Therefore the objective of this project is to develop a hand-held probe that highly improves the identification of tumour-free margins and local (lymph node) metastases during surgery using Near InfraRed (NIR) fluorescence imaging techniques.

The principle of fluorescence imaging is shown in Figure 1.1(a). The patient is imaged using a NIR (laser) light source, while he is injected with a fluorescence contrast agent (ICG). The fluorescence response is captured on the camera. Such a NIR fluorescence intensity image is shown in (b). For better visibility it can be merged with a colour image of the same spot as in (c).

Working in the NIR, i.e. in a 700-900 nm wavelength window has the substantial advantage of enabling deeper penetration in tissue (up to about 8 mm) as well as reducing the background due to the autofluorescence of (naturally occurring) endogenous fluorophores.

The additional feature added to the probe is the Fluorescence Lifetime Imaging (FLIM) capability. This approach is mainly motivated by the need to overcome several limitations of ‘simple’ intensity imaging. In particular, it shows the following potential advantages:
1.1 Photoluminescence

Photoluminescence is the process of absorbing light and re-emitting light with a usually lower energy. It can be divided into two types: fluorescence and phosphorescence. The lifetime of luminescent compounds is defined as the time for the compound to emit light after the absorption of photons.

The energy diagram for the absorption and emission from luminescence molecules is given in Figure 1.2. Both fluorescence and phosphorescence emission energy paths are drawn.

1.1.1 Fluorescence

The fluorescence process takes place in a fluorophore, a fluorescent chemical compound, some tissues show natural autofluorescence. After the absorption of a photon from a high energetic light source, the atom is excited. An electron is elevated into a higher energy state, forming a singlet state with a ground state electron.

The fluorescence mechanism relies on the fact that the excited electron falls back to its energetic ground state by emitting a fluorescence photon through a radiative transition. While the excited and ground state electron have opposite spin, the excited electron is allowed to fall back rapidly, as shown in Figure 1.2. This causes the lifetime to be short, i.e. in the order of nanoseconds and below. The lifetimes can be influenced by local parameters such as the pH value, or in particular the fact of being bound or unbound (with other molecules). [5–7]
1.1.2 Phosphorescence

Phosphorescence on the contrary is formed by a triplet electron state, also shown in Figure 1.2. The excited electron is first going through the singlet state after which it comes in the triplet state. In this state both electrons have the same spin, a rapid transition to the ground state is therefore not possible. This causes the lifetimes of phosphorescence to be large, i.e. milliseconds, seconds and larger. [5]

As the system is based on FLIM, the focus is on fluorescence and fluorophores.

1.1.3 Fluorophores

There exist many molecules that exhibit fluorescence excitation, commonly known as fluorophores. Fluorophores are excited at a specific wavelength $\lambda_{\text{abs}}$ and also exhibit photons at a typical wavelength $\lambda_{\text{emit}}$. [7]

The emission wavelength, i.e. the wavelength at which the fluorescence sample re-emits light, is commonly higher, while energy is lost in non-radiative transitions [7]. The energy $E$ is inversely proportional to the wavelength $\lambda$ with:

$$E = \frac{h \cdot c}{\lambda} \tag{1.1}$$

in which $h$ and $c$ are respectively Planck’s constant and the speed of light.

An example of absorption and emission spectra is shown in Figure 1.3. The shift in wavelength between the emission and absorption spectra's maxima is known as Stokes shift [8].

1.2 IndoCyanine Green

IndoCyanine Green (ICG) is currently the only NIR fluorophore FDA approved for clinical trials [9].

ICG fluorescence is currently used to determine liver failure. As ICG is only metabolised and broken down in the liver, a good indication of liver health can be given [10].

ICG is also used in ophthalmology. To diagnose intra ocular melanoma, i.e. leakage or blockage in the retina of the eye. This can be done by injecting ICG in the blood. While the ICG spreads through the eye, the retina is pictured and the effects can be observed whether a large amount of ICG is trapped in...
1.2. INDOCYANINE GREEN

the eye veins or not [11].

Finally ICG is useful in oncology. As tumour cells react differently to the fluorophore injected in the body than the surrounding tissue, they can be discriminated on basis of the fluorescence intensity and also the lifetime. This can help the surgeon to precisely find the cancer tissue and remove all tissue affected [12]. To make the ICG more specific to cancer, it can be chemically conjugated with an Arg-Gly-Asp (RGD) peptide sequence so as to target cancer cells with much higher efficiency.

Photodynamic therapy (PDT) with ICG does not only allow to find the cancer cells, but also to eliminate them. According to [12] ICG has successfully been used in combination with infrared laser excitation to kill colonic cancer cells.

[13] has shown that ICG and infrared LED activation kill 84% of oral cancer cells after six hours using 20 µM ICG concentration. ICG melanoma pre-treatment is up to 10 times more effective than conventional laser pre-treatment [14].

As shown in Figure 1.3, ICG’s peak in spectral absorption is around 790 nm [9, 15, 16]. Following Stokes shift, it emits fluorescence photons at 820 nm with a decreasing exponential, having a subnanosecond lifetime.

Note that these excitation and emission wavelengths can vary with the ICG concentration and solvent, as shown in Figure 8.5. Therefore many different excitation wavelengths are used.

Spectral stability is a major concern while using ICG. [17] notes that using ICG in biological environments leads to unexpected lifetime results. ICG in dilutions may lead to unexpected results if not taken care of the temperature, the solvent and the time [16, 18]. Aggregation of the dyes is dependent on time and concentration [16].

E.g. solutions in water are stable for around 4 hours [18]. Also storing the solution in the dark aggregates the ICG and after 7 days the absorption maximum is shifted up to 900 nm [18].

Various attempts to stabilize the ICG have been undertaken, using protein binding, membranes, charged polymer binding and more recently using silver particles [19]. A higher intensity, decreased lifetimes and better photostability have been observed.

1690 nm [17] (Concentration > 50 µM), 780 nm [16, 17], 785 nm [13], 805 nm [12], 810 nm [10, 16], 830 nm [14]
A second concern is photobleaching the fluorophore. A high energetic light source may destroy the fluorophores and thus reduce the brightness of the image. Photobleaching is more problematic for long exposure times and low intensity images.

The ICG dyes can only produce a limited number of photons before they are destroyed [7].

1.3 Fluorescence lifetime measurement techniques

The fluorophore is excited, with a light source, and the fluorescence is captured on a camera. The excitation is normally done with a laser with a specific wavelength. The fluorophores have an exponential fluorescence decay described by

$$I(t) = I_0 \cdot e^{-\frac{t}{\tau}}$$

from which the lifetime $\tau$ is the parameter to be extracted. This can be done both in time and frequency domain [20].

1.3.1 Frequency domain

In frequency domain a modulated light source is used to excite the sample.

$$L(t) = A \cdot \sin(\omega t)$$

Due to the samples fluorescence exponential response (Equation 1.2), the re-emitted light is of a different modulation and phase.

$$RE(t) = A' \cdot \sin(\omega t - \theta)$$

The excitation and re-emission response waveforms are given in blue respectively red in Figure 1.5. The re-emission has a lower amplitude and a shifted phase. The amplitude is lower because not all injected photons are necessarily re-emitted.

According to [21] the lifetime follows from the shifted phase $\theta$ by first applying a differential equation:

$$\frac{dRE(t)}{dt} = -\frac{RE(t)}{\tau} + L(t)$$

\[1.5a\]
Solving the equation results in

\[ \tau = \frac{\tan(\theta)}{\omega} \]  

(1.6)

so \( \tau \) can be directly estimated from the measured phase shift. The intensity of the fluorescence is somewhat harder to find being a fraction of the different modulations, w.r.t. the phase shift.

Extracting this information gives lifetime estimates and above all it can give fast results \([20–22]\).

### 1.3.2 Time domain

In time domain, the emitted pulse is reconstructed. This pulse follows the exponential decay \(^2\) as Equation 1.2

\[ I(t) = I_0 \cdot e^{-\frac{t}{\tau}}. \]

The lifetime extraction can be done with different techniques, e.g. by exponential fitting on this pulse or centre of mass methods. A pulsed excitation source has to be used to see the fluorescence exponential emission.

To reconstruct the pulse, two main techniques are used in time domain: Gating and Time Correlated Single Photon Counting (TCSPC). According to \([20]\) TCSPC is the superior way of reconstructing the pulse in time domain. As TCSPC reconstructs the pulse from single photons, it is faster compared to gating which needs multiple integration periods, one for each timeframe.

**Gating**

Figure 1.6(a) displays the gating technique used on FluoCAM. Although this description is specific to this system, it is similar to other gating systems. In general a very small gate (tens of picoseconds) is used. The gate in FluoCAM is dependent on the clock frequency, here 100 MHz, making a 5 ns gate. This gate is shifted over the entire fluorescence pulse to reconstruct the signal step by step. Only the photons in the gate are counted on every shift.

---

\(^2\)For simplicity a mono-exponential decay is assumed here, however there is a convolution with the Instrument Response Function (IRF) as described in Chapter 5 and the decay can be multi-exponential.
On FluoCAM this reconstruction is done by the use of two counters: $C_0$ and $C_2$.

As the laser is triggered, the sample emits its fluorescence exponential light, which is focused on the camera. With delaylines the laser trigger signal is delayed in small steps. This pulse is used as switch between $C_0$ and $C_2$. Part of the arrived photons are stored in $C_0$ and the remainder in $C_2$. Shifting the switch between $C_0$ and $C_2$ will cause a larger part of the photons to arrive in the other counter.

An integration time of some milliseconds is needed to accumulate enough photons in the counters after which a shift to the next delay is done. At present the integration time per frame is 1 sec.

After shifting over the whole delayline range, a histogram, containing a representation of the original waveform can be constructed. Taking the value in $C_2$ on a delay and subtracting the $C_2$ from the previous delay, gives the height of the bin. This measure is taken with respect to the intensity of both this frame and the previous one.

$$Signal(timeframe_n) = \frac{C_2(timeframe_{n})}{Intensity(timeframe_{n})} - \frac{C_2(timeframe_{n-1})}{Intensity(timeframe_{n-1})}$$  \hfill (1.7)

The intensity on every frame is given by the sum of $C_0$ and $C_2$, i.e. the integral over the whole pulse. For an integration time that is unequal to 1 second, the value $C_0 + C_2$ should be divided by the integration time to get a measurand in Hz.

$$Intensity(timeframe_{n}) = \frac{C_0(timeframe_{n}) + C_2(timeframe_{n})}{t_{int}} Hz$$  \hfill (1.8)

with $t_{int}$ the integration time per timeframe.

To be able to build the whole pulse, a total integration time of at least some minutes is needed. This can lead to problems as the ICG sample is exposed for a long time with a high intensity light source, causing
1.4. SINGLE PHOTON AVALANCHE DIODES

e.g. photobleaching effects. This in turn has a huge impact on the measured intensity and therefore also on the pulse shape, therefore a correction with intensity is done.

The acquisition process:
1. Sending laser light to the ICG sample;
2. The ICG sample emits photons depending on its lifetime;
3. The emitted photons are received on the camera;
4. Depending on the active counter, the photons are stored in C0 or C2;
5. This process is repeated over all delay values of the delayline (or a specified range);
6. The signal can finally be computed using Equation 1.7 and 1.8.

**Time Correlated Single Photon Counting**

TCSPC in Figure 1.6(b) gives each photon that arrives a time stamp, instead of using gates and counting the photons that arrive in a window.

With a Time to Digital Converter (TDC), the time between the arrival of the photon and the laser trigger pulse is measured. This time $\Delta t$ is stored in the bins of a histogram. For each possible $\Delta t$ a histogram bin exists, which is incremented every time $\Delta t$ is measured.

$$Bin(\Delta t) = Bin(\Delta t) + 1 \quad (1.9)$$

After an integration time of some tens of milliseconds, the histogram with the original pulse is ready. TCSPC is thus a much faster technique w.r.t. gating to estimate the shape of the original pulse.

The intensity of the signal is given by the integral over the histogram divided by the integration time.

$$Intensity = \frac{\sum_{T=\Delta t_{\text{min}}}^{\Delta t_{\text{max}}} Bin(T)}{t_{\text{int}}} \text{ Hz} \quad (1.10)$$

The acquisition process:
1. Sending laser light to the ICG sample;
2. The ICG sample emits photons depending on its lifetime;
3. The emitted photons are received on the camera;
4. For each photon hit, the time between the SPAD pulse and the laser trigger is calculated with a TDC;
5. The timings from the TDC are stored in a histogram using Equation 1.9.

With either Gating or TCSPC the signal / histogram is generated, after which the lifetime has to be extracted using fitting methods etc.

1.4 Single Photon Avalanche Diodes

For the time domain fluorescence lifetime measurement techniques, especially TCSPC, a very accurate photon detector is required, preferably able to measure individual photons. [23] gives an overview of single photon detectors: photomultiplier tubes (PMTs), single photon avalanche diodes (SPADs), Microchannel Plates (MCPs), quantum dots and superconducting wires. The only device that can be easily implemented in silicon and does not require ultra low temperature is a SPAD.

The SPAD is a solid state photodetector: a reverse biased p-n junction as shown in Figure 1.7(a). The diode is protected with a guard ring to prevent breakdown at the edges of the device. As the voltage on the diode is above breakdown, a very high electric field is present on the junction. An incoming photon will create an avalanche with very high gain (of the order of $10^6$), causing the current of the device to rapidly rise, with an edge below 500 ps. In order not to destroy the device, the SPAD has to be quenched, with the circuit of either Figure 1.7(b) or (c). [24]
The SPAD response does not depend on the number of photons impinging or of their wavelength. It has a digital behaviour: either one or several photos are sensed and an intrinsic current pulse is observed, or no photon is sensed and no response is observed.

A passive quenched SPAD with a resistor is the basic quenching system. Compared to a transistor, active, quenched SPAD the dead time is longer as is the SPADs rise time. Both are improved with the active quenching system and buffering of the SPAD output. This digital SPAD gives a sharp digital pulse with a subnanosecond rise time.

The SPADs dead time is defined as the time after the avalanche to quench the SPAD to its initial voltage. Afterpulsing is caused by trapped carriers initiating another (unwanted) avalanche after the original (intended) avalanche.

Two such SPAD-based camera chips developed in the group are FluoCAM (SPSD) and LinoSPAD. The latter is intended to be integrated in the final medical probe.

1.4.1 FluoCAM

FluoCAM is a prototype lifetime extraction system developed at EPFL. It is one of the few systems that uses the gating technique to find the lifetime.

The camera consists of three boards as can be seen in Figure 1.8. The camera is composed of a $60 \times 48$ SPAD array; each pixel contains two 8 bit counters and a switch. The camera is mounted on the FPGA board (Figure 1.8(a)), on which the main processing is done. [25]

The delaylines on the second board (Figure 1.8(b)) have a resolution of around 10 ps, allowing a high time accuracy. The third board (Figure 1.8(c)) is to regulate the power on the other boards.

A fan mounted besides the boards cools the system. As both the FPGA and especially the delaylines can heat up to $60^\circ$, they can be cooled towards room temperature. This decreases noise (DCR) and makes the delaylines resolution more accurate.

The camera is mounted together as shown in Figure 1.9.

In the FPGA no analysis functions are integrated; only the raw values of $C_0$ and $C_2$ are read-out. Therefore all analysis can only be done after the measurement. This and the long integration time of around 8 minutes per measurement make a realtime system impossible.
1.4. SINGLE PHOTON AVALANCHE DIODES

(a) FPGA board with camera.  (b) Delaylines board.  (c) Power board.

Figure 1.8: The camera consists out of three boards: FPGA (and camera), delaylines and power.

(a) Camera.

(b) Schematic.

Figure 1.9: FluoCAM.

1.4.2 LinoSPAD

LinoSPAD is intended to become a clinical probing solution. A prototype is developed in order to assess its performance and implement the firmware and software.

The LinoSPAD chip consist of a line of 256 pixels, as shown in Figure 1.10. The pixels are shown in Figure 1.7(c), the outputs are directly connected to the chips bond pads. The digital functionality is not integrated in the chip, but in a following FPGA. The chip is compared with the camera chip used for FluoCAM in Table 1.1. The main advantage of LinoSPAD is the increased fill factor: $55 \times$ higher than on FluoCAM. The SPADs themselves are smaller limiting the effective active area increase to a factor 4.3 per pixel.

The main limitation of LinoSPAD is it being a line instead of a 2D array. Although this simplifies the chips structure and output, more complex optics is needed to scan for a 2D image.

(a) LinoSPAD, a SPAD line with 256 pixels.

Figure 1.10: LinoSPAD, a SPAD line with 256 pixels.
Table 1.1: Comparison of the SPAD chips from FluoCAM and LinoSPAD.

<table>
<thead>
<tr>
<th>Pixel performance</th>
<th>FluoCAM (SPSD)</th>
<th>LinoSPAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAD area [mm$^2$]</td>
<td>5.1 × 4.1</td>
<td>6.1 × 0.024</td>
</tr>
<tr>
<td>SPAD layout</td>
<td>60 × 48</td>
<td>256 × 1</td>
</tr>
<tr>
<td>Pitch [µm]</td>
<td>85</td>
<td>24</td>
</tr>
<tr>
<td>Fill factor [%]</td>
<td>0.8</td>
<td>44</td>
</tr>
<tr>
<td>Active SPAD area [µm$^2$]</td>
<td>57.8</td>
<td>253.4</td>
</tr>
<tr>
<td>Quantum efficiency @ 800 nm [%]</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Dead time [ns]</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>CMOS process</td>
<td>0.35 µm</td>
<td>0.35 µm</td>
</tr>
</tbody>
</table>

The chip is bonded on a daughterboard, which will be connected to the LinoSPAD mainboard. The mainboard in Figure 1.11 contains a Xilinx Spartan 6 FPGA and Cypress FX3 USB3 controller as main components. The other components are schematically drawn in Figure 1.12.

The FPGA needs to contain TDCs and histogrammers in order to build histograms directly on the FPGA. This makes a fast analysis possible after the read-out of the histograms. As integration times in the order of tens of milliseconds are sufficient, realtime lifetime information can be given.

Figure 1.11: LinoSPAD mainboard containing a Spartan 6 FPGA and FX3 USB3 controller.

Figure 1.12: LinoSPAD mainboard.
1.5 Camera systems

Camera systems for both single point and 2D fluorescence (lifetime) imaging exists. An overview of some commercial fluorescence (lifetime) systems is given in Table 1.2.

Table 1.2: Comparison of fluorescence measurement systems

<table>
<thead>
<tr>
<th>Camera system</th>
<th>Lifetime range [ns]</th>
<th>Accuracy [ps]</th>
<th>Camera type</th>
<th>Camera resolution</th>
<th>Time / meas.</th>
<th>Lifetime Technique</th>
<th>Light source</th>
<th>Wavelength [nm]</th>
<th>Power [mW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>FluoCAM (prototype)</td>
<td>0.1-1.5</td>
<td>&lt; 100</td>
<td>SPAD</td>
<td>60 × 48</td>
<td>8 min</td>
<td>Gating</td>
<td>Laser</td>
<td>790</td>
<td>1.5</td>
</tr>
<tr>
<td>LinoSPAD (prototype)</td>
<td>0.1-1.5</td>
<td>&lt; 100</td>
<td>SPAD</td>
<td>256 × 1</td>
<td>30 ms</td>
<td>TCSPC</td>
<td>Laser</td>
<td>790</td>
<td>1.5</td>
</tr>
<tr>
<td>Lifa (microscopy)</td>
<td>0.05-300</td>
<td>100</td>
<td>CCD</td>
<td>1392 × 1024</td>
<td>70 ms</td>
<td>Frequency</td>
<td>Laser / LED</td>
<td>405-630</td>
<td>20-300</td>
</tr>
<tr>
<td>MicroTime 200 (microscopy)</td>
<td>0.1-10</td>
<td>?</td>
<td>SPAD</td>
<td>1 scanning</td>
<td>1 s</td>
<td>TCSPC</td>
<td>Laser</td>
<td>375-900</td>
<td>?</td>
</tr>
<tr>
<td>Picosecond (spectroscopy)</td>
<td>1-10,000</td>
<td>15</td>
<td>Streak scope</td>
<td>1</td>
<td>50 ns</td>
<td>Frequency</td>
<td>Laser</td>
<td>375-980</td>
<td>10-100</td>
</tr>
<tr>
<td>PicoMaster (spectroscopy)</td>
<td>0.02-1000</td>
<td>1</td>
<td>PMT</td>
<td>1</td>
<td>&gt; min</td>
<td>TCSPC</td>
<td>Laser</td>
<td>375-670</td>
<td>?</td>
</tr>
<tr>
<td>EasyLife TCSPC (spectroscopy)</td>
<td>0.1-3000</td>
<td>?</td>
<td>PMT</td>
<td>1</td>
<td>&gt; min</td>
<td>TCSPC</td>
<td>LED</td>
<td>185-820</td>
<td>?</td>
</tr>
<tr>
<td>IVIS (pre-clinical)</td>
<td>NA</td>
<td>NA</td>
<td>CCD</td>
<td>2048 × 2048</td>
<td>1 s</td>
<td>NA</td>
<td>Lamp</td>
<td>420-740</td>
<td>150 W (filtered)</td>
</tr>
<tr>
<td>FLARE (clinical)</td>
<td>NA</td>
<td>NA</td>
<td>CCD</td>
<td>640 × 480</td>
<td>200 ms</td>
<td>NA</td>
<td>LEDs</td>
<td>700-800</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>SPY (clinical)</td>
<td>NA</td>
<td>NA</td>
<td>CCD</td>
<td>?</td>
<td>33 ms</td>
<td>NA</td>
<td>Laser</td>
<td>806</td>
<td>2.7 W</td>
</tr>
<tr>
<td>Photodynamic Eye (clinical)</td>
<td>NA</td>
<td>NA</td>
<td>CCD</td>
<td>?</td>
<td>?</td>
<td>NA</td>
<td>LEDs</td>
<td>760</td>
<td>?</td>
</tr>
<tr>
<td>Fluostick (clinical)</td>
<td>NA</td>
<td>NA</td>
<td>CCD</td>
<td>720 × 576</td>
<td>40 ms</td>
<td>NA</td>
<td>Laser</td>
<td>750</td>
<td>150</td>
</tr>
<tr>
<td>LinoSPAD (clinical)</td>
<td>0.1-1.5</td>
<td>&lt; 100</td>
<td>SPAD</td>
<td>256 × 256</td>
<td>30 ms</td>
<td>TCSPC</td>
<td>Laser</td>
<td>790</td>
<td>?</td>
</tr>
</tbody>
</table>

? = Not specified, NA = Not Available in this system.

As summarized in Table 1.2, there is a large variation between the systems and their purpose. Some systems focus on point measurements (spectroscopy) and others on making 2D images, with both high and low resolutions. Camera modules include SPADs, CCDs, PMTs and specifically designed optical elements. Systems can operate at multiple wavelengths, dependent on the laser / filter used. Measurements for lifetime are done in frequency domain or time domain. Time domain’s TCSPC technique is dominant.

A short description of the different systems is given per category. It shows what current commercial systems can do in terms of lifetime calculation range and accuracy. Their resolution and the frame speed are also important parameters to compare with the developed LinoSPAD system. Especially for the clinical systems details are missing as this information is not released by the companies. Those systems are therefore a reference to the clinical applications only, furthermore those systems do not have the lifetime extraction ability.

The lifetime systems light sources can be split in two, namely lasers and LEDs. Whereas lasers can provide a sharper, faster, impulse response, laser systems can reach lower lifetimes. Using LEDs makes a cheaper system possible, while having less range and usually also less accuracy.

---

3A measurement is the acquisition of one (lifetime) image.
1.5.1 Microscopy: Lifa / Microtime 200

The best examples of currently (commercially) available fluorescence lifetime imaging systems are to be found in microscopy systems, such as the Lifa and Microtime 200.

Lifa [26], Figure 1.13(a), is a microscopy system from Lambert Instruments that uses a CCD and measures the lifetimes in the frequency domain. It has a high resolution and high number of frames per second. The systems light source is a laser, however no wavelengths in the infrared can be selected. Therefore such a system would be incompatible with ICG.

Another example is the Microtime 200 from PicoQuant [27] in Figure 1.13(b). This microscopy system uses SPADs and therefore measures in time domain using TCSPC technique. It is more like the LinoSPAD system described in this document than Lifa. Scanning is needed as there is only one SPAD available, therefore acquisition times are with 1 second longer than Lifa. A laser in the infrared is available as to be able to measure with e.g. ICG.

(a) Lifa.  
(b) Microtime 200.

Figure 1.13: Fluorescence Lifetime Imaging systems in Microscopy.

1.5.2 Spectroscopy: Picosecond / PicoMaster / Easylife TCSPC

A second category of fluorescence lifetime estimation systems are to be found in spectroscopy. However spectroscopy systems are not designed for imaging, they only give the fluorescence lifetime of one point.

Here again systems in the frequency domain are found, such as Picosecond C11200 by Hamamatsu [28], shown in Figure 1.14(a). It is based on a streak scope, apparently a Hamamatsu proprietary measurement device, of which no detailed information is given. While using the frequency domain it is the fastest system shown here. It has a large lifetime range up to several microseconds. The system uses a laser as its illumination source, also in the near infrared.

Two time domain systems employing TCSPC are PicoMaster from Photon Technology International [29] and EasyLife TCSPC from the Optical Building Blocks Corporation [30]. The systems are displayed in Figure 1.14(b) respectively Figure 1.14(c). The PicoMaster claims a lifetime accuracy of 1 ps, EasyLife doesn’t give a specification, this makes PicoMaster the most accurate system described. Both systems use Photomultiplier Tubes (PMTs), they are similar to SPADs, however not having the ability to see single photons.
1.5. CAMERA SYSTEMS

1.5.3 (Pre-)Clinical: IVIS / FLARE / SPY / PDE / Fluostick / LinoSPAD

Current (pre-)clinical systems don’t have the ability to extract the fluorescence lifetime, they only provide an intensity image. All systems are equipped with CCD cameras.

The IVIS built by PerkinElmer [31] is a pre-clinical system for mouse imaging, shown in Figure 1.15(a). Therefore the system also provides mouse anaesthesia. Instead of a laser or LEDs, a lamp is used, from which the light is filtered. This makes a large range of wavelengths possible and due to a high lamp power of 150 W images can be made fast and for different excitation wavelengths.

Examples of systems that are already used in surgery are FLARE from Israel Beth Deaconess Hospital [32], SPY by Novadaq [33], PDE (Photodynamic Eye) from Hamamatsu / Pulsion [34] and Fluostick by Fluoptics [35]. As can be seen in Figure 1.15(b) and (c) the FLARE and SPY systems are large apparatus.
which makes in body imaging less ideal.

The PDE is more like the goal of the project; a small probe like camera system as in Figure 1.15(d). Unfortunately little information is released regarding the system specifications. An even smaller fluorescence probe is the Fluostick in Figure 1.15(e). It has the ability of making realtime fluorescence intensity images with a resolution of $720 \times 576$. In the probe a laser head is integrated.

LinoSPAD will be transferred to a medical probe, the same chip will be used. The boards need to be miniaturized to fit inside the probe, but will contain the same FPGA and FX3 controllers. The rest of the system has to fit inside the probe as well, including the scanning optics and the NIR light source.

The probe with the main components is drawn in Figure 1.16. The surgeon can hold the probe and scan the part of the patients body where the cancer is located. This will tell the surgeon exactly where the cancer is spreading and what to remove.

![Figure 1.16: LinoSPAD probe for clinical applications, including the miniaturized prototype components plus optics and light source.](image)

**Outline**

This thesis focusses on the firmware and software development for LinoSPAD, therefore it is divided into three main parts: Firmware development Part I, Software development Part II and finally preliminary lifetime measurements in Part III.

In Part I first an overview of Time to Digital Converters is given in Chapter 2 and the best possible TDC for inside an FPGA is selected. Around the TDC the remainder of the system is built in Chapter 3. Finally the system is calibrated in Chapter 4 and its performance is tested.

The second part focusses on the implementation of the read-out and analysis software. In Chapter 5 the data analysis functions are described and the accuracy is estimated with simulations. In Chapter 6 the implementation of the read-out and the GUI are discussed.

In the last part some preliminary measurement results are shown with both FluoCAM in Chapter 8 and LinoSPAD in Chapter 9. However the system, which is in great lines identical for both cameras, is described first in Chapter 7. Here the measurements are described and how they are executed.

From the above work, the conclusions are finally presented in Chapter 10. Suggestions for further work are also given.
Part I

Firmware development

Implementation of TCSPC in a Spartan 6 FPGA for Fluorescence Lifetime Imaging using LinoSPAD

VHDL firmware was written and optimized for multichannel; adapted from Fishburns work for TDC implementation. Histogrammers to store time-events and read-out with FX3 are newly developed.
Chapter 2

Time to Digital Converters

Converting the timing information from the individual photons into digital information is the most important part of the system. For this purpose a Time to Digital Converter (TDC) has to be implemented. The LinoSPAD chip offers no additional functionality, i.e. there are no hardware TDCs available. Therefore the TDCs have to be implemented inside the FPGA.

This directly implies that the possible choices in TDC design are limited to what the FPGA can do. A FPGA is not optimized for large scale asynchronous designs and therefore inconsistent delays are considered irrelevant. However for a TDC the delays are important as it influences the timings.

From the design brief in Appendix A, the most important parameters for the design can be extracted:

- Resolution < 100 ps
- Range = 10 ns
- Throughput = high, as TDCs are shared

In this chapter various TDCs that can be implemented in a FPGA are studied and compared in Section 2.1. Specialized hardware designs are not taken into account as they cannot be integrated in a FPGA.

Finally one design is taken to be implemented in the LinoSPAD FPGA in Section 2.2

2.1 TDCs overview

A TDC converts a time difference into a digital code. The idea is displayed in Figure 2.1. Both a start and stop signal are needed in order to calculate a timing. The difference in time between the rising edge of the start and stop signal is $\Delta T$. $\Delta T$ is converted into a digital code that can be processed and stored.

![Figure 2.1](image)

Figure 2.1: A TDC converts the time between the arrival of the start and stop signal into a digital code.

2.1.1 Time Counter

A straightforward TDC implementation is starting a counter on the arrival of the start signal and stopping it on the stop pulse [36, 37]. This is schematically shown in Figure 2.2. A high frequency clock is needed to drive the counter. The timing resolution that can be achieved is directly proportional to the clock
frequency:

\[ Resolution = \frac{1}{f_{CLK}}. \] (2.1)

In order to achieve a resolution of 100 ps, a clock frequency of 10 GHz is needed. In a FPGA, clock speeds of more than 1 GHz can normally not be reached. Furthermore using high clock speeds will linearly increase the systems power consumption. A stable and low jitter clock is needed.

On the contrary the systems area consumption will be small, only consisting of one counter. Above all the non linearities of the system will be small as only a counter is used.

\[ \text{start} \]

\[ \text{stop} \]

\[ \text{digital time stamp} \]

\[ Q_{\text{enable}} \]

\[ \text{counter} + 1 \]

\[ \text{clock} \]

\[ D \]

\[ Q \]

\[ CLK \]

\[ \text{Figure 2.2: Time counter, clock is a high frequency reference. Redrawn from [37]} \]

It has been shown by [38] that it is feasible to implement a time counter inside an FPGA, using clock interpolation techniques. Resolutions up to 100 ps can be achieved.

2.1.2 Delayline

A solution that does not involve high frequency clocks is based on displacing the start signal in time [36, 37, 39]. The start signal is delayed with small delays, usually inverters or buffers. The design is displayed in Figure 2.3. The resolution of the TDC is dependent on the component delay:

\[ Resolution = \tau. \] (2.2)

\[ \text{start} \]

\[ \text{stop} \]

\[ D \]

\[ Q \]

\[ CLK \]

\[ \text{Q1} \]

\[ \text{Q2} \]

\[ \text{Q3} \]

\[ \text{QN} \]

\[ Q \]

\[ D \]

\[ \text{clock} \]

\[ \text{enable} \]

\[ \text{counter} + 1 \]

\[ \text{digital time stamp} \]

\[ \text{Figure 2.3: Tapped delayline TDC, start signal is displaced in time; stop signal freezes the time stamp.} \]

The output is a thermometer timestamp, therefore this has to be decoded to binary in order to make it compact. The area consumption will largely depend on the measurement range needed. Adding more blocks also increases non linearities as the delays cannot be well controlled, especially inside FPGAs.

To increase the resolution, to below the component delay, a double delayline approach can be chosen [36, 40]. Instead of delaying only the start signal, also the stop can be delayed, as shown in Figure 2.4. This increases the resolution to

\[ Resolution = \tau_1 - \tau_2. \] (2.3)

Therefore \( \tau_1 > \tau_2 \). Very high (down to some picoseconds) resolution can thus be achieved. A larger area and higher energy consumption compared to a single delayline are in contrary required.

With delayline based TDCs high throughput can be achieved as a conversion per clock cycle is possible.
2.1.3 Oscillator / Pulse Shrinking

Some more comprehensive techniques include oscillator and pulse shrinking methods [37, 41, 42].

An oscillator based TDC uses two ring oscillators with slightly different frequencies $f_{\text{slow}}$ and $f_{\text{fast}}$ as shown in Figure 2.5. The start and stop signals trigger the ring oscillators to start oscillating. The start oscillator is already running when the stop oscillator starts, therefore the rising edge of both signals is on a different time. As there is a difference in frequency, the edges will come closer. Counting the number of cycles it takes before the rising edge of the stop oscillator matches the edge of the start oscillator gives a measure for the time difference between start and stop based on the frequency difference of the oscillators.

A pulse shrinking TDC, Figure 2.6, uses a time attenuator or pulse shrinker. The input pulse has a width: the difference between start and stop signals arrival times. This width is shrunk in the $\alpha$-OR gate by a factor $\alpha$. The delayline after the OR is to ensure that the feedback pulse arrives later than the width of the original pulse. Every loop, the pulse is shrunk by the $\alpha$ factor until it disappears. Every loop, the counter is incremented on the rising edge of the pulse, again until disappearance. The number of cycles the pulse can make before disappearing gives the measure for the time between start and stop.

According to [37] both methods can achieve high resolutions with relatively low area consumption com-
pared to delayline TDCs. However the throughput is lower as multiple clock cycles are needed to measure the timings. Furthermore frequencies have to be accurate and stable for the oscillator TDC. The \( \alpha \) factor in the pulse shrinking TDC has to be well known and controlled.

Implementations in FPGAs are possible [43], however as the frequency of the oscillation should be well known and controlled. Therefore exact placement and precise calibration is needed. Making a large system with other functionality can be more complicated as precise control is more difficult.

2.1.4 Comparison

A comparison of the different TDC design is given in Table 2.1. Comparing the different design options with the listed requirements and the fact that a FPGA implementation is required, the time counter first falls off. The time counter needs a too high clock frequency to achieve the required resolution, this is not possible inside a FPGA.

The double delaylines can be implemented, however thorough calibration of the delays in both lines is needed. Furthermore a range of 10 ns is required, with two delaylines, this range would use a huge number of stages, compared to a single delayline.

Ring oscillator structures require very precise frequencies as the resolution is dependent on this. In FPGAs the delays and thus the final frequencies cannot be well controlled and can vary a lot between different FPGAs. This problem is also apparent with delaylines, however less problematic as there is no feedback.

Pulse shrinking can also be implemented, however the \( \alpha \)-factor has to be made with a structure inside the FPGA. Furthermore it should be stable and well known.

Both ring oscillator and pulse shrinking methods require multiple clock periods and are therefore harder to use in a multichannel system.

As a result, a single delayline approach is best in the system. It has relatively high resolution, the range can be easily controlled by adding more stages and the throughput is high, as it can do one conversion per clock cycle. Above all, this design is best used while sharing the TDC among multiple SPAD inputs.

Inside the FPGA the implementation of such delayline can be done in the carry structure, which is a long chain of the same blocks, with a more or less stable delay.

Another advantage of using the carry structure is the fact that the synthesizer can automatically place the blocks on the right places on top of each other. This allows for easy prototyping and adaptations of the firmware.

Some disadvantages however are the implemented carry look ahead, which is used to make fast arithmetic operations possible. This carry look ahead changes delays between subsequent stages and can even pass stages. The carry look ahead is assumed to behave every conversion in the same way. Therefore swapped delays caused by carry look ahead can be overcome by manually swapping the output pins.

2.2 TDCs for the LinoSPAD system

As discussed, the best feasible TDC implementation is that of a simple delayline based FPGA TDC. Instead of buffers or inverters, the Carry4 blocks of the FPGA are used to make a delayline. The original idea was proposed by [44] and later used and enhanced by [45, 46] among others.

The start signal is the OR of a number of SPADs, as the TDC needs to be shared among multiple SPAD inputs. A detailed description of the input filter’s implementation is given in Section 3.1.

The interest lies in relative timing information and not absolute timing information. The timing position from or towards the next laser trigger has to be found, not taken into account that the start signal may come from another clock period. Therefore the implementation can be simplified, there is no need to count multiple clock periods with a counter.

Furthermore the laser clock is 100 MHz, sufficient to be used as clock source for the FPGA logic.
Table 2.1: Comparison of possible TDC implementations.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Time counter</th>
<th>Single Delayline</th>
<th>Double Delaylines</th>
<th>Oscillator</th>
<th>Pulse shrinking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time counter Scales with $f_{CLK}$</td>
<td>Component delay</td>
<td>Difference in component delays</td>
<td>Difference in oscillation $f$</td>
<td>$\alpha$ factor</td>
<td></td>
</tr>
<tr>
<td>Non linearity performance</td>
<td>Only one counter</td>
<td>Delayline balancing</td>
<td>Delayline balancing + matching</td>
<td>Frequency stability</td>
<td>$\alpha$ stability</td>
</tr>
<tr>
<td>Range</td>
<td>Counter size</td>
<td>Delayline length + Coarse counter</td>
<td>Difference in delayline length + Coarse counter</td>
<td>Counter size</td>
<td>Counter size</td>
</tr>
<tr>
<td>Throughput</td>
<td>One conversion / CLK period</td>
<td>One conversion / CLK period</td>
<td>One conversion / CLK period</td>
<td>One conversion / CLK period</td>
<td>One conversion / CLK period</td>
</tr>
<tr>
<td>Multiplexing</td>
<td>Easily integratable</td>
<td>Easily integratable</td>
<td>Easily integratable</td>
<td>More involved</td>
<td>More involved</td>
</tr>
<tr>
<td>FPGA feasibility</td>
<td>No, too low $f_{CLK}$</td>
<td>Yes, carry structure can be used</td>
<td>Difficult, as hard to get delay match</td>
<td>Difficult, both implementation and multiplexing</td>
<td>Difficult, both implementation and multiplexing</td>
</tr>
</tbody>
</table>

This implies that the TDC stop signal is derived from the laser clock and the dedicated clock routing network in the FPGA can be used for all latches in the TDC.

The final TDC that was implemented in the Spartan 6 FPGA is shown in Figure 2.7. A detailed discussion is given in Section 3.1.

![Figure 2.7: Carrychain delayline TDC, OR of SPADs as start is displaced in time; laser clock as stop signal freezes the time stamp.](image-url)
Chapter 3

Time Correlated Single Photon Counting - Spartan 6 implementation

This chapter will focus on the implementation of a multichannel TCSPC system inside a Xilinx Spartan 6 FPGA. The required building blocks are presented in Section 3.1, from which the total system is built and described in Section 3.2.

3.1 Components

The building blocks are designed in VHDL and implemented in the Spartan 6 using the Xilinx ISE.

The main components for TCSPC are the TDC and the Histogrammer. The general system consists of several TDC banks with multiple input channels to be connected to the LinoSPADs. Each bank has one TDC and up to 64 inputs are multiplexed on this one TDC.

3.1.1 Input filter

As the LinoSPAD chip’s SPADs are directly connected to inputs of the FPGA, a filter is needed to shorten the SPAD pulses and get the address of which SPAD has fired.

As a complex input filter will add additional jitter and noise on the input pulses, the filter should be kept as simple as possible. However the filter has to function as a door for the SPADs into the system and therefore the door should always be able to open.

The functionality of the input filter is schematically displayed in Figure 3.1. The SPAD pulses need to be shortened till the next rising clock edge, as to allow a new hit (of another SPAD) in the next clock edge. The address of the fired SPAD has to be stored in the next clock cycle as well.

![Figure 3.1: Timing diagram of the Input filter.](image-url)
Filter with feedback reset

The filter with feedback reset employs only two flip flops. The design is shown in Figure 3.2. As soon as a SPAD fires, the first flip flop is set. The pulse directly goes into the OR gate towards the TDC.

On the rising edge, the signal is set in the second latch, creating the address bit and also resetting the first flop.

As there are delays between the rising edge of the clock and the reset triggered on the first flop, pulses firing directly after the clock cannot be seen.

![Figure 3.2: Input filter with reset feedback.](image)

Filter with self reset

The input filter with a self resetting flip flop is more complex compared to feedback reset, three flip flops and an inverter are needed. The design is in Figure 3.3.

As soon as a SPAD fires, the first flop is set. This will trigger the flop to directly reset itself and also reset the second flop. On the second flop, a ‘0’ appears at the output, which is converted to a ‘1’ by the inverter.

On the next rising clock edge, the signal is stored and a ‘1’ is directly propagated. This is turned into a ‘0’ by the inverter and the system is in standby mode again.

As the signal coming out of the first flop needs to reset both itself and another flop, it has to be ensured there is enough time to also reset the second flop.

To increase the reliability an enhanced design with one AND gate for the reset signal on the first flop can be used. The proposed design can be seen in Figure 3.4.

The first flop is in this case only reset when the second flop is set, causing the signal to always propagate.

![Figure 3.3: Input filter with self reset.](image)

![Figure 3.4: Input filter with robust self reset.](image)
3.1.2 TDC

The Time to Digital Converter is the main core of the TCSPC system. It converts a time into a digital thermometer code, which in term needs to be converted to a digital code.

The resolution of the TDC, i.e. the smallest measurable time window, depends on the architecture and in the case of a FPGA on the structure that is already there. To achieve a high resolution, a small delay between each element is needed.

The TDC is implemented in a FPGA by using the carrychains, which have a dedicated routing structure for the carries into the next slice. As mentioned by [44] resolutions down to 17 ps can be reached in FPGA devices, [45] shows even better results, with resolutions down to 10 ps.

However the resolution is strongly dependent on the FPGA device class. The Spartan 6 FPGA is in the lower end and therefore the inner systems delays are larger compared to the FPGAs used by [44, 45].

A large carrychain is implemented to fill one clock period, i.e. 10 ns. The structure is drawn in Figure 3.5(a) when there is no input hit. On a rising input, the signal is propagated through the carrychains as Figure 3.5(b).

On the rising edge of the next clock, the current values of the propagating carries are stored in latches. This value can in turn be transformed into a binary number.

![Carrychain in rest.](image)

![Carrychain with propagating input.](image)

**Figure 3.5:** Propagation of an input signal through a carrychain.

However the FPGAs carrychain structure has some properties that are not so helpful for the implementation of a delayline.

The main concern is the slice structure. Each slice contains one Carry4 block, i.e. 4 adders in one slice. This makes the delay in the slice smaller than the inter slice delay.

Secondly the blocks have a fast carry look ahead, a system implemented in adders to make the adder perform faster. However for a delayline it implies that the signal can hop some codes and arrives earlier in the next carry block.

Thirdly the FPGAs structure is divided into several clock regions, in which the clock is equally distributed. However because of the length of the delayline, multiple clock regions are used. This makes the clock arrival time over the whole line uneven.

All phenomena add to the non linearities of the line, i.e. unequal delay times.
3.1.3 Thermometer to Binary decoder

The next step is to convert the thermometer output of the TDC into a binary code. A thermometer to binary decoder is implemented with a pipelined design in order to allow one conversion per clock cycle. E.g. the thermometer code ‘0000111’ will be converted into a binary code ‘011’.

**Decoder principle**

There are different ways of making a thermometer to binary decoding. A comparison of different decoding implementations is given by [47]. [47] claims that a multiplexer based decoder has the lowest resource consumption and easily scales to larger number of bits. Both are important aspects for the FPGA decoder. The principle of this binary decoder is based on divide and conquer.

The middle bit of the thermometer code is the most significant bit (MSB) for the final binary result. Furthermore it serves as a select for the upper or lower part. When it is high, the upper part is selected, otherwise the lower.

This technique is repeated till the last three bits which form the least significant bit (LSB) and the one before.

A pipelined design example of a thermometer to binary decoder is given in Figure 3.6(a). The design of a (in this case 15 to 4 bit) thermometer decoder employs mux stages to convert the code in several clock cycles to binary.

This allows for one conversion per cycle, enabling high throughput. In contrary, because of the added registers the area consumption will be increased.

![Thermometer decoder](image)

*Figure 3.6: Implementation of a pipelined thermometer decoder.*

Taking the example of above ‘0000111’. The middle bit is selected: ‘0’. The lower part is therefore muxed and the middle of this is taken ‘1’. Therefore the LSB will be the upper ‘1’. The result is ‘011’.
Bubble correction

However due to the TDCs behaviour also codes as ‘0000101’ are possible. This code will result in binary code ‘001’ with the decoder of Figure 3.6(a). The ‘0’ in between the ‘1’ s is called a bubble and needs to be corrected.

One way to repair the bubble is with the circuit of Figure 3.7. In the case of a bit being ‘0’ while the next is ‘1’, the circuit artificially creates a ‘1’ in the ‘0’ s place.

This creates the code ‘0000111’, where no more bubbles are present, i.e. the same code as before the bubble was introduced.

Delayline & Bubbles

In the case of the delayline, the ‘0’ can be caused by a flop not going to ‘1’ on the rising clock edge. In this case bubble correct places a correct ‘1’.

The ‘0’ can however also be from the carry look ahead module, in which case there should not be a ‘1’. Then bubble correct places a false ‘1’, that should not be there.

Therefore the design of Figure 3.6(b) is introduced. In this design, the last stage utilizes a counter instead of the muxes.

This allows to count the raw input and results in binary code ‘010’ for thermometer code ‘0000101’. The same as the number of bits that are high.

In the final design a 512 to 9 bit thermometer to binary decoder is implemented. The 16 latest bits are counted instead of muxed to remove the bubbles. The bubbles normally occur close to the transition in a range of 4 bits around this transition. Therefore a 4 bit counter (covering exactly the 16 remaining bits) is sufficient.

![Figure 3.7: Bubble correction circuit. Based on [48].](image)

3.1.4 Histogrammer

In order to store the acquired time values and make a TCSPC histogram, a histogrammer is implemented as Figure 3.8. The acquired binary time code acts as an address for a RAM. The number of RAM addresses corresponds to the total number of possible time codes.

After the Histogrammer is enabled, the current address’s (time code’s) value is read. ‘1’ is added to this value and the value is written back on the same address.

In order to allow one storage operation per clock cycle, the design is again pipelined. Due to the latency of the RAMs and the adder multiple stages are needed in between the operations. (This is not indicated
on the schematic of Figure 3.8).

The histogrammer of Figure 3.8 has to be implemented 256 times, one for each channel. This leads to a large consumption of resources, as each histogrammer would have e.g. an adder to add one to the value. However, only one channel can be active at each clock period, so some part of the components can be shared among the channels, just as the TDCs themselves are shared.

Therefore a multiplexed design was implemented as shown in Figure 3.9. The adder can be shared among multiple channels, i.e. 2 / 4 / 8 etc. This relaxes constraints on the resource occupancy, however it leads to more complex routing in the design. Therefore a trade-off between sharing and routing complexity has to be made.

This histogrammer not only allows one storage operation per clock cycle, but also acts to reduce read-out complexity.

As the histograms only need to read-out after an integration time, data transmission times can be reduced. Also the amount of data is smaller compared to the raw data of sending both time code and spatial information each clock cycle.

Development of a Multichannel TCSPC System in a Spartan 6 FPGA
3.1.5 Read-out

After an integration time, the histograms need to be read out. An USB3 read-out is implemented.

To be able to send data through the USB a Cypress EZ-USB FX3 module is placed besides the FPGA. The interface is schematically drawn in Figure 3.10 with the main connections.

![Figure 3.10: LinoSPAD interface with FPGA, FX3 and the SPAD chip.](image)

The FX3 is connected with a 32 bit bus to the Spartan 6. The FX3 outputs four flags to the FPGA. The FPGA controls the transmission from and towards the FX3 with the select signals. The read-out of the system is initiated from the host. The FX3 outputs a flag indicating the FPGA can sent data. The FPGA activates the write mode by setting select write. Each clock cycle, one 32 bit word is written into the FX3 buffer. A flag from the FX3 indicates the buffer is full and nearly full. The FPGA waits till the FX3 switches to a second buffer and starts writing again. At the end of the process the select write line is deactivated.

To send data to the FX3 from the host, a flag is set in the FX3 that data is available. The FPGA acknowledges this flag by assigning both select output enable en select read. The values are read one by one from the FX3 buffer.

The implementation of the read-out on the host side and the structure of the data is discussed in Chapter 6.

3.2 Systems overview

A complete schematic of one TDC bank is given in Figure 3.11, a larger version in Appendix B. On the schematic the different components can be identified.

On the left the inputs are the SPADs which are directly connected to the input filters. Behind the input filters the signals go into a large OR gate to route the signals into one TDC. On the same time, the address is stored, i.e. which SPAD has fired in the last clock cycle.

If more than one SPAD fires in the same clock period, the timing value will be that of the first pulse. However the address has multiple high SPADs and thus it is not known from which SPAD the time value originates. In this case, the outcoming time value will be ignored and not assigned to any histogrammer. This crosstalk phenomena is studied in more depth in Section 4.4.

After the OR gate, the input signal follows the Carry4 delayline, which value is frozen on the next clock edge. This value is a thermometer code which is converted into binary in several pipeline stages as to do one conversion per cycle. The address is latched in the same way as to arrive at the same time with the outcomeing binary value at the input of the histogrammers.

The address assigns one of the histogrammers, in which the value corresponding to the current time stamp is increased.
3.2. SYSTEMS OVERVIEW

The total system consist out of the parts given in Table 3.1. There are 8 TDCs each consisting of 512 delay stages, i.e. 128 Carry4 blocks. Each TDC has a corresponding thermometer to binary decoder coding 512 bits into 9. The 256 memories, one for each channel, have again 512 memory addresses, each of 16 bits. Therefore a total number of counts of $512 \times 65535$ can be stored per channel (assuming a pure uniform histogram).

To ensure maximum stability of the system and decrease the change of problematic routing, the TDC delaylines are fixed to constrained FPGA slices. The fixed positions makes sure the delaylines are not placed next to each other, but with space in between. This also diminishes the change of seeing ‘interference’ as will be discussed in Section 4.5.

<table>
<thead>
<tr>
<th>Component</th>
<th>Number</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input filters</td>
<td>256</td>
<td>Each channel has its own input filter</td>
</tr>
<tr>
<td>OR gates</td>
<td>8</td>
<td>One OR gate per channel bank</td>
</tr>
<tr>
<td>TDCs (delayline + thermometer decoder)</td>
<td>8</td>
<td>8 TDC, thus 32 SPADs per bank</td>
</tr>
<tr>
<td>Histogrammers</td>
<td>256</td>
<td>Each channel has its RAM, other components shared among 4 channels</td>
</tr>
<tr>
<td>Read-out</td>
<td>1</td>
<td>One FX3 read-out system</td>
</tr>
<tr>
<td>Clock management</td>
<td>1</td>
<td>One clock managment tile</td>
</tr>
</tbody>
</table>

Table 3.1: Overview of the components included in the final (prototype) LinoSPAD system.
Chapter 4

Calibration & Performance Evaluation of the FPGA TDCs

The compiled and implemented system has to be calibrated. The performance of the TDCs has to be found. Therefore different methods are used to find the TDCs resolution (Section 4.1), jitter (Section 4.2) and non linearities (Section 4.3).

While employing multiple channels, crosstalk occurs when multiple SPADs fire simultaneously. Crosstalk and possible solutions are mentioned in Section 4.4. Furthermore the presence of ‘interference’ is noted in the case the system inside the FPGA is to dense. Interference is shown and discussed in Section 4.5. Delay from the input filters till the TDC entrance is introduced with the large OR gate as discussed in Section 4.6.

Finally the system is compared in terms of TDC performance with some other FPGA TDCs in Section 4.7 and different multichannel configurations are presented in Section 4.8.

4.1 Resolution

The resolution can be checked both in simulation and from hardware.

4.1.1 Simulation

The TDC was simulated with a post place and route simulation, i.e. a simulation in which the delays of the device are taken into account. In behavioural simulations the delayline doesn’t function as the delays are not taken into account.

A pulse is injected in the TDC and the outcoming delay code is stored, the pulse is shifted with 1 ps etc. This process is repeated over the full range of 10 ns, thus generating 10.000 delay codes.

The simulated delay codes of one post place and route simulated TDC are shown in Figure 4.1. The resolution of the TDC can be calculated using the slope of the linear fit.

\[
\text{Resolution}_{\text{LinSPAD}} = \frac{1000}{-\text{slope}} = \frac{1000}{46.97} = 21.3 \text{ ps.}
\] (4.1)

The large horizontal steps that can be observed in Figure 4.1 come from clock domain crossings (smaller steps, half way in between large steps) and from empty spaces in the FPGA structure (large steps).

The clock domain crossings causes little delay difference. However the crossing of the empty spaces as indicated on Figure 4.2 cause substantial delays and therefore are main contributors to DNL / INL.

\footnote{The sign conversion is due to the fact that the TDCs inside the FPGA measure till the next rising clock edge, instead of from the previous.}
4.1. RESOLUTION

In order to find the resolution of the TDCs when implemented, a reference is needed. Therefore the delaylines of the FluoCAM are used as their resolution is known to be around 10 ps (Section 8.1).

The 100 MHz clock of the laser is fed into the FluoCAM delaylines and this clock signal is delayed. The difference between the delayed clock and the clock itself are measured on the TDCs inside the

4.1.2 Hardware

Figure 4.1: Simulation of LinoSPAD delay codes w.r.t. inserted delay times.

Figure 4.2: Part of the implemented delayline.
FPGA.

This setup is shown in Figure 4.3.

![Figure 4.3: Setup with FluoCAM to measure FPGA TDCs resolution and jitter.](image)

Repeating the experiment with different delays gives a direct measure for the conversion of a FluoCAM delay code into a LinoSPAD delay code. From Figure 4.4 this conversion can be seen to be a straight line from which the resolution can be calculated.

![Figure 4.4: Measured LinoSPAD delay code vs. FluoCAM delay code.](image)

As the resolution of FluoCAM is known to be 10 ps and the slope of the line is measured as -0.49, LinoSPADs resolution follows to be

\[
\text{Resolution}_{\text{LinoSPAD}} = \frac{\text{Resolution}_{\text{FluoCAM}}}{-\text{slope}} = \frac{10}{0.49} = 20.4 \text{ ps}.
\]

(4.2)

Comparing the 21.3 ps from Equation 4.1 and the acquired 20.4 ps in Equation 4.2 gives a difference in resolution of 1.1 ps between simulation and hardware. This can be caused by the uncertainty in the FluoCAM resolution of 10 ps (dependent on temperature) and furthermore due to the difference between hardware and simulation.

While in simulation the delays are fixed for the specified FPGA, the delays change from device to device. Therefore some deviation might be expected.

### 4.2 Jitter

As simulations yield no jitter, this can only be found in the real device. Therefore the pulses used to find the resolution are also used to find the jitter.
One of the pulses is shown in Figure 4.5. The histogram does not show a perfect Gaussian shape, because of the non-linearities of the TDC. However, the fitted Gaussian gives a good indication of the jitter in the measurement.

![Figure 4.5: Measured pulse and a Gaussian fit to estimate jitter.](image)

The estimated sigma of all measured pulses is shown in Figure 4.6. The jitter is expected to be less near the start of the delay line w.r.t. to the end. As each block in the line adds jitter, more blocks passed towards the end automatically increased the jitter.

This behavior is visible in the figure, the jitter goes down for increasing FluoCAM delay code, which corresponds to the lower LinoSPAD delay codes.

![Figure 4.6: LinoSPAD jitter (σ) vs. FluoCAM delay code.](image)
The average jitter can be seen to be 2.6 LSB, which corresponds to a jitter of around 60 ps.

Main causes of jitter are in the signal, the clock, the TDC circuit and the FluoCAM delaylines. The jitter’s $\sigma$ can therefore be defined as:

$$\sigma_{\text{jitter, total}} = \sqrt{\sigma_{\text{jitter, signal}}^2 + \sigma_{\text{jitter, clock}}^2 + \sigma_{\text{jitter, FluoCAM delaylines}}^2 + \sigma_{\text{jitter, LinoSPAD TCSPC}}^2}$$ (4.3)

in which the only unknown is the $\sigma_{\text{jitter, LinoSPAD TCSPC}}$.

As the signal is derived from the clock, the jitter is identical and the equations can be simplified to

$$\sigma_{\text{jitter, total}} = \sqrt{2 \cdot \sigma_{\text{jitter, clock}}^2 + \sigma_{\text{jitter, FluoCAM delaylines}}^2 + \sigma_{\text{jitter, LinoSPAD TCSPC}}^2}$$ (4.4)

in which

$$\sigma_{\text{jitter, total}} = 60 \text{ ps}$$
$$\sigma_{\text{jitter, clock}} = 5 \text{ ps}$$
$$\sigma_{\text{jitter, FluoCAM delaylines}} = 10 \text{ ps}$$

The clock and FluoCAM delaylines jitter are neglectable due to the squares, resulting in a

$$\sigma_{\text{jitter, LinoSPAD TCSPC}} \approx 60 \text{ ps}.$$

### 4.3 Non linearities

The differential and integral non linearities can be estimated again both through simulation and in hardware experiments.

#### 4.3.1 Simulation

Using the plot of Figure 4.1, the DNL can be extracted by taking the difference from the average step size. The average step size is the total of 10,000 injected timings divided by the number of outcoming delay codes ($490-20 = 470$ codes).

$$\text{DNL}(n) = \frac{\text{Codes in step } n}{\text{Average codes / step}}.$$ (4.5)

The INL is the integrated DNL and therefore

$$\text{INL}(n) = \sum_{i=1}^{n} \text{DNL}(i).$$ (4.6)

The resulting DNL and INL are plotted in Figure 4.7. The large steps that were seen in Figure 4.1 are here visible as large DNL peaks. Those peaks (except for the largest peak, which is a clock domain crossing), are periodic around every 64 codes. Every 16 slices, a gap can be seen inside the FPGA layout. This gaps introduces a longer routing path and therefore a longer delay.

Defining the peak to peak DNL and INL as

$$\text{DNL}_{\text{pk-pk}} = \max(\text{DNL}) - \min(\text{DNL})$$ (4.7a)
$$\text{INL}_{\text{pk-pk}} = \max(\text{INL}) - \min(\text{INL})$$ (4.7b)

gives 6.2 respectively 8.2 for $\text{DNL}_{\text{pk-pk}}$ and $\text{INL}_{\text{pk-pk}}$. 
4.3. NON LINEARITIES

4.3.2 Hardware

For hardware characterisation of the implemented FPGA delaylines, another setup compared to Figure 4.3 is needed. In order to estimate the non linearities, random times have to be injected inside the delaylines, in order to cover its full range.

For this purpose a ringoscillator is used as depicted in Figure 4.8. The ringoscillator is running at a frequency different compared to the TDC clock. This makes the arrival of the ringoscillator random w.r.t. the clock. The measured random timings are again stored in a histogram.

If the system is totally linear and the ringoscillator is completely random w.r.t. the clock, totally random timings are expected. This would result in a white (flat) noise histogram, however due to non linearities the histogram is not flat as can be seen in Figure 4.9. The histogram is not flat, but each bin has a different size due to the non linearities.

Therefore the DNL can be measured from the histogram using

$$ DNL(n) = \frac{counts(n)}{\sum \text{counts}} - 1 \quad (4.8) $$

in which $counts(n)$ is the number of counts inside one histogram bin and $\sum \text{counts}$ the average bin value.

The INL is still defined as Equation 4.6

$$ \text{INL}(n) = \sum_{i=1}^{n} DNL(i). $$

![Figure 4.7: Simulated DNL & INL of one TDC delayline.](image)

![Figure 4.8: Setup with a ringoscillator to measure FPGA TDCs non linearities.](image)
This procedure is done on the 8 TDCs that are implemented in the system. The DNL and INL of one of those TDCs is shown in Figure 4.10. The extracted peak to peak DNL and INL as defined in Equation 4.7 are extracted in Table 4.1. Comparing the different TDC shows variations between them in DNL and INL. W.r.t. the simulations the results are averagely better. However the TDC 2 has with an INL of 10.3 almost double INL compared to TDC 1.

The non linearities are mainly caused by the internal FPGA structure and the clock routing network. The carrychain structure that is used shows non optimal delays between adder blocks. Secondly the carry...
4.4 CROSSTALK

Table 4.1: Comparison of DNL & INL between simulation and hardware; implementation with 8 TDCs.

<table>
<thead>
<tr>
<th>Implemented TDC</th>
<th>DNL pk-pk [LSB]</th>
<th>INL pk-pk [LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>6.2</td>
<td>8.2</td>
</tr>
<tr>
<td>1</td>
<td>3.4</td>
<td>5.5</td>
</tr>
<tr>
<td>2</td>
<td>3.8</td>
<td>10.3</td>
</tr>
<tr>
<td>3</td>
<td>3.8</td>
<td>6.4</td>
</tr>
<tr>
<td>4</td>
<td>4.3</td>
<td>6.0</td>
</tr>
<tr>
<td>5</td>
<td>3.7</td>
<td>7.7</td>
</tr>
<tr>
<td>6</td>
<td>4.2</td>
<td>6.2</td>
</tr>
<tr>
<td>7</td>
<td>4.7</td>
<td>9.4</td>
</tr>
<tr>
<td>8</td>
<td>3.4</td>
<td>7.9</td>
</tr>
<tr>
<td>Avg.</td>
<td>3.9</td>
<td>7.4</td>
</tr>
</tbody>
</table>

lookahead imposes timing jumps.

The clock is routed in clock regions. Inside those regions, the clock is routed with the smallest differences between slices. However the clock routing is not equalized in between regions. Therefore timing variations may occur there. As the proposed delayline uses 124 carry blocks, there are 7 clock region crossings.

4.4 Crosstalk

Till here a discussion on pure TDC performance was given, however the TDC is shared among multiple SPADs. This sharing can cause crosstalk if multiple SPADs fire in the same clock period. In the case of simultaneous hits, i.e. two or more SPADs that fire in the same period, the timing information of the first SPAD to fire is measured. However the spatial information is that of all SPADs that have fired in the period.

This results in the timing information of the first fired SPAD being stored in the histograms of all fired SPADs. The results in the other histograms are therefore not correct. This phenomena can be seen in Figure 4.11. The random generated signal from Figure 4.11(a) is also seen in (b). The signal in (b) is ideally a sharp peak as the signal is a delayed clock.

![Channel A](image1)

(a) Ringoscillator random signal.

![Channel B](image2)

(b) Delayed clock pulse.

Figure 4.11: Crosstalk as part of the signal from (a) is seen in (b) as well.

To solve the before mentioned problem, measured time values will be ignored in the case of multiple SPADs firing in the same period. This results in no more crosstalk as seen in Figure 4.12. There is no
more signal blending from Figure 4.12(a) into (b).

In the case many SPADs fire each period, another filtering method should be used as a lot of information will be lost.

Figure 4.12: Crosstalk removal, no part of the signal from (a) is seen any more in (b).

### 4.5 Interference

Besides crosstalk, multiple SPADs firing at the same time, ‘interference’ can occur. An example of this interference happening inside the LinoSPAD FPGA is shown in Figure 4.13. Although the pulses should be similar to the result of Figure 4.12, the delayed clock pulse is swapped with a part of the random timings from the ringoscillator.

The delayed pulse can be clearly observed in the middle of the random signal in Figure 4.13(b), whereas this gap in the random signal is seen in (a). The number of counts in (a) and in the delayed clock pulse are similar. This indicates they are totally exchanged.

Figure 4.13: Interference, the delayed clock pulse is swapped with a part of the random ringoscillator signal.

The interference patterns were only seen in the case TDCs are close to each other. Therefore the position of the TDCs has been fixed in the design, constraining them to fixed slices.

The fixing to slices is shown in Figure B.1. The TDCs are depicted as large blue lines. They are at a fixed distance of ten slices next to each other.

---

LinoSPAD - Fluorescence Lifetime Imaging for Fluorescence Guided Surgery
4.6 Delay

In the TDC, the delay of a carry block is around 21 ps, as found in Section 4.1. However the delay from the input filter to the TDC is important as well. This delay introduces blindness, while the design is assumed to have no delay between filter and TDC entrance.

In the case of only a wire this is true and the delay is small compared to the TDCs range of 10 ns. However in the case of sharing the SPADs, multiple input filter’s outputs are combined into an OR before entering the TDC. The input filter and the OR gate are schematically drawn in Figure 4.14. The delay paths from the output of the input filter into the OR gate and into the addressing latch are drawn in red.

![Figure 4.14: Delay paths from input filter into TDC through OR gate and from input filter to address.](image)

This OR gate in the Spartan 6 FPGA is built from 6 input Lookup Tables (LUTs). It introduces delay, up to some nanoseconds. In this time window, the TDC is blind. An example implementation for an OR gate with 16 inputs is given in Figure 4.15(a). For a 32 inputs OR, a possible design is in Figure 4.15(b). Increasing the number of inputs increases the logic path length and the ORs complexity. For the 32 inputs OR many designs are possible and only one possibility is given here. Therefore the delay depends on the synthesizers OR generation.

![Figure 4.15: LUT6 OR gate.](image)
To account for this delay some artificial delay has to be introduced in the address path as indicated in Figure 4.14 with the δ delay block. This delay should match the delay of the OR gate in order to have no blindness. Some buffers can be inserted in the addressing path to compensate for the delay.

Introducing delays in the addressing path has some unwanted side effect that crosstalk again appears, but only in the time window of the introduced delay.

### 4.7 FPGA TDCs Comparison

The presented multichannel TDC is compared with other FPGA TDCs. As comparison are taken one of the first FPGA TDCs by Favi [44], Fishburn [45] and CERN [46]. They all use the same Carry4 structure. Table 4.2 compares the main system specifications.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan 6</td>
<td>Virtex 5</td>
<td>Virtex 6</td>
<td>Spartan 6</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>65</td>
<td>40</td>
<td>45</td>
</tr>
<tr>
<td>Resolution [ps]</td>
<td>21</td>
<td>17</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>Jitter [ps]</td>
<td>60</td>
<td>24</td>
<td>14</td>
<td>26</td>
</tr>
<tr>
<td>$DNL_{pk-pk}$ [LSB]</td>
<td>3.9</td>
<td>4.6</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>$INL_{pk-pk}$ [LSB]</td>
<td>7.4</td>
<td>5.6</td>
<td>6</td>
<td>?</td>
</tr>
<tr>
<td>Carry4 blocks</td>
<td>128</td>
<td>50</td>
<td>40</td>
<td>124</td>
</tr>
<tr>
<td>Range (fine) [ns]</td>
<td>11.8</td>
<td>3.4</td>
<td>1.6</td>
<td>12.9</td>
</tr>
<tr>
<td>TDCs inside FPGA</td>
<td>8</td>
<td>1</td>
<td>160</td>
<td>1</td>
</tr>
<tr>
<td>Total nr. of channels</td>
<td>128-256</td>
<td>1</td>
<td>160</td>
<td>1</td>
</tr>
<tr>
<td>Clock frequency [MHz]</td>
<td>100</td>
<td>300</td>
<td>600</td>
<td>125</td>
</tr>
</tbody>
</table>

Comparing the clock frequencies of the systems, the Fishburn TDC runs fastest at 600 MHz. A faster clock reduces the clock period and therefore the length of the carrychain can be decreased. Decreasing the number of Carry4 blocks used, improves the non lineairties, because of a shorter line. This advantage can only be used in the higher end Virtex FPGAs. The Spartan is limited to clock frequencies of maximum 400 MHz, whereas the Virtex’s can go to 550-600 MHz.

Secondly the design has to allow the high clock speed. As the LinoSPAD design employs much more logic than a pure TDC, the routing is more complicated and therefore the speed has to be reduced. More logic is required to implement the multichannel / multiplexing system and the histogrammers after the TDC. The clock speed of 100 MHz makes the use of a longer carrychain required. Expected are higher non lineairties compared to shorter carrychains as shown. Furthermore the DNL and INL are increased as a lower end FPGA is used.

The resolution is determined by the FPGA structure as well. The Spartan cannot run at higher clock speeds, therefore the synchronous system does not need to be fast. Therefore the resolution in the Spartan is lower compared to the Virtex devices. They allow faster clock, therefore the FPGA should be performing faster and the delay between elements smaller.

The resolution of the LinoSPAD system is indeed comparable to the CERN TDC, which is implemented in a Spartan as well.
4.8 Multichannel configurations

Different multichannel system configurations have been tested. The previously presented results were achieved with a $8 \times 16$ channel system. Other configurations and their stabilities are presented in Table 4.3.

The majority of the 128 channel systems perform stable and with a small amount of static noise. However multiplexing 256 inputs on one TDC makes the routing too complex and the system cannot perform at the required clock speed.

Compared to the stable 128 channel systems, 256 is much more problematic. Although the system $4 \times 64$ operates, the histograms produced are noisy and there is a lot of variation, especially offset, in between the different channels. The systems get too dense in terms of routing for optimal performance. Therefore the complete system should be even more simplified, or the systems clock speed should be decreased.

Table 4.3: Tested multichannel configurations.

<table>
<thead>
<tr>
<th>TDCs</th>
<th>Channels / TDC</th>
<th>Total channels</th>
<th>Feasible</th>
<th>Stability</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>128</td>
<td>128</td>
<td>No</td>
<td></td>
<td>Too complex routing, difficult clock performance</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>128</td>
<td>Yes</td>
<td>+</td>
<td>2$\times$64 more stable</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>128</td>
<td>Yes</td>
<td>0</td>
<td>Very stable with little variation over the channels</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>128</td>
<td>Yes</td>
<td>++</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>256</td>
<td>256</td>
<td>No</td>
<td></td>
<td>As $1 \times 128$</td>
</tr>
<tr>
<td>2</td>
<td>128</td>
<td>256</td>
<td>No</td>
<td></td>
<td>As $1 \times 128$</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>256</td>
<td>Yes</td>
<td>0</td>
<td>Much more static noise compared to $8 \times 16$</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>256</td>
<td>Yes</td>
<td>-</td>
<td>Noisy and dirty histograms</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>256</td>
<td>Yes</td>
<td>--</td>
<td>Too dense, not stable</td>
</tr>
</tbody>
</table>

Currently the most stable and therefore usable configuration consists of 8 TDCs each with 16 channels multiplexed. However for the 256 channel system some measures have to be taken to ensure a stable and noise free system. Therefore either the routing density needs to be reduced or a part of the system can run at a lower clock frequency. Introducing a ring of empty slices around the delaylines (a guard ring) as proposed by [45] doesn’t improve either the stability or the static noise.
Part II

Software development

USB 3 Read-out & Data analysis

Development of the analysis and read-out software using C++ and QT.
Study of different analysis algorithms and comparison in terms of speed and accuracy.
Chapter 5

Theoretical Lifetime study

The development of the software is divided into two main objectives. First a study of potential data analysis tools is made in this chapter. Whereas the implementation itself together with the read-out and GUI are discussed in Chapter 6.

The main goal of this study is to find a reliable way of measuring fluorescence lifetime. It has to be both fast and ideally needs a low number of photons.

First the signal is build using Matlab, Section 5.1. Thereafter the signal is analysed with the Wiener Filter Section 5.2 and a Centre of Mass Method Section 5.3.

A comparison in terms of speed and accuracy is given in Section 5.4. Noise and non lineairities influences are finally studied in Section 5.5 and Section 5.6.

5.1 Histogram generation

The TCSPC histogram is built in Matlab by using the instruments response function (IRF) as a basis, on top of the IRF exponential decays are added over which the counts are randomly distributed.

5.1.1 Instrument Response Function (IRF)

Figure 1.6 shows the general shape of a fluorescence signal. From literature and measurements it is known that this signal is de facto a convolution between an exponential decay and the IRF of the setup.

The laser pulse would ideally be a Dirac function, from which all photons arrive at the same time on the camera. The IRF however causes fluctuations of the time of arrival of those photons. This fluctuation is due to the laser, the optics and the electronics, which plays a role in the final signal shape. The fluctuations are characterized by measuring the Instrument Response Function (IRF).

Therefore the simulation starts with inserting the IRF, as measured, into the model. The IRF used is taken from measurements done on FluoCAM (Subsection 1.4.1). The IRF is measured by replacing the ICG sample (Figure 1.6) with a mirror. It shows the distribution of the arrived raw laser photons over time.

The obtained IRF can be found in Figure 5.1. The full width at half maximum (FWHM) of the IRF pulse is around 240 ps. The presence of a laser queue can be noticed in the longer roll of after the semi-Gaussian pulse. The bobbles are mainly caused by the FluoCAM delaylines.

5.1.2 Creation of the Fluorescence Signal

The histogram or fluorescence signal is built in simulation starting with the IRF. According to the model the number of photons that arrive on the fluorescence sample are equally distributed over the IRF, i.e.

\[ N_{\text{bins}(i)} = N_{\text{total}} \cdot IRF_{\text{bins}(i)} \]  \hspace{1cm} (5.1)

with \( N_{\text{total}} \) the total number of photons received in one measurement, \( N_{\text{bins}(i)} \) the total number of photons in one time window (\( \text{bins}(i) \)) corresponding to the normalized value of \( IRF_{\text{bins}(i)} \) in the same time window.
5.1. HISTOGRAM GENERATION

Figure 5.1: Instrument Response Function as measured with FluoCAM.

From each time window the photons will arrive according to an exponential decay over which they are randomly distributed. The exponential decay is defined by the fluorescence lifetime (as described by Equation 1.2). The process is visualized in Figure 5.2. The IRF as in Figure 5.1 is taken as a reference; on each time window an exponential is drawn, from which the photons randomly arrive on the SPADs. The amount of photons generated over this exponential curve is $N_{bins(i)}$. This results in times of arrival

$$\text{Times}_{\text{arrivals}} = bins(i) + \text{random}(e^{-\frac{t}{\tau}}, N_{bins(i)})$$

in which $bins(i)$ functions as an offset for the photons generated in that particular bin and the random function generates $N_{bins(i)}$ random times from the exponential with a lifetime of $\tau$.

The histogram is finally computed by summing the times in different bins, with a width: the resolution of the TDC.

Some examples of histograms generated with this method are given in Figure 5.3. Histograms (a) and (b) are generated with a total of 50,000 photons, the bins have a width of 20 ps, as this is around the resolution of the TDCs in the LinoSPAD system. Histograms (c) and (d) show 2500 photons and added noise (DCR).

As the pulses are generated, a lot of parameters can be changed, e.g. the number of photons (intensity or measurement time), the injected lifetime, the bin width (resolution), the noise and the non linearities. The influence of those parameters on the resulting lifetimes was studied.
Figure 5.2: Instrument Response Function + Random Exponentials on some time windows with visualized photon hits as red dots.

Figure 5.3: Example histograms with different intensities and lifetimes.
5.2 Lifetime - Wiener filter

From the generated pulse, multiple ways of extracting the lifetime were examined. This section studies lifetime extraction with the Wiener filter.

The simulated histograms need to be deconvoluted to be able to make an exponential fit on the data and extract the lifetime from the signals. A fit could be directly made on the histogram data, however as the IRF changes the exponential behaviour and thus the lifetime, this gives off results. Especially for lifetimes < 1 ns. Therefore those results can be satisfactory for larger lifetimes of over 1-1.5 ns, far above the pulse width of the laser.

5.2.1 Direct Deconvolution

Deconvolution is basically the same procedure as for convolution, only with a division instead of a multiplication. That is basically also the main limitation, as divisions by very small values can lead to infinity. This results in noisy and unsmooth curves.

The basics is

$$\hat{Hist} = \mathcal{F}(Hist)$$ (5.3a)

$$\hat{IRF} = \mathcal{F}(IRF)$$ (5.3b)

$$\hat{Deconv} = \frac{\hat{Hist}}{\hat{IRF}}$$ (5.3c)

$$Deconv = \mathbb{R}\mathcal{F}^{-1}(\hat{Deconv})$$ (5.3d)

with $Hist$ the histogram data / generated fluorescence signal and the $IRF$ being the measured Instrument Response function. $Deconv$ is the deconvoluted signal, which should be an exponential. For the Fourier transforms, the Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) are used.

The result for the deconvolution of a generated pulse with 50,000 photons, $\tau = 500$ ps and a bin width of 20 ps (Figure 5.3(b)) is given in Figure 5.4.

![Image](image_url)  

**Figure 5.4:** Result from direct deconvolution using the FFT and IFFT.
The deconvoluted signal is shown in red, however it is totally noisy and non-exponential. Therefore a solution is needed to smooth the signal first; the Wiener constant is introduced in the deconvolution. This is the Wiener filter.

### 5.2.2 Wiener filter

The goal of the Wiener filter is to divide always by a certain value, i.e. to smooth the curve in the process and make the final signal less noisy \[49\]. However because a certain constant is added to the division in the deconvolution, an offset is introduced. Therefore the error on the extracted lifetime is increased. However compared to the normal deconvolution in which no lifetime at all is found, the method shows close results.

The process of Wiener deconvolution is

\[
\hat{\text{Hist}} = \mathcal{F} (\text{Hist}) \\
\hat{\text{IRF}} = \mathcal{F} (\text{IRF}) \\
\hat{\text{Deconv}} = \frac{\hat{\text{Hist}} \cdot \hat{\text{IRF}}}{|\text{IRF}|^2 + \text{Wiener Constant}} \\
\text{Deconv} = \Re \mathcal{F}^{-1}(\hat{\text{Deconv}})
\]  

(5.4a) (5.4b) (5.4c) (5.4d)

In this case, the value of the Wiener Constant is crucial for finding a correct lifetime. The value should be as small as possible in order not to create a large offset as compared to the signals themselves. However it cannot be made to small or the problems of the normal convolution reappear.

From Figure 5.5 it can be seen the smoothness of the signal is increased with a larger Wiener value (Wiener constant 500 compared to 1). However from the extracted lifetime an offset of over 100 ps is found. In the case of a 500 ps injected lifetime, the measurement error is \(> 20\%\). Therefore a trade-off between error and smoothness has to be found.

Taking generated histograms with different lifetimes, the lifetime is extracted from the pulse with different Wiener constants. The process is done for both a high number of photons of 50,000 and a low number of 2500.

The extracted lifetimes are presented in Table 5.1. From the extracted lifetimes the absolute average error can be calculated. The average error has an optimal minimum for Wiener constant 50. For higher Wiener constant the average error rapidly increases due to the offset introduced. For the lower Wiener constant, the signals are not smooth causing the deviation. Therefore the Wiener constant should be chosen around 50 to have a low error and smooth signals.

<table>
<thead>
<tr>
<th>Inserted lifetime</th>
<th>50,000 photons</th>
<th>2500 photons</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau = 100) ps</td>
<td>110, 137, 171</td>
<td>105, 135, 169</td>
</tr>
<tr>
<td>(\tau = 250) ps</td>
<td>232, 249, 295</td>
<td>244, 251, 291</td>
</tr>
<tr>
<td>(\tau = 500) ps</td>
<td>474, 509, 554</td>
<td>434, 480, 560</td>
</tr>
<tr>
<td>(\tau = 750) ps</td>
<td>715, 759, 820</td>
<td>674, 772, 844</td>
</tr>
<tr>
<td>(\tau = 1000) ps</td>
<td>961, 1000, 1070</td>
<td>952, 1040, 1110</td>
</tr>
</tbody>
</table>

Average error [ps]: 26, 11, 62, 40, 24, 75
5.2. LIFETIME - WIENER FILTER

Figure 5.5: Result from Wiener deconvolution.

5.2.3 Exponential Fitting

On the deconvoluted signal, an exponential fit has to be made. Several fitting methods exist. The standard Matlab exponential fit is based on a non linear Least Squares exponential fit within an iterative process for finding the optimal curve. The problem of this fitting method is its slowness and the difficulty of rewriting into C++.

However taking the ln (natural logarithm) of the exponential data results in a linear line, which makes a (simpler) linear fit a possibility. In the case of a simple linear line

\[ y = at + b \]  

(5.5)
a and b have to be found in order to make y as close as possible to the measured data \( y' \). The values of a and b can be found with a least mean squares method. The general idea is to minimize

\[
\sum_{n=1}^{N} (y(n)' - y(n))^2
\]

(5.6)

the least squares.

According to [50] the solution of the least squares problem is found to be

\[
a = \frac{\sum_{n=1}^{N} (t(i) - i) \cdot (y'(i) - \bar{y})}{\sum_{n=1}^{N} (t(i) - i)^2}
\]

(5.7a)

\[
b = \bar{y} - a \cdot i
\]

(5.7b)

The lifetime of the (normalized) exponential (Equation 1.2)

\[I(t) = e^{-\frac{t}{\tau}}\]

of which the ln is taken

\[\ln(I(t)) = \ln\left(e^{-\frac{t}{\tau}}\right) = -\frac{t}{\tau}\]

is found to be from the linear line:

\[\tau = -\frac{1}{a}.
\]

(5.8)

In order to test the speed and accuracy of Matlab’s exponential fit vs. this Linear fit, both algorithms were benchmarked and Matlab. Noisy exponentials with different inserted lifetimes were generated and fitted against with both algorithms.

In Table 5.2 the algorithms are compared in terms of speed and accuracy. For each inserted lifetime 100 exponentials were generated, the difference between average extracted lifetime and inserted lifetime has been taken as measure for the algorithms accuracy in %.

<table>
<thead>
<tr>
<th>Inserted lifetime [ps]</th>
<th>Avg. extracted lifetime [ps]</th>
<th>Speed [ms]</th>
<th>Accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Matlab</td>
<td>LMS</td>
<td>Matlab</td>
</tr>
<tr>
<td>100</td>
<td>102</td>
<td>104</td>
<td>20.6</td>
</tr>
<tr>
<td>250</td>
<td>255</td>
<td>261</td>
<td>20.7</td>
</tr>
<tr>
<td>500</td>
<td>526</td>
<td>522</td>
<td>20.8</td>
</tr>
<tr>
<td>1000</td>
<td>1108</td>
<td>1106</td>
<td>22.7</td>
</tr>
</tbody>
</table>

For the noisy exponentials, the lifetime extraction is with both methods accurate to 90%. The main difference is the speed. Whereas Matlab’s exponential fit takes roughly 21 ms to make one fit, with LMS linear fitting it takes only 0.03 ms. A speed improvement of roughly 700×.

### 5.3 Lifetime - Centre of Mass method

Another approach to extract the lifetime of the pulse is given by the Centre of Mass method. First a mathematical proof is given, after which the same analysis is done as with the Wiener Filter.
5.3.1 Mathematical analysis

The calculation of the Centre of Mass for a pulse is given as

\[
CoM = \frac{\int_0^\infty t \cdot f(t) \, dt}{\int_0^\infty f(t) \, dt}
\]  

(5.9)

For a pure exponential, the pulse is given by Equation 1.2:

\[
I(t) = I_0 \cdot e^{-\frac{t}{\tau}}
\]

The centre of mass over this pulse can be found by integration as shown by [51]

\[
CoM = \frac{\int_0^\infty t \cdot I_0 \cdot e^{-\frac{t}{\tau}} \, dt}{\int_0^\infty I_0 \cdot e^{-\frac{t}{\tau}} \, dt} = \left[\frac{-I_0 \cdot \tau \cdot e^{-\frac{t}{\tau}} \cdot (t + \tau)}{-I_0 \cdot \tau \cdot e^{-\frac{t}{\tau}}}\right]_0^\infty = \frac{I_0 \cdot \tau^2}{I_0 \cdot \tau} = \tau
\]

(5.10)

In the case of an experiment, the pulse can never be measured until \(\infty\). Therefore the integration time should at least be \(7 \times \tau\) [51]. In that case

\[
CoM \approx \tau.
\]

(5.11)

5.3.2 IRF correction

The Centre of Mass for the histograms is defined as:

\[
CoM = \frac{\sum_{n=1}^{N} bin_n \cdot counts_n}{\sum_{n=1}^{N} counts_n}
\]

(5.12)

in which \(bin_n\) is the bin number and \(counts_n\) the value corresponding to that bin. To get \(\tau\) in terms of time, \(bin_n\) has to be multiplied by the resolution of the TDC.

\[
CoM = \frac{\sum_{n=1}^{N} bin_n \cdot Resolution \cdot counts_n}{\sum_{n=1}^{N} counts_n}
\]

In the case of a measured signal however, the pulse is not pure exponential but convoluted with the IRF. Therefore a correction needs to be made. From simulations a correction with the Centre of Mass of the IRF seems to be adequate. Therefore

\[
\tau \approx CoM_{signal} - CoM_{IRF}.
\]

(5.13)

Taking the pulse with \(\tau = 500\) ps and 50,000 photons (still from Figure 5.3(b)), the CoM of the pulse and the IRF can be found in Figure 5.6. Now \(\tau\) is found to be:

\[
\tau \approx 1292 - 790 = 502\) ps.
\]

The resulting lifetime is almost exactly the injected lifetime of 500 ps.

Applying the Centre of Mass method to the same signals as used in Table 5.1, the lifetimes are again extracted and shown in Table 5.3. Comparing the average errors of the CoM and Wiener shows that the
Figure 5.6: Centre of Mass calculation for the 50,000 photons pulse with $\tau = 500$ ps.

Table 5.3: Extracted lifetimes [ps] with the Centre of Mass method.

<table>
<thead>
<tr>
<th>Inserted lifetime</th>
<th>50,000 photons</th>
<th>2500 photons</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau = 100$ ps</td>
<td>99.4</td>
<td>97.3</td>
</tr>
<tr>
<td>$\tau = 250$ ps</td>
<td>249.8</td>
<td>241.7</td>
</tr>
<tr>
<td>$\tau = 500$ ps</td>
<td>502</td>
<td>499.2</td>
</tr>
<tr>
<td>$\tau = 750$ ps</td>
<td>748.6</td>
<td>752.2</td>
</tr>
<tr>
<td>$\tau = 1000$ ps</td>
<td>1000.5</td>
<td>989.9</td>
</tr>
<tr>
<td>Average error [ps]</td>
<td>0.9</td>
<td>4.8</td>
</tr>
</tbody>
</table>

The accuracy of the CoM is always better than the Wiener filter. The average error for Centre of Mass is < 5 ps even for low number of photons.

The Centre of Mass technique is a simple and fast method to find the lifetime. The average error is smaller compared to the Wiener filter; in the case of added noise / non lineairities similar.

Finally the Centre of Mass algorithms can be translated into the firmware system and implemented in the FPGA. This significantly improves the lifetime calculations performance and reduces the data read-out time (in the case of only sending lifetime information and intensity).
5.4 Comparison

A comparison between Wiener and CoM is made using both a high photons count pulse and a low photons count.

For a high number of counts, as Figure 5.7, the pulse will be smooth and closer to an ideal pulse with ∞ counts. Therefore the average error is expected to be lower compared to Figure 5.8.

![Figure 5.7: Wiener vs. CoM, 50,000 photons.](image)

![Figure 5.8: Wiener vs. CoM, 1,000 photons.](image)

From Table 5.4, in which the mean error is calculated, the average error can indeed seen to increase while lowering the intensity.

Secondly the difference between Wiener and CoM can be seen to be substantial; CoM has a 3× smaller error compared to Wiener. Above all the speed of CoM is higher compared to Wiener, which is shown in Table 5.5. The performance boost is up to 100×.

The main speed limitation of the Wiener filter is the need of convolution and deconvolution, which operations are slow compared to normal arithmetic’s.
Table 5.4: Wiener vs. CoM, average error for different intensities over a range of inserted lifetimes.

<table>
<thead>
<tr>
<th>Nr. of Photons</th>
<th>Average error [ps]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wiener</td>
</tr>
<tr>
<td>50,000</td>
<td>9.7</td>
</tr>
<tr>
<td>25,000</td>
<td>12.5</td>
</tr>
<tr>
<td>10,000</td>
<td>14.3</td>
</tr>
<tr>
<td>5,000</td>
<td>26.5</td>
</tr>
<tr>
<td>2,500</td>
<td>33.9</td>
</tr>
<tr>
<td>1,000</td>
<td>52.8</td>
</tr>
</tbody>
</table>

Table 5.5: Wiener vs. CoM, speed on different setups; both algorithms are written in C++ and run 256 times, in parallel if applicable (one SPAD line).

<table>
<thead>
<tr>
<th>Processor</th>
<th>Average time / line [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>I7 3630QM @ 2.40 GHz</td>
<td>6</td>
</tr>
<tr>
<td>Core2Duo L9400 @ 1.86 GHz</td>
<td>32</td>
</tr>
</tbody>
</table>

5.5 Noise (DCR)

In the previous study of Wiener and CoM an ideal system with no noise and non-linearities was assumed. This results in clean pulses on which lifetime extraction is accurate with errors up to 53 ps.

However in practice there is noise, mainly in Dark Counts and background. Furthermore there are non-linearities in the delay lines. Therefore a more detailed overview of the influence of noise and non-linearities is provided.

Taking a light intensity of $5 \cdot 10^4 \text{ Hz}$ and a noise rate of 5 kHz, the same analysis is repeated. The DCR is taken to be 5 kHz as the LinoSPAD chip is expected to have a similar DCR when cooled (Section 9.3). The noise is assumed to be white noise, i.e., flat. The noise is randomly distributed over the clock period and added to the generated histogram. Other noise sources, such as jitter and 1/f noise, are not taken into account.

The Signal to Noise Ratio (SNR) is defined as:

$$ SNR = 20 \cdot 10^{\log\left(\frac{\text{Photons}_{\text{total}}}{\text{Noise}_{\text{hits}}}\right)} \text{[dB]}.$$

(5.14)

Note that this can only be exactly stated in the case of simulations, as in practice the number of photons is increased by the noise counts (not distinguishable). Therefore in practice this can only be estimated. Thus the

$$ SNR = 20 \cdot 10^{\log\left(\frac{5 \cdot 10^4 \text{ Hz}}{5 \cdot 10^3 \text{ Hz}}\right)} = 20 \text{ dB} $$

This is roughly the average SNR expected with a cooled LinoSPAD system as detailed in Section 9.3.

The results of the Wiener filter and Centre of Mass method applied to histograms with noise, is presented in Table 5.6. Now comparing Table 5.6 with Table 5.4 it can be directly seen that the average errors are more random and not as systematic as without noise. The results are averaged over multiple runs as to eliminate a part of the randomness.

The average error goes up and down for increasing number of photons and above all the results of the Wiener filter and CoM are much closer compared to the no noise case. For some cases the average error...
on CoM is even bigger than Wiener’s error.

In the case of noise, both algorithms perform similarly. However CoM still has the advantage of being much faster. The noise floor results in a larger centre value, i.e. a larger lifetime. Therefore the noise floor has to be measured, after which the offset can be compensated for.

Table 5.6: Wiener vs. CoM, average error for different intensities over a range of inserted lifetimes in the case of noise ($SNR = 20$ dB) being present.

<table>
<thead>
<tr>
<th>Nr. of Photons</th>
<th>Nr. of Noise counts</th>
<th>Average error [ps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50,000</td>
<td>5,000</td>
<td>24.8</td>
</tr>
<tr>
<td>25,000</td>
<td>2,500</td>
<td>21.9</td>
</tr>
<tr>
<td>10,000</td>
<td>1,000</td>
<td>20.3</td>
</tr>
<tr>
<td>5,000</td>
<td>500</td>
<td>36.9</td>
</tr>
<tr>
<td>2,500</td>
<td>250</td>
<td>42.0</td>
</tr>
<tr>
<td>1,000</td>
<td>100</td>
<td>62.8</td>
</tr>
</tbody>
</table>

5.6 Non linearities

Finally non linearities are added in the simulation as to test if non linearity corrections on the final system are necessary. In the case the lifetime extraction performance is similar with or without non linearities no software measures have to be taken for correction.

The results with both the same noise floor as introduced in Section 5.5 and non linearities measured from the FPGA TDCs are presented in Table 5.7. Now comparing the results with and without noise, especially in the case of a low number of photons, around the same error is found.

Table 5.7: Wiener vs. CoM, average error for different intensities over a range of inserted lifetimes in the case of noise ($SNR = 20$ dB) and non linearities being present.

<table>
<thead>
<tr>
<th>Nr. of Photons</th>
<th>Nr. of Noise counts</th>
<th>Average error [ps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>50,000</td>
<td>5,000</td>
<td>27.7</td>
</tr>
<tr>
<td>25,000</td>
<td>2,500</td>
<td>30.4</td>
</tr>
<tr>
<td>10,000</td>
<td>1,000</td>
<td>32.3</td>
</tr>
<tr>
<td>5,000</td>
<td>500</td>
<td>29.5</td>
</tr>
<tr>
<td>2,500</td>
<td>250</td>
<td>31.4</td>
</tr>
<tr>
<td>1,000</td>
<td>100</td>
<td>51.4</td>
</tr>
</tbody>
</table>

As the error of the system with or without non linearities is similar, non linearity correction is not necessary. However a slight improvement might be possible doing corrections.

In the end around 5,000 photons would be sufficient for a system that is accurate from 30 ps (Wiener) up to 23 ps (CoM). The system should be calibrated with a known lifetime sample fluorophore, such as ICG in milk or water, to account for possible offsets introduced. Especially in the case for CoM a correction for offset is required as a shift in CoM directly introduces a shift in lifetime.
Chapter 6

Implementation of read-out & GUI

Besides the data analysis algorithms (Chapter 5), a GUI is designed in which the data is graphically displayed (Section 6.2). Besides the GUI, a description of the FX3 read-out is given first in Section 6.1.

6.1 FX3 read-out

The FPGA sends data to the host through the FX3, USB3 controller. The data is sent serially over the USB3 cable, grouped in Bytes. The connection between FX3 and FPGA is a 32 bit bus. The 32 bit data word is divided into four parts:

$\begin{array}{c}
\text{Histogram bin} \\
01001011
\end{array}$

$\begin{array}{c}
\text{Bank} \\
11001
\end{array}$

$\begin{array}{c}
\text{Channel} \\
00010110011100
\end{array}$

$\begin{array}{c}
\text{Histogram bin value} \\
11001 00010110011100
\end{array}$

As there are 512 bins inside a histogram, the representation would be a 9 bit binary value, of this only the 8 lowest bits are sent, forming the first Byte. The second Byte consist of the TDC Bank number and the current Channel number, of which the maximum product is always 256, the amount of SPADs and thus the total amount of histograms.

The value of each bin is represented with the last two Bytes, forming a maximum of 16 bits. So in one histogram bin a maximum amount of 65,535 hits can be stored. This makes the total amount of SPAD timings that can be accumulated $512 \cdot 65,535 > 33$ million.

To read-out one histogram, $512 \cdot 32$ bits or $512 \cdot 4$ Bytes have to be sent. To read-out the complete system, i.e. 256 histograms, the amount of data to sent is:

$$Data = \text{Banks} \cdot \frac{\text{Channels}}{\text{Bank}} \cdot \frac{\text{Bins}}{\text{Bin}} \cdot \frac{\text{Bytes}}{\text{bin}} = 256 \cdot 512 \cdot 4 = 524,288 \text{ Bytes} = 0.5 \text{ MB}. \quad (6.1)$$

The FX3 has buffers of 1024 Bytes that are to be filled by the FPGA before being sent to the host, this means a total of 512 buffers need to be filled, switching in between two buffers in the FX3. One being filled and one being transmitted and vice versa.

6.1.1 USB3 speed

The speed at which the 32 bit interface is driven is maximum 100 MHz. This directly dictates the maximum speed that can be achieved by sending data through the FX3. This speed can be calculated to be

$$Throughput_{\text{theoretical}} = 32 \text{ bits} \cdot 100 \text{ MHz} = 3.2 \text{ Gbit/sec} = 381.5 \text{ MByte/sec}. \quad (6.2)$$

Therefore this 524,288 Bytes could be sent in just 1.31 msec. However this theoretical maximum speed is lowered due to overhead for sending acknowledgements and other control signals.
The speed that the current system can achieve is measured by constantly sending data and measuring the time taken to receive a substantial amount of data. Sending more data lowers the overhead and makes transferring more efficient.

In each transfer a total of 524,288 Bytes + 4096 Bytes are sent. The extra amount of 4096 Bytes is overhead. It is intended to send additional system information.

Performing 80 of those transfers with a total amount of data of around 40 MByte takes 140 msec. Therefore one transfer takes 1.75 msec instead of the theoretical 1.32 msec. An overhead of 0.43 msec is present on each transfer. This reduces the theoretical maximum transfer speed to:

$$\text{Throughput}_{\text{real}} = \frac{1.32}{1.75}\cdot3.2\ \text{Gbit/sec} = 2.4\ \text{Gbit/sec} = 286\ \text{MByte/sec}.$$  (6.3)

### 6.1.2 USB3 stability

The stability of the system is tested by running transfers over a longer period, i.e. multiple hours. Performing one transfer every 40 msec (25 fps), the system keeps performing after 8 hours.

The FX3 and the FPGA read-out are stable. However, the temperature of the system might change, causing fluctuations on the TDCs resolution and non-linearities. Those fluctuations don’t cause any problems on the read-out architecture.

### 6.2 QT GUI

For the end-user a user friendly Graphical User Interface (GUI) was designed. The software is completely written in C++. For the GUI the Qt libraries [52] are used. Qt is an open-source UI platform that can be used on different platforms.

The program presents an overview of the different information that can be given on the fluorescence signal. The program is divided into four tabs as shown in Figure 6.1.

On the first tab **Histograms**, the raw or processed histograms are shown. Here the fluorescence signals can be evaluated.

The second tab **Lifetime Data** gives the processed information about all channels. It shows both the measured intensity (either in number of photons or Hz) and the extracted fluorescence lifetime (in ps).

On the third tab **Image** the information that is presented on tab 2 is graphically displayed, it includes no new information.

On the last tab **USB Status** information about the USB connection is given.

The important tabs are thus tab **Histograms** and **Lifetime Data** and the presented information is discussed in more detail below.

<table>
<thead>
<tr>
<th>Histograms</th>
<th>Lifetime Data</th>
<th>Image</th>
<th>USB Status</th>
</tr>
</thead>
</table>

**Figure 6.1:** The GUI is divided with four main tabs: Histograms, Lifetime Data, Image and USB Status.

#### 6.2.1 Histograms

The **Histograms** tab shows the basic information. After the read-out from the FX3, the histograms are plotted in this tab. No extraction of information from this pulse is presented here. However when doing the Wiener filter, the deconvoluted signal can be shown as well.

The basic interface is shown in Figure 6.2(a), a simulated pulse is filling the plot. On the top the tabs can be seen to select the other views and on the right some settings are available.
With checkboxes, the analysis tools can be enabled. Both Wiener filter and Centre of Mass method are available. In the case of doing a Wiener filter analysis, also the deconvoluted signal can be plotted. This plot can be selected under plot types also on the right, resulting in Figure 6.2(c).

For both analysis methods, an IRF has to be present. To view the IRF, a plot mode is available here leading to Figure 6.2(b).

(a) Showing the Histogram in one of the measured channels.

(b) Showing the IRF.

(c) Showing the deconvoluted histogram and the part of the exponential fit in red (Wiener lifetime).

**Figure 6.2:** Overview of the **Histograms** tab; the data shown is derived from simulations as discussed in Chapter 5.
6.2.2 Lifetime Data

On the second tab information of all the channels is presented, both in graph and text mode as shown in Figure 6.3. On the top half of the screen the extracted lifetime and intensity are plotted in blue respectively red. Lifetime is only shown in the case the boxes on the right are checked for one of the data analysis tools.

In the lower half of the screen the same information is given in text mode.

![Figure 6.3: Overview of the Lifetime Data tab, both intensity and lifetime in all channels is plotted; the data shown is derived from simulations as discussed in Chapter 5.](image)

All information presented on tab Histograms and Lifetime Data can be easily saved to make further processing possible.

Plots are made in real-time, i.e. the maximum plotting speed is around 25-30 fps. This includes read-out, analysis and graphical drawing time. The latter is the limiting factor and when disabled, higher speeds are possible.

The speed or integration time can be set. This allows the user to choose whether to perform long and more accurate measurements (more photons are counted) or shorter and thus faster measurements.
Part III

Measurements

Comparing the quality and performance of FluoCAM vs. LinoSPAD

This part will give an overview of the measurements done with both the FluoCAM camera and the newly developed LinoSPAD. A comparison between results is presented.
Chapter 7

Fluorescence Measurement Setup

This chapter provides a short description of the setup to measure both the fluorescence lifetime and intensity. The current setup and experiments are given in depth in [25].

First the targets are described in Section 7.1, a study on ICG in solvents, cells and mice has been done. The setup to measure is shown in Section 7.2 together with the systems components.

For the LinoSPAD setup additional optics are required (Section 7.3).

7.1 Measurement Targets

The different targets to measure fluorescence are shortly described in the overview below.

7.1.1 ICG diluted in solvents

A stock ICG solution in the specified solvent is used to make different dilutions. The stock solution of 200 $\mu$L has a molarity of 200 $\mu$M. The way the dilutions are made are given in the dilution protocol of Table 7.1. A total of seven dilutions were made for both pork blood, milk and water, i.e. the solvent.

The dilutions are kept in a well plate as schematically displayed in Figure 7.1. Each well is measured with the point detection setup.

<table>
<thead>
<tr>
<th>Amount [µL]</th>
<th>Resulting Dilution</th>
<th>Molarity [µM]</th>
<th>Diluted Amount w.r.t. stock</th>
<th>Requisites</th>
<th>Amount pure solvent [µL]</th>
<th>Amount dilution [µL]</th>
<th>Dilution Molarity [µM]</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 (-100)</td>
<td>200</td>
<td>0 ×</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>2 ×</td>
<td></td>
<td>50</td>
<td>50</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>4 ×</td>
<td></td>
<td>75</td>
<td>25</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>100</td>
<td>25</td>
<td>8 ×</td>
<td></td>
<td>87.5</td>
<td>12.5</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>200 (-70)</td>
<td>12.5</td>
<td>16 ×</td>
<td></td>
<td>187.5</td>
<td>12.5</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>100</td>
<td>6.25</td>
<td>32 ×</td>
<td></td>
<td>50</td>
<td>50</td>
<td>12.5</td>
<td>12.5</td>
</tr>
<tr>
<td>100</td>
<td>2.5</td>
<td>80 ×</td>
<td></td>
<td>80</td>
<td>20</td>
<td>12.5</td>
<td>12.5</td>
</tr>
</tbody>
</table>
7.2 FLUORESCENCE SETUP

![Image of dilutions](image.png)

**Figure 7.1:** Dilutions made from the stock solutions with 200 μM in blood, milk and water.

7.1.2 ICG injected in vivo

Pre-clinical in vivo study on mice tumours. The tumours are located on the mice ears as indicated on Figure 7.2. The mice are injected with ICG-RGD as discussed in detail in Section 8.3.

![Image of a mouse with ear tumours](image.png)

**Figure 7.2:** Ear tumours on mouse.

7.2 Fluorescence setup

The setup is shown in Figure 7.3 and consists of the following components:

![Fluorescence setup diagram](image.png)

**Figure 7.3:** Fluorescence setup.
The laser emits light with a wavelength of 790 nm with an average power of 1.5 mW. It is connected to a 100 MHz pulse generator as to have a new light pulse of 200 ps every 10 ns.

The light is reflected on the dichroic mirror and focused on the sample (with a specific ICG concentration). The illumination process is shown in Figure 7.4(a). The distance between lenses can be adjusted to increase or decrease the spot size and focus on the camera.

The sample emits fluorescence light of a higher wavelength and with a lifetime, depending on target and ICG concentration. The fluorescence is emitted in all directions, as in Figure 7.4(b). Only a fraction of the photons reaches the lenses and are eventually registered by the SPADs.

Because of the higher wavelength, the light is able to cross the dichroic mirror and is again focused on the camera. In between is a wavelength selective filter to only allow the sample emitted light to fall on the camera. This eliminates both background noise (from ambient light) and internally reflected laser light.

7.2.1 Point setup

For point setup, one fluorescence sample is measured. Only one fluorescence lifetime is therefore expected. There is no interest in making a 2D image and therefore the spot can be slightly defocused to make it larger on the camera. This makes more pixel signals to be above the noise threshold and thus more statistics for the analysis.

In point setup, the laser spot size is fixed to its original size. The focus on the sample can be adjusted by moving closer or further to the sample.

The efficiency of the camera in point setup mode is calculated for FluoCAM in Section 8.1. From those results, the efficiency for LinoSPAD is estimated in Section 9.3.
In point setup the lifetime is calculated in every pixel, however only one lifetime value over the whole spot is to be found. Therefore the spot lifetime can be found by taking the average over all pixels or by applying a Gaussian distribution.

\[
Gaussian(t) = A \cdot e^{-\left(\frac{t-\mu}{\sqrt{2}\sigma}\right)^2}
\]

(7.1)

The spot lifetime is the \( \mu \) of the Gaussian and the \( \sigma \) indicates the variations over the entire spot. A lower \( \sigma \) indicates a more accurate measurement with less variation.

### 7.2.2 Wide field setup

In wide field mode, the interest lies in making a 2D image. Therefore the system should be in focus to accurately capture the structure details and to accurately find the fluorescence lifetime in the different regions.

Furthermore a larger spot is illuminated by the laser. Therefore an additional lens is used to increase the size of the laser spot on the sample.

### 7.3 LinoSPAD optics

The LinoSPAD chip has a line of SPADs instead of a rectangular camera shape such as FluoCAM. Therefore the optics have to be adjusted to focus a line instead of a circular shape. Furthermore scanning is needed to make a 2D image. The sample area is scanned, focussing each time one line of the image on the camera. Combined results in a 2D image.
7.3.1 Line optics

The laser has to be focused differently w.r.t. FluoCAM. As the original shape of the laser spot is circular, it has to be transformed in a line. The spot shapes for LinoSPAD and FluoCAM are compared in Figure 7.5.

Thereafter the line has to be focused on the LinoSPAD camera. The spot on the camera is a line as well.

(a) Circular spot (FluoCAM). (b) Linear spot (LinoSPAD).

Figure 7.5: Illuminated laser spot.

A second possibility is to illuminate a larger (circular) spot and to focus only a part of the spot on the camera. This makes the illumination optics easier, however the intensity on camera is decreased. A (large) part of the photons spread over the whole area will be lost.

7.3.2 Scanning

To make a 2D image out of a single line, a scanning mechanism is needed.

(a) Rotational. (b) Translational. (c) Optics mirroring.

Figure 7.6: 2D scanning systems: rotating around an axis, translating in one direction or shifting over a spot with mirrors.

It can be done by rotation of the system around a stable axis (Figure 7.6(a)). The system has to allow this kind of rotation, making a wired interface connection difficult.

Second possibility is to displace the system along one direction (Figure 7.6(b)). This requires more complex mechanics, however a wired interface is more feasible.

Another possibility is to fix the camera position and only move the optics (Figure 7.6(c)). Mirrors are required to shift the spot area on the sample and allow the scanning. This option makes the optics more complicated, however the mechanics are easier compared to moving the complete system. Also the camera system is stable, making it easier to connect with cables. Above all there are less moving parts compared to the other solutions.
Chapter 8

Measurements FluoCAM

This chapter will focus on the measurements done with FluoCAM. First an overview of the camera performance is given in Section 8.1. The system is first tested and calibrated with ICG diluted in water and milk, of which the lifetimes are well known. The results are shown in Section 8.2.

Finally a pre-clinical study on mice with FluoCAM was done in Section 8.3.

8.1 Camera performance

First the camera’s performance in terms of noise is briefly described at different temperatures. Afterwards the non linearity performance of the delaylines at different temperatures is given.

8.1.1 Noise (DCR)

A fan is installed on the camera to cool the FPGA and delaylines. While they can heat up till 60°C, the fan ensures a stable temperature of room temperature (25°C).

This benefits both the stability of the system, the delay in the delaylines is more accurate as no drift over temperature is possible. Furthermore the Dark Count Rate (DCR) is suppressed. DCR is caused by thermal electrons generating an avalanche in the SPAD independent of photons arriving. The rate is increased with temperature.

A comparison between the noise on both 40°C and 25°C is given in Figure 8.1 and Table 8.1. The average noise floor drops from 1.3 kHz to 0.4 kHz, an improvement of $3 \times$. Note that the hot pixels are excluded from the noise picture.

<table>
<thead>
<tr>
<th>Noise</th>
<th>40°C</th>
<th>25°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum [kHz]</td>
<td>3.0</td>
<td>1.1</td>
</tr>
<tr>
<td>Average [kHz]</td>
<td>1.3</td>
<td>0.4</td>
</tr>
</tbody>
</table>

8.1.2 Hot & Dead

As can be noted from the noise images in Figure 8.1, there are pixels that have no noise. Those pixels are (partly) disabled in software while they are either dead or hot.

Dead pixels are not influenced by light, i.e. no electrons are generated from the photons and thus no events are exhibited.

Besides pixels that are dead, there are hot pixels. Hot pixels generate continuously electrons, also when there is no photon hit, thus automatically generating events. As the events are counted, the hot pixels have a much higher counts value than would be expected. They clip to the counters limit of 65,535.
8.1.3 Resolution

The delayline used in FluoCAM is characterised by the manufacturer at different temperatures. The data sheet [53] provides the delays in picoseconds for assigning the delay pins. A total of 10 bits or 1023 delays can be assigned. Each bit has a slightly different delay causing the line to have non linearities.

In Figure 8.2 the delays in ps at 25°C and 85°C degrees are extracted from the data sheet, the 45°C is interpolated. The delaylines range is around 12 ns, with an offset of around 2 ns.

The average resolution of the delaylines can be extracted by fitting on the given lines. The fit results for the different temperatures are given in Table 8.2 and are around 10 ps per delay. The difference between 45°C and 25°C is only 0.32 ps (3%). Although the influence of cooling on the delaylines resolution seems small, the difference over the entire range is greater than 300 ps.

Figure 8.1: Noise performance for different temperatures.

Figure 8.2: Delaylines picoseconds delay vs. delay code for different temperatures.
Table 8.2: Delaylines resolution for different temperatures.

<table>
<thead>
<tr>
<th></th>
<th>85°C</th>
<th>45°C</th>
<th>25°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution [ps]</td>
<td>10.35</td>
<td>9.70</td>
<td>9.38</td>
</tr>
<tr>
<td>Offset [ps]</td>
<td>2371</td>
<td>2290</td>
<td>2249</td>
</tr>
</tbody>
</table>

8.1.4 Non linearities

The non linearities are presented in Figure 8.3 for different temperatures. They can be seen to be most prominent on the larger delay bits. A transition ‘01111’ → ‘10000’ has a more deviant step compared to ‘01101’ → ‘01110’.

According to Figure 8.3 the non linearities are smaller in the case the delaylines are at 85°C compared to 25°C. The difference in the peak to peak DNL is around 5 LSB or 50 ps as shown in Table 8.3. Thus the influence of the cooling on the non linearities is negative, increasing the DNL and INL.

However cooling is necessary to decrease noise in the camera, especially in the case of low intensity signals. Therefore care has to be taken to correct for the non linearities differently at each temperature and calibrate for this. This is done by removing those parts of the signal where the delay is negative and recovers to the already reached delay value.

![DNL and INL plots](image)

Figure 8.3: Linearity performance for different temperatures.

Table 8.3: Linearity performance for different temperatures.

<table>
<thead>
<tr>
<th>Temperature</th>
<th>$DNL_{pk-pk}$ [LSB]</th>
<th>$INL_{pk-pk}$ [LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>25°C</td>
<td>15.9</td>
<td>17.7</td>
</tr>
<tr>
<td>45°C</td>
<td>13.5</td>
<td>15.9</td>
</tr>
<tr>
<td>85°C</td>
<td>10.6</td>
<td>12.9</td>
</tr>
</tbody>
</table>

8.1.5 Efficiency

The efficiency of the whole setup is $\ll 1\%$, i.e. only a very small number of photons that are sent by the laser are eventually seen on the camera.

The integrated energy of the laser is around 1.5 mW, with a wavelength $\lambda = 790$ nm. The energy per photon can be calculated with Equation 1.1,

$$E = \frac{h \cdot c}{\lambda} = \frac{6.626 \cdot 10^{-34} \cdot 2.998 \cdot 10^8}{790 \cdot 10^{-9}} = 2.52 \cdot 10^{-19} \text{ J/photon}.$$  

(8.1)
Expressing the energy sent by the laser per second gives the irradiation in Joule/s, which can be translated to the incoming photon flux in Hz:

\[ P = 1.5 \text{ mW} = 1.5 \text{ mJ/s} \]  \hspace{1cm} (8.2a)

\[ \Phi = \frac{1.5 \cdot 10^{-3} \text{ J/s}}{2.52 \cdot 10^{-19} \text{ J/photon}} \approx 6 \cdot 10^{15} \text{ Hz} \]  \hspace{1cm} (8.2b)

The intensities of all measured samples are shown in Figure 8.11. The total number of photons inside the spot area are summed. Taking the brightest sample, the best efficiency can be estimated. The brightest sample is ICG in blood with a concentration of 100 µM.

The measured intensity can be read from Figure 8.11, it is around \(2.8 \cdot 10^7\) Hz, summed over the total spot seen on camera.

The quantum efficiency of the setup is therefore

\[ \frac{2.8 \cdot 10^7}{6 \cdot 10^{15}} \cdot 100\% = 4.7 \cdot 10^{-7}\%. \]  \hspace{1cm} (8.3)

The lowest efficiency of the measured samples is reached with ICG in water with a concentration of 100 µM. The total number of photons is \(5.1 \cdot 10^5\) Hz. The quantum efficiency in this case is

\[ \frac{5.1 \cdot 10^5}{6 \cdot 10^{15}} \cdot 100\% = 7 \cdot 10^{-9}\%. \]  \hspace{1cm} (8.4)

Main limitations are the yield of the ICG fluorescence itself, \(\ll 1\%\) \[54\], the fact that light is emitted in all directions instead of only towards the camera. Furthermore the fill factor of the pixels is with 1% another large yield loss. Last the quantum efficiency of silicon in the near infrared is around 2% \[55\].

Similar efficiencies are reached with other systems. With the LinoSPAD system, the fill factor is much higher, leading to an increase in efficiency.

### 8.2 Lifetime calibration & Comparison with literature

To test and calibrate the system, some solvents with a known lifetime were first investigated. This includes ICG in water and milk. Afterwards a sample that is more likely in a clinical environment, blood, was studied.

All experiments were done with the cooled FluoCAM system at room temperature (25°C) in point setup.

#### 8.2.1 Water

ICG in water has a steady lifetime over all concentrations, therefore doing this measurement is a good reference for the goodness of the analysis and the outcome of the other measurements. This is mainly to be sure of the delay in the delaylines, which can have variation that directly influences the lifetime.

From literature the lifetime in water is found to be 150 ps \[56\], 166 ps \[22\], 170 ps \[57\], 190 ps \[58\], 200 ps \[17\]. On average a result round 170 ps for every molar concentration is expected.

The brightest sample from this measurements is ICG in water with a concentration of 50 µM. This sample yields a peak value of \(6.3 \cdot 10^4\) Hz. Which is almost a decade below the highest intensity observed for ICG in milk. The total number of photons in the spot area is \(2.3 \cdot 10^6\) Hz, again a decade below milk.

The intensity of the sample is plotted in Figure 8.4 (top view in Figure 8.5(a)). On each pixel the Wiener filter is applied and an exponential fit is made on the deconvoluted signal. From this the lifetimes are extracted and plotted in Figure 8.5(b). A threshold is set to only estimate the lifetime in the pixels that are (far) above the noise level. This gives more stable results with a smaller deviation.
Measurements FluoCAM

Figure 8.4: Intensity plot of ICG 50 µM in water.

Figure 8.5: Intensity and Lifetime plots for ICG in water at 50 µM.

The spot lifetime $\mu = 169$ ps. The $\sigma = 25$ ps, which implies the deviation is small. This result is in good agreement with the literature value of 170 ps.

All extracted lifetimes for the different molarities of ICG in water can be found in Figure 8.10. The other lifetimes are also in the order of 170 ps, showing the system is able to accurately extract lifetimes.

For ICG in milk with the lower concentrations $< 25$ µM, the intensity was too low to accurately measure the fluorescence lifetime. The signals become very noisy.

8.2.2 Milk

The sample of ICG in milk, 6.25 µM is the brightest, is shown in Figure 8.6. The same procedure as for water is repeated. The lifetime and intensity of the example are plotted in Figure 8.7.

Extracted $\mu = 679$ ps and $\sigma = 40$ ps are found. The extracted $\mu$ is somewhat below the expected value from literature, which is in the order of 720 ps. However this fits inside the error range of $\sigma = 40$ ps.

All results can again be found in Figure 8.10. The overall trend from the milk results is in good agreement to the results found in literature. The $\sigma$’s found for ICG in milk are larger than for ICG in water. ICG in
water is more stable and the deviations found smaller.

Figure 8.6: Intensity plot of ICG 6.25 µM in milk.

Figure 8.7: Intensity and Lifetime plots for ICG in milk at 6.25 µM.
8.2.3 Blood

The brightest blood sample had a concentration of 100 $\mu$M. Plotted in both Figure 8.8 and Figure 8.9(a). The lifetimes are extracted in Figure 8.9(b).

The results: $\mu = 561$ ps and $\sigma = 30$ ps. The deviation is comparable to milk, higher than for ICG in water.

![Figure 8.8: Intensity plot of ICG 100 $\mu$M in blood.](image1)

![Figure 8.9: Intensity and Lifetime plots for ICG in blood at 100 $\mu$M.](image2)

The same repeated for blood gives the results again in Figure 8.10. The extracted lifetimes are in the same range as ICG in milk. However the lifetimes are more stable. Especially at concentrations below 50 $\mu$M, similar lifetimes are found $\approx 600$ ps. There is in contrast to ICG in both water and milk no comparison with literature.
8.3 In vivo

A pre-clinical study with FluoCAM has been made on mice at the EPFL animal facility. The mice are imaged with the IVIS machine (presented in Subsection 1.5.3) and FluoCAM. Although the IVIS cannot measure the fluorescence lifetime, it can show where the fluorescence intensity is high. The two machines in the facility are shown in Figure 8.12(a) and (b).

The IVIS machine has built in mouse anaesthesia, however for the FluoCAM a mobile anaesthesia station has to be used, shown in Figure 8.12(c).

Figure 8.10: Extracted lifetimes for ICG in water, blood and milk for different molarities. Literature values from [22] are given as reference.

Figure 8.11: Extracted intensities summed in the spot area for ICG in water, blood and milk for different molarities.
8.3.1 ICG-RGD

The mice were injected in the tail vein with ICG-E[c(RGDfK)₄]. ICG is conjugated with four arms of cyclic pentapeptide (RGD), which makes the ICG more specific and with a higher activity against integrins αᵥβ₃ and αᵥβ₅ [59]. ‘Integrins are the main receptor proteins that cells use to both bind to and respond to the extracellular matrix.’ [60]. So the ICG can better connect to the cells with the αᵥβ₃ integrins with the RGD arms. The αᵥβ₃ integrins are specifically active on tumour cells, leading to a large part of the ICG-RGD to connect to those cells.

8.3.2 IVIS & FluoCAM

The fluorescence from the tumour in the mouse ear was measured, after injection, in periods of one to multiple days. This was first measured with the IVIS machine and afterwards under the FluoCAM. The injection is done while the mice are anaesthetized as shown in Figure 8.13(a).

FluoCAM is used with point setup to measure the fluorescence from the mouse tumour; only a small laser spot is generated on the mouse ear. The spot and the mouse under the FluoCAM setup are shown in Figure 8.13(b) and (c).

24 hours after injection, the ICG is well spread throughout the mouse body and the mouse is measured with IVIS. The IVIS measures the fluorescence at different wavelengths using filters in front of a light source. From this the optimal wavelength for the fluorescence emission can be found. The IVIS image is presented in Figure 8.14 and takes around 30 seconds to measure per wavelength.

The tumours show very bright fluorescence emission compared to the background. A quantitative analysis shows a fluorescence signal from the Region of Interest 1 (ROI 1: the tumour region) 19 times higher than for the background signal from ROI 2. This demonstrates the excellent specificity of the ICG-E[c(RGDfK)₄] on the tumour targeted.

The IVIS machine does not produce lifetime information, therefore there is no comparison whether FluoCAM provides correct lifetime estimates.

With FluoCAM a measurement was done on the mouse ear, where the tumour is located. The measurement takes around 8 minutes; 1 second integration time per frame with 500 frames.

This relatively long time, compared to the IVIS, introduces some limitations. First on the number of measurements that can be done, as the mouse cannot stay anaesthetized for too long.

Secondly on the mouse movements. As the mouse breathes, it slightly moves up and down, changing the focus. Above all the mouse can shift too much, causing the laser spot to drift away from the tumour.

Finally photobleaching is observed. The intensity on the tumour is seen to decrease to 50% of its initial value after 8 minutes.
As LinoSPAD requires measurement times in the order of tens to hundreds of milliseconds, those limitations should be lifted.

The fluorescence intensity on the mouse tumour measured with FluoCAM is shown in Figure 8.15. The intensity in the spot area is around $3 \cdot 10^6$ Hz. A clear fluorescence signal was observed when the laser spot was located on the tumour. This signal disappeared when any other region of the mouse was targeted.
The lifetimes are extracted and plotted in Figure 8.16. The lifetimes show a larger spread compared to the measurements made in the solvents. The calculated $\sigma$ of the Gaussian fit is around 100 ps. This is mainly due to the distortions in the signals as for the reasons mentioned above.

The mean value $\mu$ of the extracted lifetimes is in the order of 250-270 ps. Repetitive measurements over several days confirm this finding.

Those preliminary results show that fluorescence lifetime imaging can be done with FluoCAM on mouse. Although the system has limitations in signal stability over the measurement time, findings are conformed over several measurement sessions.

Extensive research has to be done to confirm lifetimes and compare with non-tumour tissue lifetimes.
Chapter 9

Measurements & Fluorescence emulator
LinoSPAD

Note from the author: As the LinoSPAD chip is not behaving as expected, no photosensitivity, I can unfortunately not present real measurement results here. Instead an emulator was designed and some experiments with a single SPAD are done.

As the LinoSPAD chip is not finished, the system has to be tested using an alternative approach. Therefore an emulator was designed, that can mimic the fluorescence behaviour. The design of this emulator and its results are shown in Section 9.1.

To show the system is able to receive SPAD pulses and measure its timings, a single SPAD was connected. From this SPAD the noise (DCR) can be found together with the systems resolution and non linearities. The measurements are described in Section 9.2.

As no fluorescence measurement data is available, the final performance of the LinoSPAD system can only be estimated. Estimations of its performance in comparison with FluoCAM are therefore drawn in Section 9.3.

9.1 Emulator

An emulator was designed in a second FPGA in order to simulate the behaviour of the LinoSPAD chip, i.e. to test the LinoSPAD FPGA routines. Therefore the Spartan 6 SP605 Evaluation board was connected with its FMC connector to the LinoSPAD board. The SP605 contains a similar FPGA as on the LinoSPAD board, only smaller. The board can contain the emulator and has the advantage of having switches and buttons to change operating mode; switch between different signals.

First the working principle and the idea is sketched in Subsection 9.1.1, after which the implementation is discussed in Subsection 9.1.2. Finally a discussion on the setup and results is given in Subsection 9.1.3 respectively Subsection 9.1.4.

9.1.1 Principle

For building a fluorescence emulation, an exponential has to be generated. The pulses have to arrive in time according to Figure 9.1(a). For the timings to have an exponential some mapping has to be made.

In this emulator the clock signal is delayed with a carrychain delayline, as earlier shown for the TDCs in Chapter 3. The delayed signals are schematically drawn in Figure 9.2. The likelihood of occurrence of the shortly delayed clock pulses has to be higher than of the longer delayed pulses as to emulate the signals arriving exponentially. The exponential probability is drawn as reference.

The mapping is done on a random number generator as shown in Figure 9.1(b). Taking a Linear Feedback Shift Register which can generate 1023 pseudo random numbers, parts of the numbers are ‘mapped’ on the exponential.
9.1. EMULATOR

(a) The fluorescence exponential signal. (b) LFSR random number generator.

Figure 9.1: The fluorescence exponential signal is mapped onto a LFSR random number generator, area of exponential corresponds to a certain number of random numbers.

E.g. the first bin of the exponential (the bins are the resolution of the delayline, i.e. around 21 ps) has the largest area. This area represents the probability of the hit Δ1 to occur in this part of the exponential curve. The area is represented by a part of the randomly generated numbers, e.g. the first 100. The second bin is smaller in area and thus is mapped to less random numbers, e.g. 80 etc.

The random number generated each clock period thus decides which Δx is connected to the output. The area or the range of random numbers in each bin is chosen with respect to the fluorescence lifetime. Higher lifetime gives a wider exponential and thus less area in the beginning and more in the end.

Figure 9.2: Delayed clock signals and probability of occurrence with exponential behaviour.

9.1.2 Implementation

For the implementation of this emulator, two main components are used, the carrychain delayline and a LFSR with the mapping properties.

Delayline

The delayline is similar to the delayline shown in Subsection 3.1.2. The carrychain is again used, however instead of flip-flops, the raw output signals of the delayed input clock are selected. The delayed signals are Δx.

Linear Feedback Shift Register with mapping

The generation of the exponential probability is done with a LFSR. A 9 bit LFSR can generate 1023 pseudo random numbers with linear feedback. The numbers are pseudo random, as the order is the same
after every reset. The design is shown in Figure 9.3. The registers are arranged as a shift register, the value is shifted towards the next register each clock cycle. For the generation of the numbers, the first register is connected as

\[ \text{IN}(0) = \text{rand}(6) \text{ XNOR } \text{rand}(9). \]

In Table 9.1 a part of the sequence that is generated with the LFSR is shown together with the mapped output \( \Delta x \). Even in this small part the likelihood of the lower \( \Delta x \) to be selected is higher.

<table>
<thead>
<tr>
<th>Generated number</th>
<th>16</th>
<th>33</th>
<th>67</th>
<th>134</th>
<th>269</th>
<th>539</th>
<th>54</th>
<th>109</th>
<th>218</th>
<th>436</th>
<th>873</th>
<th>723</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output ( \Delta x )</td>
<td>( \Delta 1 )</td>
<td>( \Delta 1 )</td>
<td>( \Delta 2 )</td>
<td>( \Delta 3 )</td>
<td>( \Delta 7 )</td>
<td>( \Delta 19 )</td>
<td>( \Delta 2 )</td>
<td>( \Delta 3 )</td>
<td>( \Delta 6 )</td>
<td>( \Delta 14 )</td>
<td>( \Delta 50 )</td>
<td>( \Delta 31 )</td>
</tr>
</tbody>
</table>

Dependent on the outcome of the LFSR one of the \( \Delta x \) outputs of the delayline is connected to the output. Every clock cycle a new number is generated and thus a new \( \Delta x \) is selected.

**Output generation**

With a second, but smaller LFSR the generated pulses are randomly connected to the FMC outputs. The FMC has 32 differential outputs (64 pins), to each one random LFSR number is connected.

In the case the LFSR generates the number 1, the output pulse is connected to FMC1, for 2 FMC 2 etc. The complete emulator structure is drawn in Figure 9.4.
9.1. EMULATOR

Limitations

The main concern of this emulator is the delayline. The delaylines carrychain structure will cause the $\Delta t$s to be non consistent. Some delays are smaller and others larger, i.e. the non linearities. Secondly delays are swapped due to the carry look ahead.

Furthermore as the raw outputs of the carrychain are used, they are all differently routed towards the output causing again different delays. Different routing paths can cause substantial differences. The FPGA is clearly not intended for this asynchronous system. This will cause crude exponentials in the end, however the general shape might be observed.

9.1.3 Setup

The emulator is implemented in the SP605 FPGA as described above, the emulator is connected through the FMC connector with the LinoSPAD board. On the LinoSPAD FPGA the TCSPC system is implemented. The 256 channels are divided in 8 TDC banks, each having 32 channels. Therefore the 32 differential FMC channels are directly connected with the 8 banks.

The test setup with the emulator is schematically displayed in Figure 9.5. On the SP605 a switch is implemented to toggle between different emulation modes. This includes a delayed clock pulse, a ringoscillator signal for random timing generation and the exponentials with different $\tau$'s.

The test system is finally shown in Figure 9.6. The LinoSPAD mainboard can be seen to be directly connected on top of the SP605 emulator.

Figure 9.5: Test setup with the SP605 emulator.

Figure 9.6: LinoSPAD test setup with fluorescence emulator and OneSPAD.
9.1.4 Results

The emulator was tested using the LinoSPAD FPGA system. The different signals show semi-exponential behaviour, however due to the increased jitter of two delaylines, the signals look a little wider w.r.t. the reference exponentials. Also non linearities of both delaylines combined decrease the signals quality. Three measurement results are plotted in Figure 9.7.

Figure 9.7: Measured exponential signals from the SP605 emulator, the correct exponentials are drawn as reference.

Figure 9.7(a) shows an exponential with a lifetime of 100 ps. The jitter’s $\sigma$ is around 100 ps, which is around the lifetime. Therefore the measured signal is more like a convolution of a pure exponential with a pulse with a width of about 250 ps. This pulse can be seen as an IRF.

Figure 9.7(b) and (c) show exponentials with lifetimes of 750 respectively 1000 ps. Both fit better in the reference exponential curve. However the exponentials themselves look distorted. Especially for 1000 ps, there are dips / gaps in the exponential.
9.2 OneSPAD

Instead of a line of SPADs (LinoSPAD), a single SPAD from the chip described in [61] is connected to the system (OneSPAD). This makes verification of at least the systems resolution and non-linearities possible. Furthermore, it shows the system is able to handle SPAD pulses and can accurately find DCR.

9.2.1 Setup

The buffered active quenched SPAD of Figure 1.7(c) has been tested. A passive quenched SPAD is found to be too slow with a rise time of over 100 ns due to parasitics. The active quenched SPAD on the other hand is first connected to a buffer. This buffer makes the rise time of the SPAD much shorter. The active quenched SPAD is described in [61], it is a SPAD with a 6 μm diameter and a breakdown voltage of 19.2 V.

Whereas the rise time of the passive SPAD is in the order of 200 ns, the rise time of the active system is only around 3 ns. With a shorter rise time, the accuracy of the hit registration is increased, as the flip flop to which the SPAD is connected sees a much sharper rising edge.

The SPAD dead time is around 1 μsec. This is a trade-off between dead time and after pulsing. Lower dead time increases the number of after pulses.

The test setup with the SPAD chip was already shown in Figure 9.6. The SPAD chip needs different voltages, coming from the power supplies and a control board.

9.2.2 SPAD pulse verification

The pulses from the SPAD need to be verified to be purely random over time. This verification is done with a LeCroy WavePro oscilloscope, which has built-in TDCs to make a similar histogram as the LinoSPAD FPGA.

First the SPAD pulses are confirmed. In Figure 9.8(a) the active quenched SPAD pulse after the buffer is shown together with the passive quenched pulse without buffer. The rise time of the buffered pulse can be seen to be in the order of 3.5 ns, comparing with the 96 MHz system clock. The systems clock period is around 10.5 ns.

The SPAD pulses over time are shown in Figure 9.8(b). The pulses in a 100 msec time window are plotted. The pulses are indeed random w.r.t. time, there is no frequency of occurrence visible.

Integrating over several minutes, the histogram can be built from the SPAD pulses. The LeCroy TDC measures the difference between the rising edge of the SPAD pulses and the systems clock.
LinoSPAD board. This histogram is drawn in Figure 9.9 and should be similar to the LinoSPAD histogram measurements. It is a flat (white) noise histogram as expected.

![Figure 9.9: DCR histogram as measured from the SPAD pulses with the LeCroy oscilloscope.](image)

9.2.3 LinoSPAD system with OneSPAD

The same SPAD pulses were measured with the LinoSPAD FPGA system. A long integration time (1 hour) in the dark is needed as the SPAD exhibits low DCR in the order of < 100 Hz.

DCR measurements with the LinoSPAD system find 80-120 Hz. This can be caused by the system not being perfectly in the dark. The noise histogram from (one of the channels of) the LinoSPAD FPGA system is shown in Figure 9.10. It is in comparison with Figure 9.9 less flat due to the non linearities of the FPGA TDCs.

Non linearities

As done with the ringoscillator setup in Section 4.3, the systems non linearities can as well be estimated from Figure 9.10. INL and DNL are still defined as Equation 4.6 respectively Equation 4.8. The DNL and INL are extracted from Figure 9.10 and plotted in Figure 9.11.

The peak to peak DNL and INL as defined in Equation 4.7 are again found for the 8 TDCs and shown together with the simulation and average ringoscillator results in Table 9.2.

Comparing the results shows similar results as with the ringoscillator. This confirms the ringoscillator findings and shows OneSPAD has indeed purely random DCR behaviour.

Resolution

The TDC resolution can as well be found from Figure 9.10. The resolution from the noise histogram is:

\[
\text{Resolution} = \frac{1}{(\text{max}_\text{bin} - \text{min}_\text{bin}) \cdot f_{\text{CLK}}} \tag{9.1}
\]
9.2. ONESPAD

Figure 9.10: DCR histogram as measured from the SPAD pulses with the LinoSPAD FPGA system.

Figure 9.11: DNL & INL for OneSPAD generated histogram of Figure 9.10.

in which \( \text{max}_{\text{bin}} \) and \( \text{min}_{\text{bin}} \) are respectively the maximum histogram bin with a value and the minimum bin with a value.

The resolution is found to be

\[
\text{Resolution} = \frac{1}{(510 - 20) \cdot 96 MHz} = 21.3 \text{ ps}
\]

which is the same as the resolution found by simulation (21.3 ps), but higher with calibration against FluoCAM (20.4 ps). The FluoCAMs resolution is not so accurate, causing this deviation.

Both resolution and non linearities are temperature dependent. As a long integration time was needed to
Table 9.2: Comparison of DNL & INL between simulation, average result from ringoscillator experiment and OneSPAD; implementation with 8 TDCs.

<table>
<thead>
<tr>
<th>Implemented TDC</th>
<th>$DNL_{pk-pk}$ [LSB]</th>
<th>$INL_{pk-pk}$ [LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>6.2</td>
<td>8.2</td>
</tr>
<tr>
<td>Ringoscillator</td>
<td>3.9</td>
<td>7.4</td>
</tr>
<tr>
<td>OneSPAD</td>
<td>4.1</td>
<td>7.4</td>
</tr>
</tbody>
</table>

obtain sufficient histogram data, the temperature of the FPGA was changed. Therefore some difference between those and previous findings can be accounted for.

The non linearities are increased and the resolution decreased for increasing FPGA temperature. A temperature calibration of the full system should be taken into account to achieve best results.

9.3 LinoSPAD expectations

Although the chip is not available, some expectations w.r.t. the expected photon flux and necessary integration times can be made.

From Chapter 8 the photon flux on FluoCAM was found to be in the range $5 \cdot 10^5$ Hz - $3 \cdot 10^7$ Hz. With adjusted optics, the spot projected on FluoCAM can be projected onto the line of LinoSPAD as shown in Figure 9.12. As the fill factor on LinoSPAD is $44 \times$ higher compared to FluoCAM a theoretical improvement in photon flux of almost $44 \times$ could be achieved.

However as schematically drawn in Figure 9.12(b) it is difficult to focus such a long, but narrow line. Therefore a part of the incident photons is lost. The improvement compared to FluoCAM is estimated to be in the order of $20 \times$.

![Fluorescence spot on FluoCAM.](a)
![Fluorescence spot on LinoSPAD.](b)

Figure 9.12: Optical projection of the fluorescence spot on the FluoCAM and LinoSPAD SPAD arrays.
A study of the upper and lower intensity cases is made in Table 9.3. The intensity on LinoSPAD is 20 ×
higher compared with FluoCAM and focuses on less pixels. However the DCR is increased as well as
the active area is roughly 4 × higher than FluoCAM. Still, the SNR is increased making faster imaging
possible.

<table>
<thead>
<tr>
<th>Detection system</th>
<th>FluoCAM</th>
<th>LinoSPAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spot on camera</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pixels in spot (focus)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measured photon flux in spot [Hz]</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>$5 \cdot 10^5$</td>
<td>$3 \cdot 10^7$</td>
</tr>
<tr>
<td>Average photons / pixel [Hz]</td>
<td>$1 \cdot 10^3$</td>
<td>$5 \cdot 10^4$</td>
</tr>
<tr>
<td>DCR (cooled) [Hz]</td>
<td>$1 \cdot 10^3$</td>
<td></td>
</tr>
<tr>
<td>SNR [dB]</td>
<td>0</td>
<td>34</td>
</tr>
</tbody>
</table>

In Chapter 5 the minimum amount of photons needed for accurate lifetime estimation was found to be
between 2500 and 5000. In this subsection an amount of 3000 photons is assumed to be satisfying.

With the required 3000 photons per measurement and the photon fluxes given in Table 9.3 for
LinoSPAD the integration time $T$ is found.

$$T_{\text{fast}} = \frac{3000}{2 \cdot 10^6} \approx 2 \text{ ms}$$  \hspace{1cm} (9.2a)

$$T_{\text{slow}} = \frac{3000}{4 \cdot 10^4} \approx 75 \text{ ms}$$  \hspace{1cm} (9.2b)

With $T_{\text{fast}}$ are realtime system can be made with video speed (25 Hz, 40 ms / frame).

The integration time for a 2D image of $256 \times 256$ pixels will be a 256 multiple of $T$.

$$T_{\text{2D, fast}} = 256 \cdot 2 \text{ ms} \approx 0.5 \text{ s}$$  \hspace{1cm} (9.3a)

$$T_{\text{2D, slow}} = 256 \cdot 75 \text{ ms} \approx 20 \text{ s}$$  \hspace{1cm} (9.3b)

This shows that 2D imaging with video frame rate 25 fps = 40 msec integration time cannot be achieved
with the current configuration. However acquiring 2D lifetime images in less than a second is an im-
proved compared to FluoCAM of 500 ×.
Chapter 10

Conclusion

This final chapter gives the conclusions on the work that has been performed in EPFL and at TU Delft. The prototype is evaluated in Section 10.1, the requirements given in Appendix A are used as evaluation basis. Furthermore the achieved results are briefly summarized.

Finally, in Section 10.2 suggestions for future work are given and explained. The future work is an important part as a prototype has been developed and the final system has to be designed and tested.

10.1 Evaluation of the LinoSPAD prototype

The results that were shown in the previous chapters are compared with the requirements (listed in Appendix A). The main requirements for the LinoSPAD system are listed in Table 10.1. Not only an overview of the firmware and software is given, but also the status of the hardware.

The requirements that were achieved are marked ✓. The majority of the requirements have been achieved, however testing with the LinoSPAD chip could not take place as the chip is not functional. A new chip is not expected before August.

Firmware

A complete solution to analyse fluorescence signals was developed. The system consists of a mainboard with a Xilinx Spartan 6 FPGA and a Cypress FX3 controller and a daughterboard with the LinoSPAD chip with 256 SPADs. The SPAD outputs are directly connected to the FPGA. With the raw SPAD pulses, the FPGA can measure with the Time Correlated Single Photon Counting Technique the fluorescence signals. Finally software was designed to analyse the fluorescence signals.

For the firmware a FPGA TCSPC system was implemented inside the Spartan 6. The system is capable of handling 256 channels, which are divided in 8 TDC banks. The 8 TDCs have a resolution (LSB) of 20-21 ps. This has been confirmed in simulation and hardware testing with both a density test using SPAD noise and with a calibration against a 10 ps delayline.

The non linearities of the TDCs were found with density tests using a ringoscillator and SPAD noise. Both tests showed similar results. $DNL_{pk-pk} \approx 4 \text{ LSB}, \ INL_{pk-pk} \approx 7.5 \text{ LSB}.$

The TDCs are free running with the clock (100 MHz), giving one time conversion per clock cycle. To complete the TCSPC system, histograms are generated on the FPGA. A RAM memory was implemented in which the timings from the TDCs are stored. Each of the 256 channels has an own memory and therefore an own histogram which is read-out using the FX3 USB3 controller.

As the system is completely implemented inside a FPGA, it can be adapted to the needs of the application at any time. E.g. applications that do not require the histograms can eliminate this part of the design and implement different read-out.
### Table 10.1: The various system requirements that are achieved (√) in the LinoSPAD prototype.

<table>
<thead>
<tr>
<th>System Requirements</th>
<th>Prototype</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCSPC</td>
<td>✓</td>
<td>The system is capable of building the TCSPC histogram on FPGA</td>
</tr>
<tr>
<td>Multichannel system implementation</td>
<td>✓</td>
<td>Up to 256 channels and 8 TDCs in one FPGA</td>
</tr>
<tr>
<td>Up to 100 million-8 time conversions per second</td>
<td>✓</td>
<td>Each TDC runs freely at 100 MHz, doing one conversion each CLK cycle</td>
</tr>
<tr>
<td>Histogram read-out with FX3</td>
<td>✓</td>
<td>Fast USB3 read-out has been implemented to read-out the histograms</td>
</tr>
<tr>
<td>Raw read-out with FX3</td>
<td>✓</td>
<td>The basics for raw read-out (SPAD + time code) have been laid out, no software implementation</td>
</tr>
<tr>
<td>Expandable</td>
<td>✓</td>
<td>The FPGA can be reprogrammed at any time for easy prototyping</td>
</tr>
<tr>
<td>Processing and display (1D) &gt; 25 fps</td>
<td>✓</td>
<td>Handling 256 analyses and read-out in less than 30 ms</td>
</tr>
<tr>
<td>Processing and display (2D) &gt; 25 fps</td>
<td></td>
<td>Handling 256 × 256 analyses and read-outs in less than 30 ms cannot be reached</td>
</tr>
<tr>
<td>Lifetime extraction</td>
<td>✓</td>
<td>Both Wiener filter and Centre of Mass implemented for analysis</td>
</tr>
<tr>
<td>Realtime plotting of graphs</td>
<td>✓</td>
<td>Multiple graphs can be rendered with video speed</td>
</tr>
<tr>
<td>LinoSPAD mainboard</td>
<td>✓</td>
<td>The LinoSPAD mainboard is fully tested and functional</td>
</tr>
<tr>
<td>LinoSPAD chip</td>
<td></td>
<td>The chip is not functional</td>
</tr>
<tr>
<td>LinoSPAD daughterboard</td>
<td>✓</td>
<td>The daughterboards are ready for bonding</td>
</tr>
<tr>
<td>Optics</td>
<td></td>
<td>Proposed for line optics focusing</td>
</tr>
<tr>
<td>Laser</td>
<td>✓</td>
<td>A pulsed laser for the prototype is used, ( \lambda = 790 \text{ nm}, P = 1.5 \text{ mW}, \text{Pulse width} = 200 \text{ ps} )</td>
</tr>
<tr>
<td>Data analysis (simulation)</td>
<td>✓</td>
<td>The analysis tools have been extensively tested</td>
</tr>
<tr>
<td>Accuracy up to 30 ps (simulation)</td>
<td>✓</td>
<td>The lifetimes can be extracted with accuracies up to 30 ps</td>
</tr>
<tr>
<td>Data analysis (practice)</td>
<td></td>
<td>The analysis tools are not tested as LinoSPAD is not functional</td>
</tr>
<tr>
<td>Read-out speed around 2 ms</td>
<td>✓</td>
<td>Read-out of 256 histograms</td>
</tr>
<tr>
<td>Realtime 2D</td>
<td></td>
<td>Realtime 2D processing to be tested</td>
</tr>
<tr>
<td>Well documented</td>
<td>✓</td>
<td>All parts of the design documented in this thesis</td>
</tr>
<tr>
<td>Suggestions for future work</td>
<td>✓</td>
<td>List included for further development</td>
</tr>
</tbody>
</table>

### Software

The histograms are read-out over USB3 in less than 2 ms. The designed GUI can analyse the signals to extract the lifetime in two different ways. Both the Wiener filter and Centre of Mass methods were implemented in C++.

In simulations, the accuracy of both analysis algorithms was tested. In the worst case, in which both noise and non-linearities have been added in the simulation, they perform similar with an average lifetime error of 30 ps over a range from 100 ps to 1 ns. The amount of photons used in this simulation is 2500. The speed in contrary is significantly different. The Centre of Mass outperforms the Wiener filter.
Analyses are up to 100× faster with CoM (dependent on the systems hardware). The time to analyse one LinoSPAD read-out (256 channels) with CoM is 0.1 ms.

The total time to process one measurement is thus < 2.5 ms. Realtime processing and display is possible, in which the main limitation is the speed in which the graphs are drawn.

A rectangle read-out and analyse of 256 lines requires at least 640 ms, in which the integration time is not even taken into account. Realtime processing of 2D images with video speed (25 fps, 40 ms) can therefore not be achieved with the current system.

**Testing**

As the LinoSPAD chip is not available, no extensive measurements were done with the LinoSPAD system w.r.t. analysing fluorescence. Reference measurements have been done with FluoCAM.

The FluoCAM system was calibrated with different ICG samples that have literature correspondence. The extracted lifetimes found with the FluoCAM are in good agreement with the results expected from literature.

Furthermore, first in vivo measurements were done using FluoCAM. ICG-RGD in mouse was studied and revealed some limitations of the FluoCAM system, such as mouse movement in the long measurement time. Photobleaching from the ICG was observed limiting the signals quality. However lifetimes were extracted for the in vivo samples and showed little difference over days.

With the results from FluoCAM expectations were made for LinoSPAD. The required integration times were estimated. They are strongly dependent on the intensity of the measured ICG sample. Integration times are in the range from some milliseconds to several tens of milliseconds.

For 2D imaging scanning has to be applied to the LinoSPAD system, the required integration times for a square image are in the range of some to several seconds. This is above the realtime requirement, however a huge improvement compared to the 8 minutes measurement time required on FluoCAM. Although the analysis and read-out may not be able to process in realtime 2D, the measurement times certainly will limit 2D scanning imaging.

In order to test some exponential signals, a fluorescence emulator was designed. Random exponential timings were generated inside a second FPGA and measured with the LinoSPAD system. Although the LinoSPAD histograms showed some exponential behaviour the quality of the signals is decreased by the non linearities of the system. The non linearities are increased as the timings are generated in a delayline and measured with a similar structured TDC, this doubles the non linearities of the total system compared to the LinoSPAD system. Furthermore the delayed signals in the emulator have different routing paths, increasing the timing differences towards the output. The analysis tools could therefore not be tested with the fluorescence exponentials from the emulator.

In order to test with a SPAD (LinoSPAD not being available) a single SPAD was connected to the LinoSPAD FPGA system. With this setup findings w.r.t. the systems non linearities and resolution were confirmed. Furthermore the SPADs DCR was extracted which was inside the expected range.

In this thesis the complete path of the system development was shown. First a detailed description of the firmware and software was given, after which reference measurement results and the measurements system were introduced.

Although the system didn’t achieve all requirements, main contributions to a new fluorescence lifetime camera have been made. Some points for further improvements are listed in the following section.
10.2 Further Work

During the development of the prototype and its testing, points of interest for further work have been gathered. The most important points are listed below.

I. **Testing the system with a working LinoSPAD chip.** The most important step is to test the system with a working chip, investigate the SPAD signals and calibrate the system for the delays in between SPADs and the TDCs.

II. **Testing the line optics and implement 2D scanning functionality.** The software should be adapted to store the lines of a 2D frame and bundle it in one picture.

III. **Improve the algorithms and read-out for higher performance.** The current system is limited to 30 fps, however the final system should scan an area in the time that currently only one line is measured. Therefore improvements with respect to the read-out and analysis speed are necessary.

IV. **Correct for systems non linearities and calibrate at different temperatures.** The non linearities of the delaylines are currently not corrected for, an implementation for correction should be implemented either in software or on the FPGA. The calibration of the system should also be done at different temperatures to compensate for heating of the system.

V. **System should be transferred to the final probe solution.** Finally transfer of the full system to the probe that a surgeon can use.

VI. **Design of a 2D SPAD array.** A 2D SPAD array (possibly with the functionality on chip) has the advantage of not needing complex scanning optics and increasing the speed for obtaining 2D images.


# List of Figures

1.1 Principle of fluorescence imaging. ........................................... 2
1.2 Absorption and emission in fluorescence / phosphorescence molecules. .......... 3
1.3 Absorption and emission spectra of ICG. ..................................... 4
1.4 Absorption spectra of ICG in water for different molar concentrations. ............ 5
1.5 Excitation and re-emission waveforms for frequency fluorescence lifetime measurements. 6
1.6 Comparison of the Gating and TCSPC technique to reconstruct the pulse. .......... 7
1.7 Single photon avalanche diode cross section and quenching circuits. ............... 9
1.8 The camera consists out of three boards: FPGA (and camera), delaylines and power. .. 10
1.9 FluoCAM. .......................................................... 10
1.10 LinoSPAD, a SPAD line with 256 pixels. ........................................ 10
1.11 LinoSPAD mainboard containing a Spartan 6 FPGA and FX3 USB3 controller. ....... 11
1.12 LinoSPAD mainboard. .................................................... 11
1.13 Fluorescence Lifetime Imaging systems in Microscopy. .......................... 13
1.14 Fluorescence Lifetime Imaging systems in Spectroscopy. ........................ 14
1.15 Fluorescence Imaging systems for (pre-)clinical applications. ..................... 14
1.16 LinoSPAD probe for clinical applications, including the miniaturized prototype components plus optics and light source. ................................. 15

2.1 A TDC converts the time between the arrival of the start and stop signal into a digital code. 19
2.2 Time counter, clock is a high frequency reference. .................................. 20
2.3 Tapped delayline TDC, start signal is displaced in time; stop signal freezes the time stamp. 20
2.4 Double tapped delayline TDC, start and stop signals are displaced in time; stop signal freezes the time stamp. ........................................ 21
2.5 Oscillator based TDC with two ring oscillators, a phase sync and a counter. ........ 21
2.6 Pulse shrinking based TDC, counting the pulses till it disappears. .................. 21
2.7 Carrychain delayline TDC, OR of SPADs as start is displaced in time; laser clock as stop signal freezes the time stamp. ................................. 23

3.1 Timing diagram of the Input filter. ........................................... 25
3.2 Input filter with reset feedback. ............................................ 26
3.3 Input filter with self reset. .................................................. 26
3.4 Input filter with robust self reset. .......................................... 26
3.5 Propagation of an input signal through a carrychain. ............................... 27
3.6 Implementation of a pipelined thermometer decoder. ............................... 28
3.7 Bubble correction circuit. ................................................... 29
3.8 Histogrammer to store time values. .......................................... 30
3.9 Histogrammer for multiplexing multiple channels; 2:1 multiplexing shown. .......... 30
3.10 LinoSPAD interface with FPGA, FX3 and the SPAD chip. ......................... 31
3.11 Detailed System Schematic for one channel. ..................................... 32

4.1 Simulation of LinoSPAD delay codes w.r.t. inserted delay times. .................. 34
4.2 Part of the implemented delayline. .......................................... 34
8.15 ICG-RGD in mouse ear 24 hours after injection (FluoCAM). 81
8.16 ICG-RGD extracted lifetimes [ps] from the mouse ear tumour 24 hours after injection. 81

9.1 The fluorescence exponential signal is mapped onto a LFSR random number generator, area of exponential corresponds to a certain number of random numbers. 84
9.2 Delayed clock signals and probability of occurrence with exponential behaviour. 84
9.3 Linear Feedback Shift Register for pseudo random number generation. 85
9.4 Emulator consists of one delayline, two LFSRs, exponentially dependent mux and one normal demux. 85
9.6 LinoSPAD test setup with fluorescence emulator and OneSPAD. 86
9.7 Measured exponential signals from the SP605 emulator, the correct exponentials are drawn as reference. 87
9.8 OneSPAD pulses and reference system clock. 88
9.9 DCR histogram as measured from the SPAD pulses with the LeCroy oscilloscope. 89
9.10 DCR histogram as measured from the SPAD pulses with the LinoSPAD FPGA system. 90
9.11 DNL & INL for OneSPAD generated histogram of Figure 9.10. 90
9.12 Optical projection of the fluorescence spot on the FluoCAM and LinoSPAD SPAD arrays. 91

B.1 8 TDC 256 channels implemented design FPGA layout. 107
B.2 System Schematic. 108
C.1 Randomly generated histograms with a lifetime of $\tau = 500$ ps and 2500 photons. 109
C.2 Inserted lifetime vs. Extracted lifetime for different nr. of photons. 110
C.3 Inserted lifetime vs. Extracted lifetime for different nr. of photons in the case of noise ($SNR = 26$ dB) being present. 110
C.4 Inserted lifetime vs. Extracted lifetime for different nr. of photons in the case of noise ($SNR = 20$ dB) being present. 110
C.5 Inserted lifetime vs. Extracted lifetime for different nr. of photons in the case of noise ($SNR = 20$ dB) and non linearities being present. 111
List of Tables

1.1 Comparison of the SPAD chips from FluoCAM and LinoSPAD. ........................................ 11
1.2 Comparison of fluorescence measurement systems ......................................................... 12
2.1 Comparison of possible TDC implementations. ................................................................. 23
3.1 Overview of the components included in the final (prototype) LinoSPAD system. ............. 32
4.1 Comparison of DNL & INL between simulation and hardware; implementation with 8 TDCs. ....................................................................................................................... 40
4.2 Comparison of FPGA TDCs with the LinoSPAD FPGA TDC. ........................................... 43
4.3 Tested multichannel configurations. .................................................................................... 44
5.1 Extracted lifetimes [ps] for different inserted lifetimes and Wiener constants. ................. 51
5.2 Comparison in speed and accuracy of Matlab’s exponential fit and LMS linear fit. ........ 53
5.3 Extracted lifetimes [ps] with the Centre of Mass method. .................................................. 55
5.4 Wiener vs. CoM, average error for different intensities over a range of inserted lifetimes. 57
5.5 Wiener vs. CoM, speed on different setups; both algorithms are written in C++ and run 256 times, in parallel if applicable (one SPAD line). ......................................................... 57
5.6 Wiener vs. CoM, average error for different intensities over a range of inserted lifetimes in the case of noise (SNR = 20 dB) being present. ................................................. 58
5.7 Wiener vs. CoM, average error for different intensities over a range of inserted lifetimes in the case of noise (SNR = 20 dB) and non linearities being present. ............... 58
7.1 Dilution Protocol ............................................................................................................... 65
8.1 Noise performance for different temperatures. ................................................................. 71
8.2 Delaylines resolution for different temperatures. .............................................................. 73
8.3 Linearity performance for different temperatures. ............................................................. 73
9.1 Pseudo random numbers generated with the LFSR of Figure 9.3 and the corresponding output delay signal. ................................................................................................. 85
9.2 Comparison of DNL & INL between simulation, average result from ringoscillator experi- ments and OneSPAD; implementation with 8 TDCs. ...................................................... 91
9.3 Comparison of the optical performance in FluoCAM and LinoSPAD. ............................ 92
10.1 The various system requirements that are achieved in the LinoSPAD prototype. .......... 94
C.1 Wiener vs. CoM, average error for different intensities in the case of noise (SNR = 26 dB) being present. ................................................................. 111
C.2 Wiener vs. CoM, average error for different intensities in the case of noise (SNR = 26 dB) and non linearities being present. ................................................................. 111
Appendix A

Design Brief

1. Introduction
A firmware will be developed for the purpose of Fluorescence Lifetime Imaging (FLIM) for medical applications. A chip LinoSPAD was developed with a SPAD line containing 256 pixels. Together with the chip a board was designed containing an FPGA for the implementation of the chip’s readout and analysis.
As the extraction of fluorescence lifetime information is one of the important system aspects, TDCs are implemented in the FPGA. This allows the enabling of Time Correlated Single Photon Counting (TCSPC), a technique to calculate timing information from individual photons. From the TCSPC histograms the lifetime of the fluorescence can be extracted.
In this document the requirements and specifications of the firmware will be specified. Besides the development of the firmware, separate development paths are considered for the optics, the systems hardware and the final prope.

2. Requirements derived from the intended use
2.1. The firmware must be implemented on a Xilinx Spartan 6 FPGA. The model xc6slx100-3fgg676 is used as it has adequate number of pins (over 256 are needed for all SPADs) and enough slices to implement the required functionality.
2.2. The system will be able to do TCSPC on a line of 256 SPADs. Therefore each SPAD should contain a histogram with the timing information.
2.3. The final implemented system is to be built inside a probe for the purpose of fluorescence guided surgery, the system will be powered by USB and data has to be transferred over an USB 3 interface.
2.4. The system can extract lifetimes downto 100 ps, the resolution of the TDCs should be adequate as to allow this extraction. The error on the extraction should be kept below 50 ps.
2.5. The TDCs must have a resolution of at least 100 ps, derived from extensive simulations on lifetime extraction. A resolution of 100 ps would allow lifetime extraction down to 100 ps, a significant error may be involved.
2.6. The detection range must be 10 ns, i.e. the clock period of the intended laser system. The average time between a laser trigger and the corresponding photon hit is roughly 1.5 ns. With a maximum of > 10 ns.
2.7. The speed or the amount of measurements that can be done is specified as 100 MS/s, while the speed of the laser trigger is 100 MHz.
The duty cycle of the laser trigger is roughly 50% and can be used as an input clock.
2.8. The non linearities of the system should not limit its functionality in lifetime extraction. Otherwise calibration and correction have to implemented, either in hardware or in software.
2.9. The firmware needs to employ the TDCs for TCSPC and store a histogram for every pixel. The histogram data will be transmitted to a PC for further analysis. A fall-back to the raw timing informations has to be available.
2.10. The ideal output will be the lifetime measured in every of the 256 pixels. At least histogram and raw timing data should be available.

2.11. The firmware needs not generate the control signals for the laser or for the SPAD line. This data is merely available as inputs to trigger both start and stop signals.

2.12. The power consumption should not limit the functionality of the device as such, while it is powered though USB or an external supply. However heating has to be concerned for the FPGA as well for the SPAD line as it leads to PVT variations. PVT calibration has to be considered in the case large variations in TDC resolution and non linearities are observed.

2.13. The final system needs to employ a Graphical User Interface (GUI) for the display of the measurement data, this is not included in the firmware, but should be standalone software on the PC. This GUI can at least show the histogram data, intensity and calculated lifetime. Lifetime extraction algorithms have to included in the PC software.

3. Requirements derived from the (ecological) situation of the system in its surroundings

3.1. While the final system will be used for surgery, the design has to comply to all legislations regarding the use of those systems.

3.2. While this document regards only the firmware / software and not the physical hardware, this constraint is not limiting any implementation.

3.3. The lifetime of the system will be determined by the lifetime of the components.

3.4. Requirements with respect to recycling and disposal are determined by the components of the system.

4. Requirements concerning the system as such

4.1. The stability of the firmware needs to be such that the data is in any way useful for its users.

4.2. The firmware has to be stable and easy to use by non-engineers.

4.3. The time the firmware has to run to do a measurement has yet to be determined from test data. The time has to be as short as possible to generate a histogram with enough data to extract the lifetime.

4.4. The firmware should be easily adjustable and easily implementable on the FPGA.

MoSCoW model draft

Must have
- All points mentioned in the design brief.

Should have
- Possibility of running all the desired analysis functions on board.
- Analysis include the extraction of the lifetime from the TCSPC histogram.

Could have
- Ideally one TDC for each pixel.

Would have (would be)
- A fully implemented firmware with all desired functions outputting only the lifetime data for all pixels, displaying this data real-time in a GUI on a PC.
- Realtime is defined as the time it takes to make a 2D image with video frame rates. Scanning optics is part of the final system.
Appendix B

Schematic & Layout

An overview of the total schematic of the systems firmware is given in Figure B.2. The 8 TDC 256 channels implemented FPGA layout is shown in Figure B.1. The TDCs can clearly be seen as the large blue lines.

Figure B.1: 8 TDC 256 channels implemented design FPGA layout.
Figure B.2: System Schematic.
Appendix C

Detailed lifetime analysis study

Figure C.1: Randomly generated histograms with a lifetime of $\tau = 500$ ps and 2500 photons.
Figure C.2: Inserted lifetime vs. Extracted lifetime for different nr. of photons.

(a) Wiener filter.  
(b) Centre of Mass.

Figure C.3: Inserted lifetime vs. Extracted lifetime for different nr. of photons in the case of noise (SNR = 26 dB) being present.

(a) Wiener filter.  
(b) Centre of Mass.

Figure C.4: Inserted lifetime vs. Extracted lifetime for different nr. of photons in the case of noise (SNR = 20 dB) being present.
Figure C.5: Inserted lifetime vs. Extracted lifetime for different nr. of photons in the case of noise ($SNR = 20$ dB) and non linearities being present.

Table C.1: Wiener vs. CoM, average error for different intensities in the case of noise ($SNR = 26$ dB) being present.

<table>
<thead>
<tr>
<th>Nr. of Photons</th>
<th>Nr. of Noise counts</th>
<th>Average error [ps]</th>
<th>Wiener</th>
<th>CoM</th>
</tr>
</thead>
<tbody>
<tr>
<td>50,000</td>
<td>2,500</td>
<td>16.2</td>
<td>18.4</td>
<td></td>
</tr>
<tr>
<td>25,000</td>
<td>1,250</td>
<td>19.9</td>
<td>20.3</td>
<td></td>
</tr>
<tr>
<td>10,000</td>
<td>500</td>
<td>20.3</td>
<td>18.4</td>
<td></td>
</tr>
<tr>
<td>5,000</td>
<td>250</td>
<td>29.5</td>
<td>16.7</td>
<td></td>
</tr>
<tr>
<td>2,500</td>
<td>125</td>
<td>39.1</td>
<td>24.9</td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>50</td>
<td>42.6</td>
<td>45.8</td>
<td></td>
</tr>
</tbody>
</table>

Table C.2: Wiener vs. CoM, average error for different intensities in the case of noise ($SNR = 26$ dB) and non linearities being present.

<table>
<thead>
<tr>
<th>Nr. of Photons</th>
<th>Nr. of Noise counts</th>
<th>Average error [ps]</th>
<th>Wiener</th>
<th>CoM</th>
</tr>
</thead>
<tbody>
<tr>
<td>50,000</td>
<td>2,500</td>
<td>33.7</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td>25,000</td>
<td>1,250</td>
<td>30.6</td>
<td>15.6</td>
<td></td>
</tr>
<tr>
<td>10,000</td>
<td>500</td>
<td>37.5</td>
<td>14.8</td>
<td></td>
</tr>
<tr>
<td>5,000</td>
<td>250</td>
<td>30.5</td>
<td>17.2</td>
<td></td>
</tr>
<tr>
<td>2,500</td>
<td>125</td>
<td>39.7</td>
<td>21.8</td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>50</td>
<td>46.3</td>
<td>35.4</td>
<td></td>
</tr>
</tbody>
</table>