



OULUN YLIOPISTO
UNIVERSITY of OULU

DEGREE PROGRAMME IN ELECTRICAL ENGINEERING

MASTER'S THESIS

FPGA CONTROLLED SPAD MATRIX TEST ENVIRONMENT

Author	Jouni Holma
Supervisor	Juha Kostamovaara
Second Examiner	Ilkka Nissinen

December 2013

ABSTRACT

Test environments for two SPAD matrix ICs are designed and implemented in this work. The matrixes are designed as a receiver of a Raman spectrometer. The aim of the work is a computer driven system that can perform SPAD matrix measurements according to the requirements.

The background of the test environments describes the function of the SPAD and the quenching circuit. Besides, the structures and functionalities of the related ICs are presented.

The work is about building two separate systems including computer software, a USB data converter on a PCB, an FPGA development board and a PCB for the IC. The user controls the measurements setting measurement parameters, monitors the measurements and reads the measurement results via the software. The results can be saved in an appropriate format that can be postprocessed in Matlab. The communication between the computer and the measurement tools runs over USB. The USB data converter is an interface between the USB and FPGA. The FPGA is a controller that reads the measurement results from the IC, stores them temporarily and sends them to the computer. Besides, the FPGA holds the measurement settings. The PCB around the IC is a platform that offers connectivity to voltages, control and data.

Measurements were performed in order to find out how suitable the SPAD matrixes are acting as a receiver of a Raman spectrometer. It was found that a timing signal of the time gated IC reaches all the elements within 60 ps. The timing homogeneity of the other IC was estimated based on the distributions of the TDC generated time window related code words. The mean parameters from the distributions showed the delay difference between the fastest and the slowest element was 180 ps.

Keywords: SPAD matrix, quenching circuit, IC testing

Holma J. (2013) FPGA:n ohjaama SPAD-matriisin testausympäristö. Oulun yliopisto, Sähkötekniikan osasto. Diplomityö, 63 s.

TIIVISTELMÄ

Tässä työssä suunnitellaan ja toteutetaan testausympäristö kahdelle IC-piirille rakennetulle SPAD-matriisille. Matriisit on suunniteltu Raman-spektrometrin vastaanottimeksi. Tavoitteena on tietokoneella ohjattava järjestelmä, joka kykenee suorittamaan mittauksia SPAD-matriiseilla niille asetettujen vaatimusten mukaisesti.

Testausympäristön taustaosiossa kuvataan SPADin ja sammutuspiirin toimintaa. Lisäksi molempien testausympäristöön liittyvien IC-piirien rakenteet ja toiminnallisuudet esitellään.

Varsinainessa työosiossa rakennetaan kaksi erillistä järjestelmää, joihin kuuluvat tietokoneen mittausohjelma, piirilevyllä toteutettu USB-datamuunnin, FPGA-kehitysalusta sekä IC:n ympärille rakennettu testikortti. Käyttäjä ohjaa mittauksia mittausohjelmalla syöttäen mittausparametreja, kontrolloimalla mittauksia ja lukemalla mittaustulokset. Tulokset voidaan tarvittaessa tallentaa muotoon, jota voidaan jatkokäsitellä esimerkiksi matlabilla. Kommunikatio mittausjärjestelmän kanssa tapahtuu USB-protokollan yli. USB-datamuunnin toimii rajapintana USB-protokollan ja FPGA:n välillä. FPGA toimii kontrollerina, jonka tehtävinä ovat mittaustulosten lukeminen IC-piiriltä, välivarastointi ja lähettäminen tietokoneelle sekä mittausasetusten asettaminen. IC:n ympärille rakennettu testikortti toimii fyysisenä alustana, joka tarjoaa IC:lle jännite-, ohjaus- ja dataliittimet.

Testausympäristössä suoritettiin mittauksia, joilla selvitettiin SPAD-matriisien soveltuvuutta Raman-spektrometrin vastaanottimeksi. Mittauksissa havaittiin, että aikaportititussa piirissä ajoitusmerkki saapui kaikkiin SPAD elementteihin 60 ps sisällä. Toisen piirin mittauksissa ajoituksen homogeenisyyttä arvioitiin TDC:n generoimien koodisanojen jakauman perusteella. Jakaumista poimittujen keskiarvoparametrin perusteella nopeimman ja hitaimman elementin välinen erotus oli 180 ps.

Avainsanat: SPAD-matriisi, sammutuspiiri, IC:n testaus

TABLE OF CONTENTS

ABSTRACT

TIIVISTELMÄ

TABLE OF CONTENTS

FOREWORD

ABBREVIATIONS

1. INTRODUCTION	7
2. SPAD MATRIX	8
2.1. Single photon avalanche diode	8
2.2. Quenching circuitry	8
2.3. Time-gated 128x8 SPAD matrix	9
2.4. 128x4 SPAD matrix with a Time-to-Digital Converter	10
3. TEST ENVIRONMENT	14
3.1. Specification	14
3.2. Time-gated SPAD matrix test environment	17
3.2.1. Message protocol	17
3.2.2. FPGA logic	18
3.2.3. Computer software	21
3.3. SPAD matrix with TDC test environment	22
3.3.1. Test platform PCB of SPAD matrix IC	22
3.3.2. Message protocol	23
3.3.3. FPGA logic	24
3.3.4. Computer software	26
4. MEASUREMENTS	28
4.1. Time-gated measurements	28
4.1.1. Dark count measurement	28
4.1.2. Laser pulsed measurement	29
4.2. Time interval measurements	31
4.2.1. Laser pulsed measurements	31
4.2.2. Crosstalk measurement	33
5. MODELLING THE MEASUREMENTS	35
5.1. Discrete convolution and deconvolution	35
5.2. Measurement model	35
5.3. Model accuracy	37
5.4. Simulation	38
6. DISCUSSION	41

7. CONCLUSION	43
8. REFERENCES	44
Appendices	46

FOREWORD

This work is done in the Electronics laboratory of the Oulu University. For me, it has not only been a part of a project but also mental grow up from a student towards an engineer. The work has offered varied challenges which have made it interesting.

I got the position to this project from academic professor Juha Kostamovaara, who is also the supervisor of this thesis. I want to thank him for this opportunity and many advice during the work. I also thank Dr Ilkka Nissinen for his support throughout the thesis project. I thank Dr Jan Nissinen who has often helped me. As far I have been working in the Electronics laboratory, there have been helping hands when I had noticed to ask for it. I sincerely thank all who have helped me during the work. I also thank my friends and closest people for their support outside the office time.

Oulu, Finland, on Monday 2nd December, 2013,

Jouni Holma

ABBREVIATIONS

The abbreviations of the collaborators of this project and the units of the SI-system are excluded from this list.

SPAD	Single photon avalanche diode
IC	Integrated circuit
PCB	Printed circuit board
FPGA	Field programmable gate array
USB	Universal serial bus
DCR	Dark count rate
pMOS	P-channel metal oxide semiconductor
nMOS	N-channel metal oxide semiconductor
CMOS	Complementary metal oxide semiconductor
TDC	Time-to-digital converter
LED	Light emitting diode
FIFO	First-in, first-out
DLL	Dynamic-link library
MFSM	Main finite state machine
RAM	Random access memory
UI	User interface
BNC	Bayonet Neill-Concelman (connector)
GUI	Graphical user interface
V_{br}	Breakdown voltage
V_{bias}	Bias voltage
V_{anode}	Anode voltage
V_{DD}	Operating voltage

1. INTRODUCTION

This work is a part of a project funded by the TEKES. The project has the collaborators of the Optoelectronics Research Centre (ORC) of the Tampere University of Technology, the Electronics laboratory of the Oulu University and VTT electronics. The project is about the pre-commercial development of time gated Raman spectroscopy. The project has the remits of laser development, optical receiver development and system level development, whose responsible collaborators are the ORC, the Electronics laboratory and the VTT, respectively.

Within the project, the Raman spectrometer optical receiver functionality is based on a matrix of small sensitive optical sensors, namely single photon avalanche diodes (SPAD), that can detect weak Raman scattering. When the Raman scattered light is broken into colors, the different wavelengths are spatially spread and can be distinguished by the SPAD matrix. The temporal profile of the Raman scattering is separated from distracting fluorescence using time gating which means limiting the active time of the SPADs to the active Raman area and blocking the fluorescence.

This work is about developing a test environment for two integrated circuits (IC) that are designed to be an optical receiver of the Raman spectrometer. The test environments are built in order to verify new ICs's functionalities and perform measurements. User can control the measurements from a computer. The ICs are placed on a printed circuit board (PCB) and controlled by a field programmable gate array (FPGA) logic. The data runs over the universal serial bus (USB) that has a special converter PCB at the FPGA end of the cable.

The work includes also some measurements that study the characteristics of the SPAD matrixes. The characteristics can be used to analyze the SPAD matrix performance as an optical receiver of the Raman spectrometer. Besides, a mathematical model of a measurement is presented. Modelling the measurement deepens the understanding of the measurement conditions and helps the analysis of the measurement results.

At the beginning of this document, what the SPAD matrixes are about, is presented. Then, the test environments are specified and introduced. Some measurements are performed and showed. A model of a measurement is derived and represented. At the end, there is a review of the work and some analysis of the measurement results.

2. SPAD MATRIX

2.1. Single photon avalanche diode

Single photon avalanche diode (SPAD) is a type of avalanche photodiodes which are operating in Geiger-mode. In Geiger-mode, the avalanche photodiode is reverse biased with a voltage higher than the breakdown voltage V_{br} . At these conditions, a single charge carrier in the depletion region of the SPAD can trigger the avalanche breakdown. The charge carriers can be injected to the depletion region thermally or by the absorption of a photon. [1] The breakdown is self-sustaining and continues as far as the reverse bias is above the breakdown voltage or the current through the diode is limited below a critical boundary. Decreasing the bias or the current in order to terminate the breakdown is called quenching.

Non photon generated breakdowns are called dark counts. As the SPADs are semiconductors, the number of thermally released charge carriers increases as the temperature rises. Besides, lifting the reverse bias increases the number of dark counts. The measure of the dark counts is called dark count rate (DCR) and its unit is Hz.

2.2. Quenching circuitry

As the SPAD can respond to single photons, it can be used as a sensitive light detector. Detecting several photons requires quenching after each breakdown. Quenching the breakdown requires additional circuitry around the SPAD. The literature knows a couple of quenching circuits like passive and active quenching. [2] The selection of the quenching circuit depends on the application needs. Figure 1 represents a simple time-gated quenching circuit.

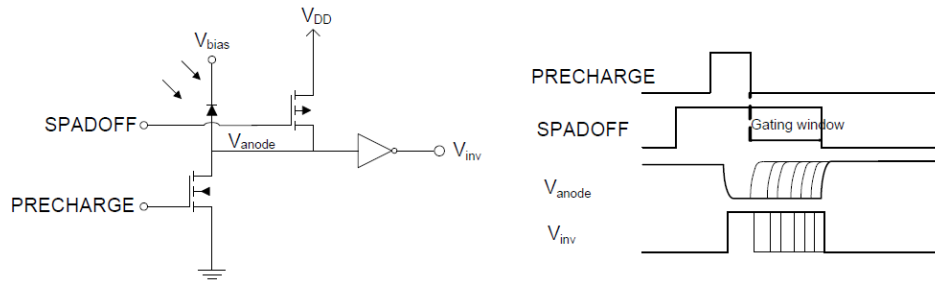


Figure 1. A simple time gated quenching circuit. [3]

The circuit represented in Figure 1 can activate the SPAD for a specific time window. V_{bias} is above V_{br} of the SPAD. Outside the time window, V_{anode} is tied to V_{DD} .

When the time window begins, *SPADOFF* sets the pMOS transistor in high impedance and *PRECHARGE* opens a path from V_{anode} to the ground via the nMOS transistor. As V_{anode} reaches the ground level, the SPAD enters to the Geiger-mode. Further, *PRECHARGE* closes the nMOS leaving V_{anode} on the ground potential but prevents a short circuit from V_{bias} to the ground. If the breakdown occurs, V_{anode} increases until the voltage over the SPAD becomes lower than V_{br} and the breakdown is quenched.

If there is no breakdown, *SPADOFF* disables the Geiger-mode setting the pMOS conductive and thus V_{anode} to V_{DD} .

The *Gating window* in Figure 1 enables adding temporal resolution to the avalanche. As there are several timing marks named time gates, the avalanche can be temporally localized between two gates.

2.3. Time-gated 128x8 SPAD matrix

This section introduces a time gated 128 x 8 pixel SPAD matrix with a serial data interface. [3] The matrix is implemented on an IC in a high voltage 350 nm CMOS technology. The physical dimension of the IC is 4,3 mm x 2,95 mm. The SPAD matrix is designed to be an optical receiver of Raman spectroscopy and, in practise, to measure the quantity and the timing of photons. The functional structure of the IC is represented in Figure 2.

The functional interface between the IC and the external electronics consists of four time window signals, four time signals and four data interface signals. The rising edges of the time window signals determine the length of the time window and the precharge time as shown in Figure 1. The time signals create references where the moment of the potential breakdown can be compared. The time signals are called gates later in this document. The gate signals are timed to rise after the quenching circuit is charged. The gating window in Figure 1 is the region where the gates are expected to rise. The data interface is detailed later in this section.

The IC has five different functional units which are the matrix, arbiters, a logic block and pulse generators for the time windows and time signals. As shown in Figure 2, the rows of the matrix are organized to groups of four SPADs. Inside a group, the four SPADs share the quenching circuitry, that is represented in Figure 1, and have a common output. The output signal signifies the moment of the breakdown and is lead to an arbiter. The arbiter compares the arriving time of the output signal and the rising edges of the gate signals. If the breakdown occurs before the gate, a pulse is generated and lead to the logic block.

The time window consists of the *SPADOFF* and *PRECHARGE* signals in Figure 1. The time window signals are generated from four external signals as it is shown in Figure 3.

The logic block consists of asynchronous 3-bit counters and an output register. As there are 128 rows in the matrix, two SPAD groups in a row and four time gates for a group, the logic block has 1024 counters in total. When an arbiter generates a pulse, the corresponding counter will be incremented. The counters act the internal memory of the IC and imply the number of breakdowns occurred after the latest readout. As the size of the counters is 3 bits, an overflow occurs if there are more than seven breakdowns in a single SPAD element without readout.

The IC has a 4-wire serial data interface at 12,5 MHz. The data interface is connected to a processor or corresponding unit that controls the operation of the SPAD matrix IC. Its timing diagram is represented in Figure 4.

The falling edge of the *aload* is synchronized to the *clock* and implies the moment when the data is going to be read out. The results of the counters are loaded into the output shift register and the counters are reset. A dashed vertical line in Figure 4

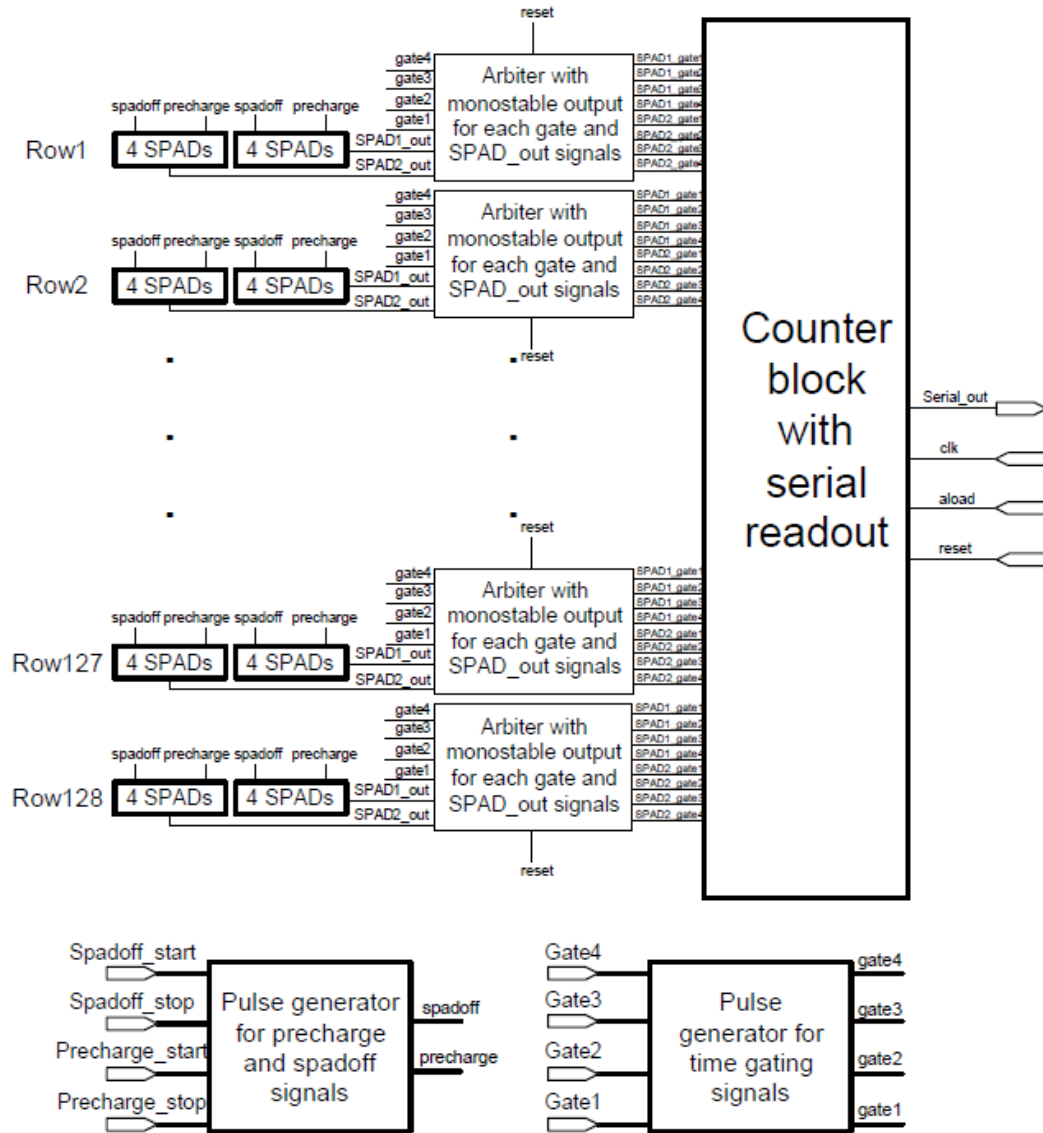


Figure 2. The structure of the time gated 128 x 8 pixel SPAD matrix IC. [3]

implies the sampling of the first bit in the external device. The *reset* is used to clear the output shift register. The data is read row-wise and a row contains information represented in Figure 5. The right most bit is read first and it is the least significant bit of the 3-bit count. The data readout is independent of the counting of the breakdowns. Thus, as the data is shifted out, the counters can meanwhile count new breakdowns.

2.4. 128x4 SPAD matrix with a Time-to-Digital Converter

A SPAD matrix of 128 x 4 pixels with a time-to-digital converter (TDC) and a parallel data interface is introduced in this section. [4] The matrix is factored in a high voltage 350 nm CMOS technology into an IC. The size of the IC is 5,0 mm x 1,9 mm. The

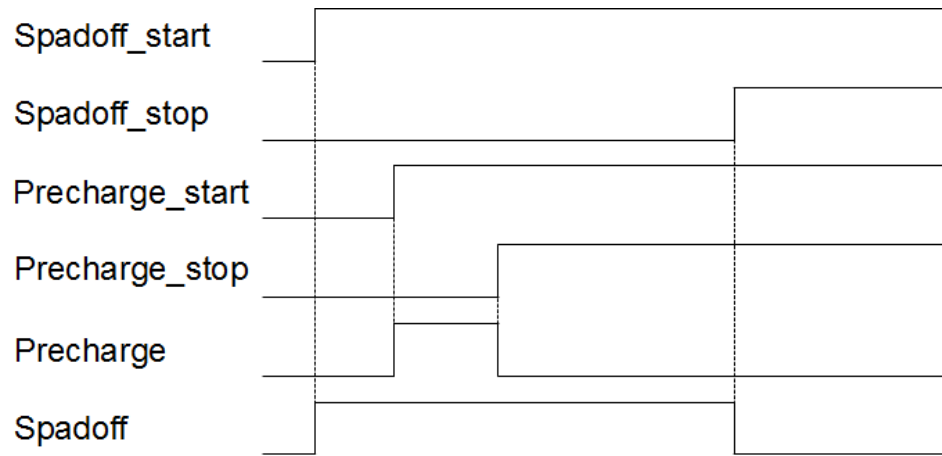


Figure 3. The generation of the time window signals.

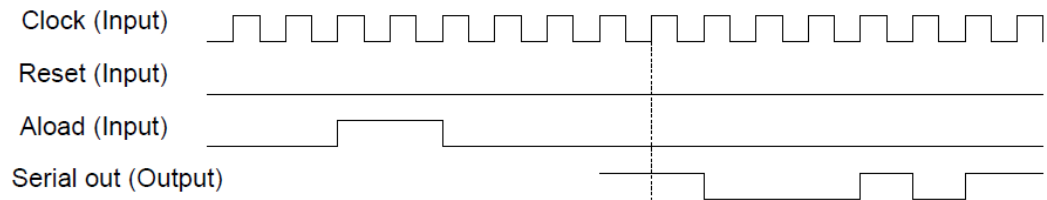


Figure 4. The timing of the data output logic.

block diagram of the SPAD matrix is presented in Figure 6. The TDC in the IC measures time interval between a reference signal *start* and the breakdown.

The operation of the SPAD matrix is based on 128 x 4 independent SPADs, quenching circuitry, external time signals and a TDC. The matrix utilizes similar quenching circuitry as represented in Figure 1. The external signals *Spadoff_start*, *Spadoff_stop*, *Precharge_start* and *Spadoff_start* define the active time window of the SPAD operation as it is represented in Figures 1 and 3. The TDC is launched by the *start* and the result of the delay line is stored as the breakdown occurs. The stored delay line values are encoded in 3-bit bytes whose relationship to the time is represented in Table 1.

The parallel data interface has a twelve wire data bus and a seven wire address bus. The data bus shows the data content of a SPAD matrix row. As there are four SPADs at each row in the matrix and each element has 3-bit word, there are twelve wires in total. The seven wires of the address bus can identify the 128 rows of the matrix. The address can be changed at the frequency up to 25 MHz.



Figure 5. The information of a matrix row.

Table 1. The output codes of the TDC at 100 MHz reference clock. [5]

Time interval	Output code
0 ps - 78 ps	000
78 ps - 156 ps	001
156 ps - 234 ps	010
234 ps - 312,5 ps	011
312,5 ps - 2578 ps	100
2578 ps - 2890,5 ps	101
2890,5 ps - 3515,5 ps	110
3515,5 ps - full range	111

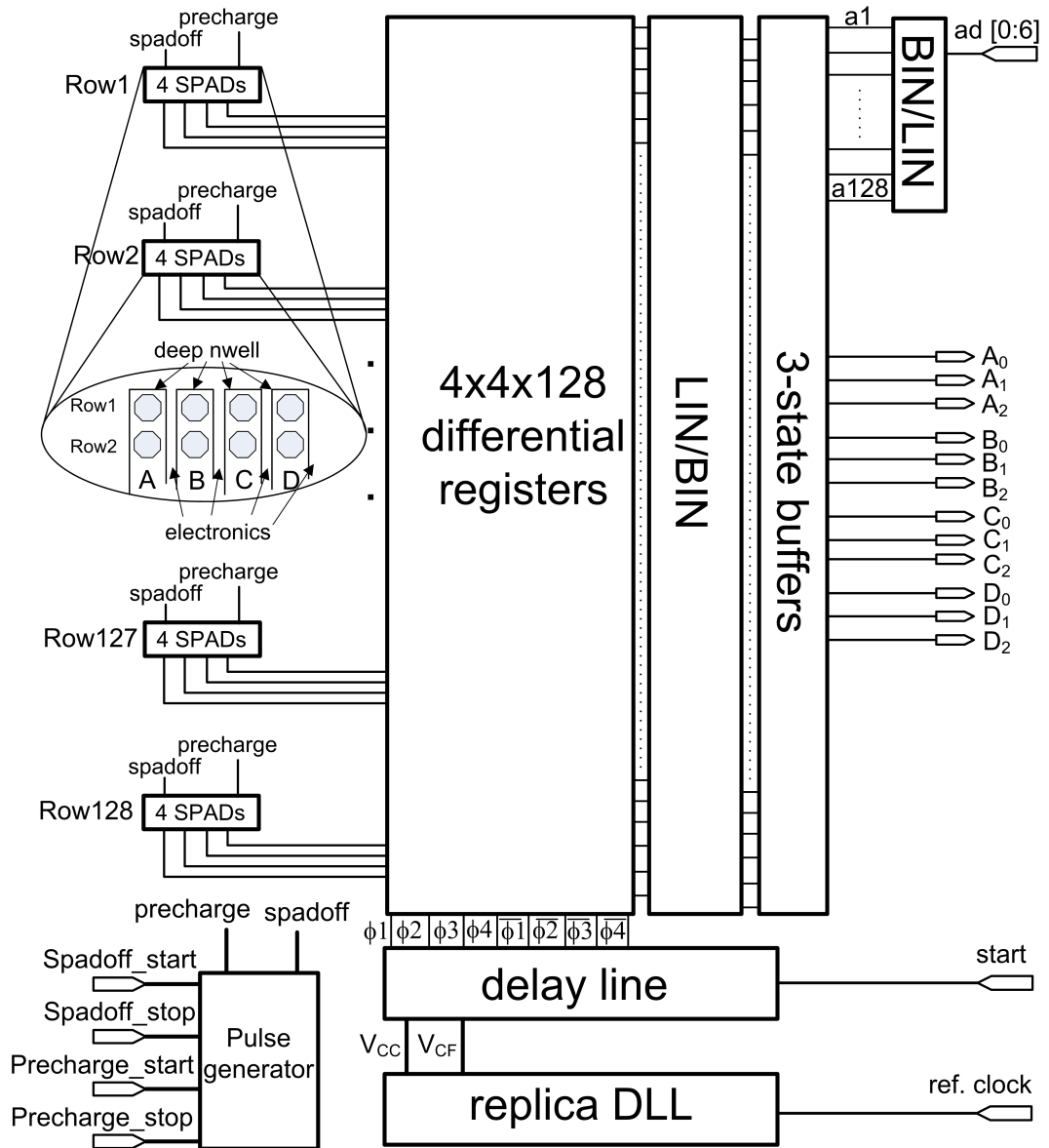


Figure 6. The structure of the 128 x 4 pixel SPAD matrix with TDC. [4]

3. TEST ENVIRONMENT

This chapter describes the requirements and the implementations of the test environments for the SPAD matrix ICs introduced in the sections 2.3 and 2.4. The specification section 3.1 covers both ICs. The detailed realizations of two test environments are represented in sections 3.2 and 3.3 separately.

3.1. Specification

The test environment performs measurements and saves the measurement results on the devices described in the sections 2.3 and 2.4. The measurements are controlled from the computer and the measurement results are stored in a format suitable for further processing. The repetition cycle of the measurements is at least 100 kHz. The data lines of the SPAD matrixes can be driven at the frequency of 12,5MHz.

According to the requirements of the test environment and the previous experience of the author of this document, a block diagram of Figure 7 was formed as a frame of the work.

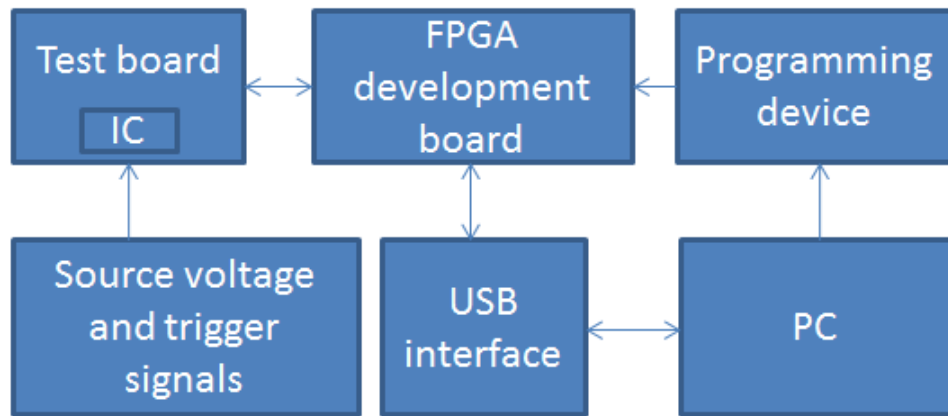


Figure 7. The block diagram of the test environment.

The test environment consists of a test board, an FPGA development board, a USB interface, a computer and their interfaces. The voltage and signal sources are also a part of the test environment but they are not built specially for this project. The programming device is needed, when the FPGA is being programmed, but its role is reduced as a voltage source of the FPGA, when the test environment is running.

The test board acts as a physical and electrical platform of the SPAD matrix IC. Physically, the test board is a PCB where the IC is bonded. The test board offers the IC all the voltages and signals that are needed. The voltages consist of the high voltage, operating voltage and ground. The signals cover both the quenching signals and the signals needed in the data interface. The quenching consists of four signals that are four top most signals in Figure 3 on the page 11. Besides, the SPAD matrix with a TDC represented in the section 2.4 on the page 10 has a reference clock and *start TDC* signals that are needed to the functionality of the TDC and represented in Figure 6 on the page 13. The data interface is IC specific.

The test board has external connectors to the high voltage, the operating voltage, the quenching signal and the FPGA. The IC, represented in the section 2.4, has also additional connectors to the reference clock and the *start TDC*. The test board divides the quenching signal into four signals mentioned in the previous paragraph. The FPGA connectors include the signals needed in the data interface and the quenching signal. The FPGA can count the number of the quenches from the quenching signal. The test board of the SPAD matrix presented in the section 2.3 was designed before this work. Thus, it is not presented in more detail in this document.

The FPGA acts as an intelligent data buffer between the IC and the computer. As the communication between the FPGA and the computer is chosen to run over the USB and the FPGA has no support for the USB protocol, the FPGA is connected to an additional USB converter that handles the protocol and is implemented on a separate PCB. Thus, the FPGA development board has interfaces to the USB converter and to the test board as detailed in the last paragraph. The interface to the USB converter is specified meanwhile the USB converter PCB is specified. The FPGA performs the instructions that the computer sends via the USB. The FPGA logic can read the measurement data from the SPAD matrix IC via the data interface, temporary store the data and resend it to the computer. The platform of the FPGA logic is Altera DE0-Nano Development and Education board. The DE-0 Nano button is set to be asynchronous reset of the FPGA logic. The LEDs of the DE-0 Nano can be used as an indicator of the logic state.

The USB converter is a system that converts the data of the universal serial bus to a parallel bus of 8 bits and vice versa. It is implemented on a PCB which has a USB-B female connector for the USB connectivity and a 2x20 female pin header connector for the parallel data interface to the FPGA. The USB protocol is handled by a multipurpose USB converter IC. [6] Using the terms of the datasheet [6], the interface between the FPGA and the USB converter realizes the FT245 Style Asynchronous FIFO Interface whose reading and writing timing diagrams are represented in Figures 8 and 9. The schematics of the PCB is built according to the USB bus powered configuration of the datasheet [6]. The USB converter has reception and transmission buffers in order to handle the bi-directional data traffic. The data from the FPGA is temporally stored in a reception buffer of the USB converter before it is sent to the computer. The bytes sent from the computer are temporally stored in a transmission buffer before they are sent to the FPGA.

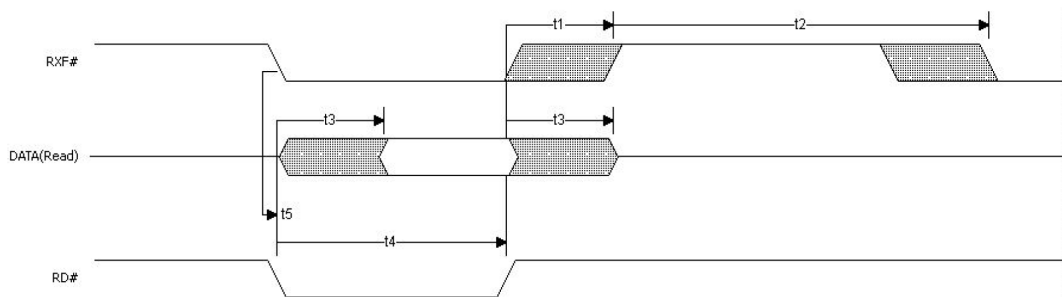


Figure 8. The timing diagram of the FT245 style asynchronous FIFO interface in the read mode.

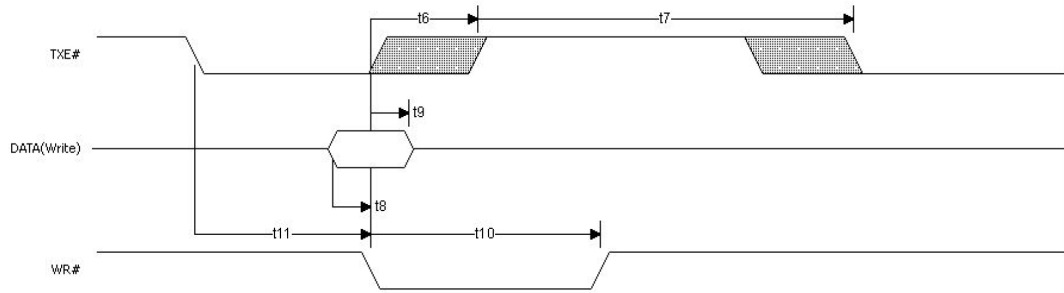


Figure 9. The timing diagram of the FT245 style asynchronous FIFO interface in the write mode.

From the view of the FPGA, the USB converter acts as an 8-bit wide FIFO (first in, first out) buffer. The USB converter has a FIFO buffer for both read and write data. Both buffers share a 8-bit wide bi-directional data bus which is later called *adbus*. The content of *adbus* is represented as *DATA(read)* and *DATA(write)* in Figures 8 and 9, respectively. The interface between the USB converter and the FPGA consists of *adbus* and four other signals from Figures 8 and 9 that form the FT245 style asynchronous FIFO interface. *rxif* and *txe* are outputs of the USB converter and the *rd* and the *wr* are its inputs.

As the USB interface has data in its read FIFO, it sets *rxif* down implying there is data available. The FPGA allows the USB converter to write data on *adbus* by setting *rd* down. Setting up *rd* implies that the FPGA has successfully read the byte. At the end, the USB converter rises up *rxif*.

When the USB interface is expecting data from the FPGA, it sets *txe* down. The response of the FPGA is to set the data on *adbus*. Setting *wr* down writes the data from *adbus* into the write FIFO of the USB converter. Rising of *txe* implies a successful writing into the FIFO.

The USB converter handles the USB protocol and converts the parallel data to the USB form and vice versa. The data runs over the USB in packets. The more detailed description of the USB protocol is outside of the scope of this work as the complexity of the protocol is hidden from the FPGA logic designer as well as the computer software developer. All the FPGA logic designer needs to know about the USB converter is the timing diagrams of Figures 8 and 9.

The desktop program includes a user interface controlling the measurements, reading the data from the FPGA, representing the measurement results and saving them. The measurements are controlled by instruction bytes which are sent to the FPGA via the USB. The measurement data is organized in frames that can be identified from the USB packets, that are read from the USB converter.

The desktop program controls the measurements and reads, represents and saves the data. The software is implemented in the Visual Studio 2010 environment. The program uses a FTDI's D2XX header file which runs over the dynamic-link library (DLL). The Windows driver model architecture is represented in Figure 10. [7] The driver handles the USB protocol from the computer.

Practically, the developer uses application programming interface of the FTDI which includes commands to build a connection to the FTDI USB device, configure the USB

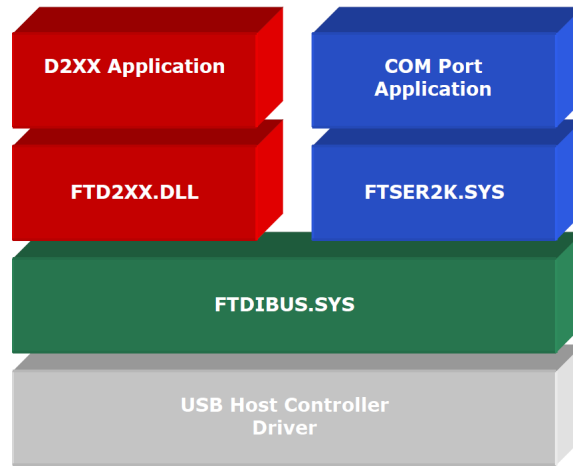


Figure 10. The Windows driver model architecture. [7]

device in a desired mode, write the instruction bytes into the transmission buffer and read data from the reception buffer. Those commands are defined in the reference [7].

3.2. Time-gated SPAD matrix test environment

This section describes the test environment of the IC represented in the section 2.3. The test environment consists of computer software, a USB converter on a PCB, an FPGA logic and a PCB for the SPAD matrix IC. The communication inside the test environment consists of control messages and data frames. The control messages are sent from the computer to the FPGA and the data frames are sent in the opposite direction.

In this section, the communication between the computer and the FPGA is represented in the subsection 3.2.1. As the test board was not a part of this work, its representation is limited to the necessary as an interface to the FPGA. The functionalities of the FPGA logic and the computer program are represented in the subsections 3.2.2 and 3.2.3, respectively.

3.2.1. Message protocol

The communication between the computer and the FPGA consists of measurement control and data flow. The computer sends instructions down to the FPGA and receives the data organized in frames from there.

The instruction set of the FPGA is represented in Table 2. The columns are the name of the instruction, the hexadecimal value of the instruction, the length of the instruction in bytes and a brief description of the instruction. The functionalities of the instructions are detailed in the section 3.2.2.

The data is organized in a frame during the transmission from the FPGA to the computer. The frame structure is built over the USB protocol. The frame consists of a preamble, the total amount of pulses fed to the IC and the data separated by bit stuff bytes. The structure of the frame is illustrated in Figure 11. The preamble enables

Table 2. The instruction set of the FPGA

Instruction	Hex	Class	Bytes	Description
RESET	0x20	Reset	1	Reset the logic
SET PULSE	0x40	Set	2	Set the number of pulses before the IC is read
SET MEAS	0x60	Set	5	Set the number of IC read cycles
READ	0x80	Action	1	Read data to the computer
MEAS	0xA0	Action	1	Start the measurement

the detection of the beginning of the frame when it is transmitted inside a bigger USB packet. The bit stuff bytes prevent the receiver from mixing the data with the preamble. The number of the total amount of the pulses is given in order to get reasonable data if the measurement is interrupted. A frame includes 2048 data bytes as a word consist of two bytes and there are 1024 different values to be stored as described in the sections 2.3 and 3.2.2.

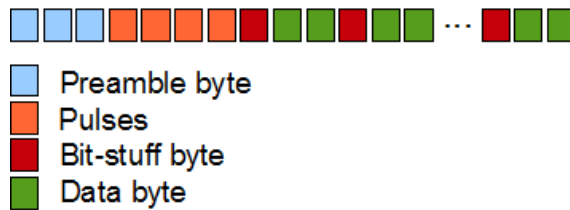


Figure 11. The structure of the data frame.

3.2.2. FPGA logic

This subsection represents the FPGA logic of the design and details the actions that the instructions of Table 2 represents.

The FPGA works as a data buffer between the SPAD matrix IC and the USB converter. Its functional block diagram is represented in Figure 12. The block diagram shows the functional units of the logic and roughly illustrates their mutual relations. A detailed port description is attached to the Appendix 1.

The *USB interface* in Figure 12 implements the FPGA side of the FT245 style asynchronous FIFO interface whose timing diagrams were represented in Figures 8 and 9. The data bus has a width of a byte and the interface is driven by the USB converter. The *USB interface* reacts to the handshakes of the USB converter. As responsible to the FPGA system, the *USB interface* indicates when a byte is sent or received.

The *instruction decoder* is the first block that handles the received bytes. It decodes the bytes according to Table 2 and generates a signal, that respects to the instruction, to the main finite state machine (*MFSM*). The *instruction decoder* holds the instruction as far as the *MFSM* acknowledges the instruction or a new instruction replaces the old one. The *MFSM* can prevent the *instruction decoder* from decoding received bytes to instructions during the data reception of the *SET* instructions of Table 2. The *SET PULSE* defines the number of quenching cycles before the SPAD matrix is read. The

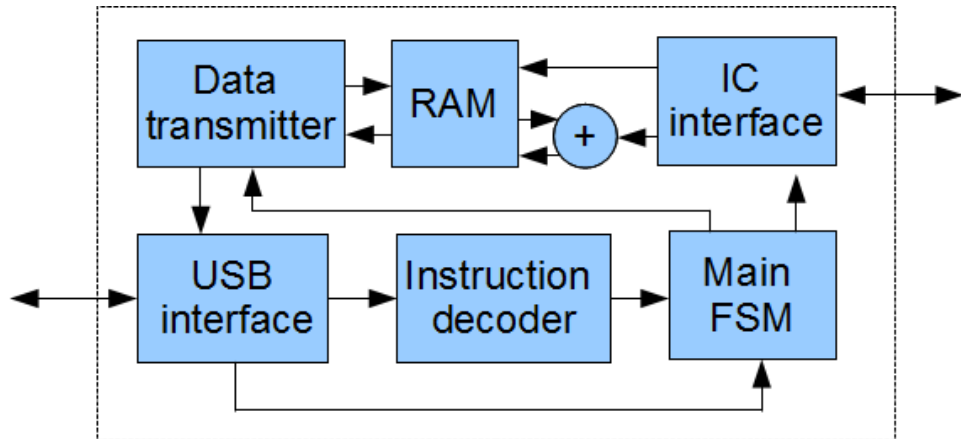


Figure 12. The functional block diagram of the FPGA logic.

maximum value of the *SET PULSE* is 255 and defined in a byte. It has to be noted, that the counter of the SPAD matrix has 3 bits and may overflow. The *SET MEAS* defines the number of the SPAD matrix read. The maximum value of the *SET MEAS* is 32 bit wide and needs four bytes. The big values in the *SET MEAS* may cause an overflow of the RAM of 16-bit wide memory if the photon detection propability is high.

The *MFSM* controls the top level functionality of the FPGA logic according to the instructions. The instructions can be divided in three classes: reset, set and action instructions. The reset instruction has a higher priority than the other instructions in order to stop a process started by an action instruction. The set instruction sets the measurement parameter and consists of two parts: the instruction and the data. The instruction part sets the *MFSM* in a data reception mode. During the set data reception, the instruction related measurement parameter is updated by the received data and the *instruction decoder* is forced not to decode any instruction. The responsible blocks for the *READ* and *MEAS* action instructions are the *data transmitter* and the *IC interface*, respondingly. The action instructions launches a procedure in the responsible blocks and the *MFSM* holds as far as the procedure is done. The state transition diagram of the *MFSM* is represented in the 13. The detailed description of the state transition diagram conditions and signals is represented in the Appendix 2.

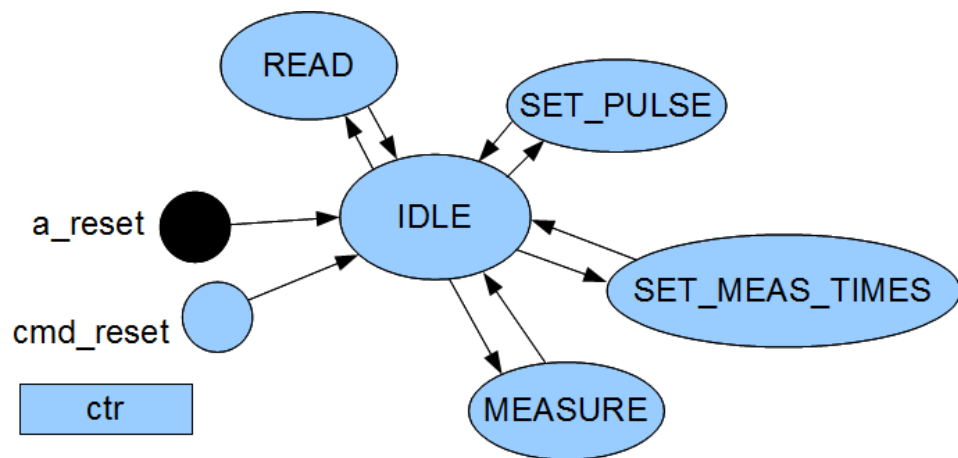


Figure 13. The state transition diagram.

The *IC interface* reads the data from the IC and stores the data into the random access memory (*RAM*). The interface between the IC and the FPGA consists of the signals represented in Figure 4 on the page 11 and a quenching indicator in order to count the number of quench cycles of the IC. As the IC is quenched predefined times according to the *SET PULSE* parameter, the *IC interface* reads data from the IC according to the timing of Figure 4. After the IC is read, the *IC interface* resets the IC according to the timing diagram of Figure 14. The *sQuench* is a synchronized quenching signal. Keeping the reset active until the quenching signal arrives prevents the SPAD matrix from counting avalanches during the reset. The reset initializes the results that the SPAD matrix counts during the reading.

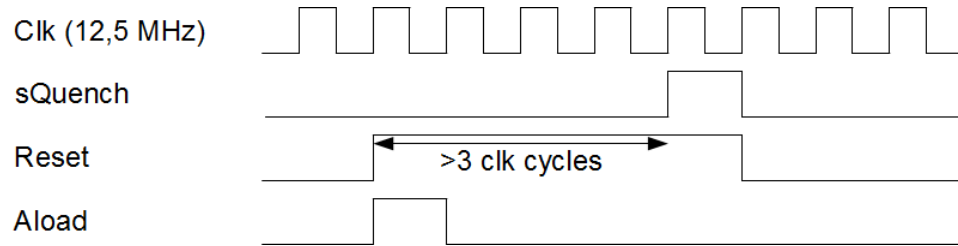


Figure 14. The timing diagram of the IC resetting.

The read operation is done predefined times according to the *SET MEAS* parameter. As soon as read operations are done, the measurement done indicator is generated to the *MFSM*.

As the SPAD matrix counts the number of hits, the results of consecutive read cycles can be totaled. The accumulation is done using a feedback structure between the *RAM* and the *adder*. At first, the old data is read from the *RAM*, then it is totaled with the new paralleled data and, at the end, the sum is written back to the *RAM* overwriting the old data. The *RAM* has no initializer itself. When a complete new measurement begins, the *adder* rejects the feedback data from the *RAM* during the first IC read cycle by a signal from the *IC interface*. As the old data is overwritten by the new data, the *RAM* is practically initialized. The *RAM* can be initialized only by reprogramming the FPGA.

The *data transmitter* receives an order from the *MFSM* to begin data transmission to the computer and sends notice back when the transmission is done. The *data transmitter* forms the data frame structure defined in Figure 11 on the page 18. At first, the *data transmitter* generates the preamble. Then, it adds the measured number of the quenching cycles into the stream. As the bus of the measured pulses is 32 bits wide, the count is split in four 8-bit bytes. The rest of the frame consists of the data separated by bit stuff bytes. As the width of the *RAM* is 16 bits and the width of the *adbus* is 8 bits, the data of the *RAM* is split into two 8-bit bytes.

The *IC interface* and the *data transmitter* have an access to the read address of the *RAM*. The simultaneous use of the read address is prevented by a multiplexer controlled from the *data transmitter*.

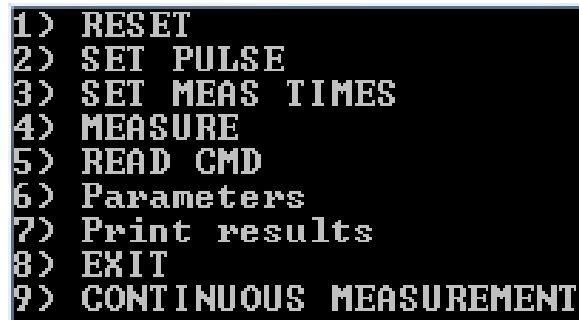
The *RESET* sets the *MFSM* in an idle state from any other state. During the transition to the idle state, the *MFSM* generates a reset signal to the *IC interface* and the *data transmitter* setting them in an idle state if a process is ongoing. The *RESET* does not change the measurement parameters or the content of the *RAM*.

The *SET PULSE* consists of two bytes. The first sets the *MFSM* in a pulse setting state from the idle state and the second is placed in a register that limits the number of quenching cycles before the SPAD matrix IC is read.

The *SET MEAS* is formed from five bytes. The first sets the *MFSM* in a measurement setting state from the idle state and the rest are placed in a register that limits the number of the reading cycles of the IC.

3.2.3. Computer software

The desktop software is a console application. The user interface (UI) of the program is represented in Figure 15.



```

1) RESET
2) SET PULSE
3) SET MEAS TIMES
4) MEASURE
5) READ CMD
6) Parameters
7) Print results
8) EXIT
9) CONTINUOUS MEASUREMENT

```

Figure 15. The user interface of the software.

The first four options of the UI are direct copies of the instructions from Table 2. Choosing the option sends a corresponding instruction byte to the USB. As the set option is chosen, the software asks the parameter that is going to be changed.

The fifth option sends the read instruction to the FPGA. The software polls the USB converter if it has data in the reception buffer. As the data is found in the reception buffer, it is transferred to the computer via the USB. As a USB packet is received on the computer, the next phase is to find the data frame inside the USB packet. As the preamble is found, the data is extracted from the frame and printed on the screen. The software asks the user if the data would be saved. If the preamble is not found or the frame is partly inside the USB packet, the read instruction will be sent again. If the data extraction is not managed in a sufficient time, the software reports the data is not found.

The data is saved in text files. The data of each time gate is save in a separate file. The data can be read and further processed in Matlab environment, for example.

The sixth option prints the measurement parameters that are set in the second and the third option. The seventh operation prints the latest results on the screen. The eight option closes the program.

The ninth option asks how many times the measurement will be done using these parameters, and collects the results together. This operation increases the number of pulses that can be measured as the width of the FPGA's *RAM* is 16 bits.

3.3. SPAD matrix with TDC test environment

This section describes the test environment of the SPAD matrix IC introduced in the section 2.4 on the page 10. As in the previous section, the communication protocol, the FPGA logic and the computer software are represented in the subsections 3.3.2, 3.3.3 and 3.3.4, respectively. Besides, the test board of the IC is introduced in the subsection 3.3.1.

3.3.1. Test platform PCB of SPAD matrix IC

The test card PCB acts as a physical platform of the SPAD matrix IC, powers the IC, generates the quenching signals and offers an interface to the FPGA. The schematics, the layout and the component placing of the test card PCB are represented in the Appendix 3, the Appendix 4 and the Appendix 5, respectively.

Figure 16 shows the component side of the PCB. The PCB has Bayonet Neill-Concelman (BNC) connectors to two operating voltages, a high voltage, the quenching signal, the reference clock of the TDC and the *start TDC*. A 20x2 pin header connector at the right side of Figure 16 is the interface to the FPGA. Besides, there are optional pin header connectors that are wired to a single SPAD that is in the IC separated from the matrix. The reference clock, the *start TDC* and the quenching inputs are handled as transmission lines and terminated.

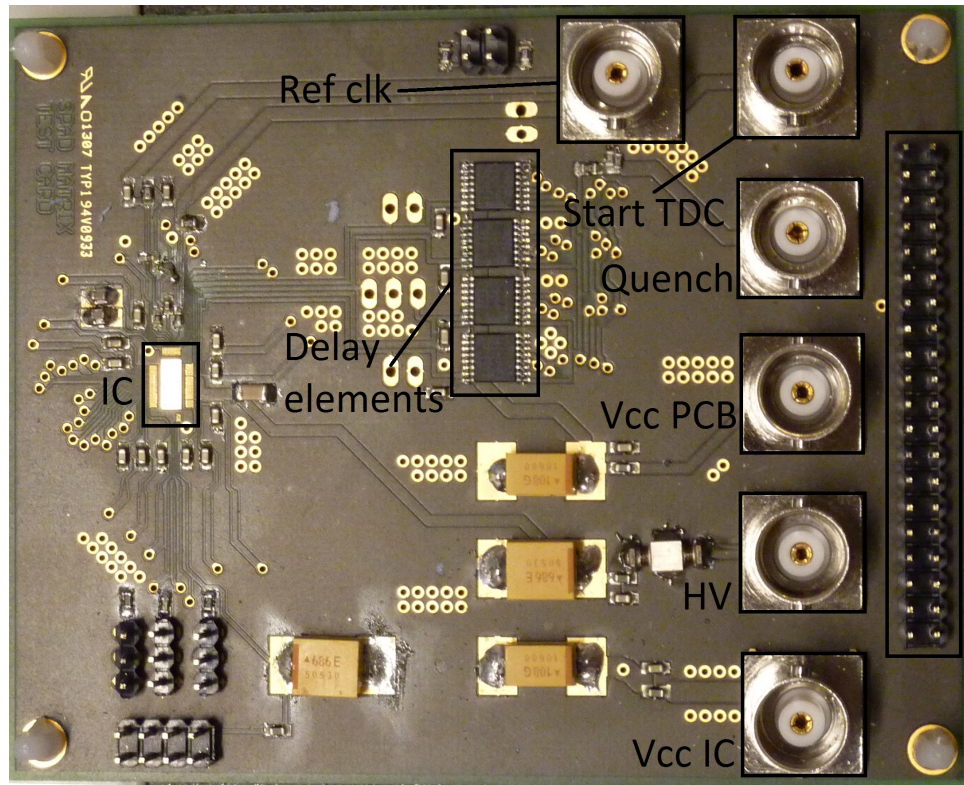


Figure 16. The component side of the test platform PCB.

The PCB has two operating voltage inputs as the PCB components and the SPAD matrix IC have different operating voltages. This arrangement is made in order to avoid the effect of the mutual interference. The high voltage enables the avalanche phenomena in SPADs.

The IC is placed in a cut sinking on the PCB and the pads of the IC are bonded to the wires around the sinking. Sinking the IC shortens the length of the bonding wires which reduces the parasitic inductance.

The quenching signal input is divided into four separate wires which are ladder-like wired to delay elements. The delay elements define the time intervals between the quenching signals introduced in Figure 3 on the page 11. The delay elements are programmable and have a delay range from 0 to 63,75 ns with 0,25 ns steps. The elements have a 8-bit latch memory, which is programmed from the FPGA. The programming interface is serial and includes a latch enable, a serial clock and data. As the latches are enabled, the data runs in at the rate of the serial clock. The delay elements also have a data output, which makes it possible to connect multiple ICs serially and use the same programming interface to all of them. [8] Thus, the FPGA interface includes three wires that program all the four delay elements. The signal outputs of the delay elements are ladder-like wired to the IC. The PCB has testpoints from where the quenching signals and the *start TDC* can be probed.

The FPGA interface is used to read data from the SPAD matrix IC to the FPGA and to program the delay elements. A jump wire can be added from the quenching input in order to tell the FPGA when the data can be read. The data interface is an asynchronous read-only memory interface, where the address defines the location of the data and it appears on the data bus. The three-wire programming interface of the delay elements is represented in the previous paragraph.

3.3.2. Message protocol

Similar to data flow of the design in the section 3.2, the computer sends instructions to the FPGA and the FPGA sends framed data packets to the computer. The instruction set of the FPGA logic is represented in Table 3. Each instruction has an instruction byte, which defines the instruction. Besides, the set instructions have data bytes that are added after the instruction byte. The functionalities of the instructions are detailed in the subsection 3.3.3.

Table 3. FPGA instruction set

Instruction	Hex	Class	Bytes	Description
INTERRUPT	0x90	Reset	1	Stop the measurement
MEAS	0xA0	Action	1	Start measurement
READ	0xB0	Action	1	Read data to the computer
PROG	0xC0	Action	1	Program the delay elements
SET MEAS	0xD0	Set	6	Set measurement ID and length
SET READ	0xE0	Set	3	Set the start address of the read
SET PROG	0xF0	Set	5	Set the delay values
RESET	0x80	Reset	1	Reset the logic

The data is organized in a frame represented in Figure 17. The frame is build above the USB protocol. The data word requires four bytes. Thus, the length of the preamble is five bytes which prevent the receiver from mixing the data between the preamble as the bit stuff byte differs from the preamble bytes. The header of the frame has a measurement identifier, which prevents from reading the same data twice. The next part of the header is the number of the quench marks during the measurement. It is represented in four bytes. The next two bytes declare the address of the FPGA RAM from where the read begins.

Data frame structure



Figure 17. The structure of the data frame.

As the size of the SPAD matrix is 128x4, each element of the matrix produces eight different codes, which are stored in different words, and each word consists of four bytes, the payload of a frame with all the data is 16384 bytes. When the header and the bit-stuff bytes are added to the payload, the entire frame has 20492 bytes. As the data is sent over the USB in packets independent of the frames, the length of the frame increases the risk of being cutted into two parts as the USB packet ends. In order to maintain the data flow from the FPGA to the computer, the size of the payload can be adjusted. This is done by setting the starting address of the read before the reading begins. If the frame is truncated during the transmission, the next read can begin from the address which is not yet received by the computer.

3.3.3. FPGA logic

The FPGA logic communicates with a USB converter, reads and stores data from the SPAD matrix IC and programs the delay elements of the test board. The logic is divided in functional blocks (Figure 18). Each external device (the USB converter, the SPAD matrix IC and the delay elements) communicates with the FPGA via an interfaces that have their own functional blocks. Besides, the FPGA has blocks that decodes the instructions from the computer, temporary stores the data and handles the measurement data at the reception and transmission. The top level functionality is controlled by a finite state machine.

The *USB interface* is the same block that is represented in the section 3.2.2 on the page 18. The *instruction decoder* is mutatis mutandis the same as in the section 3.2.2

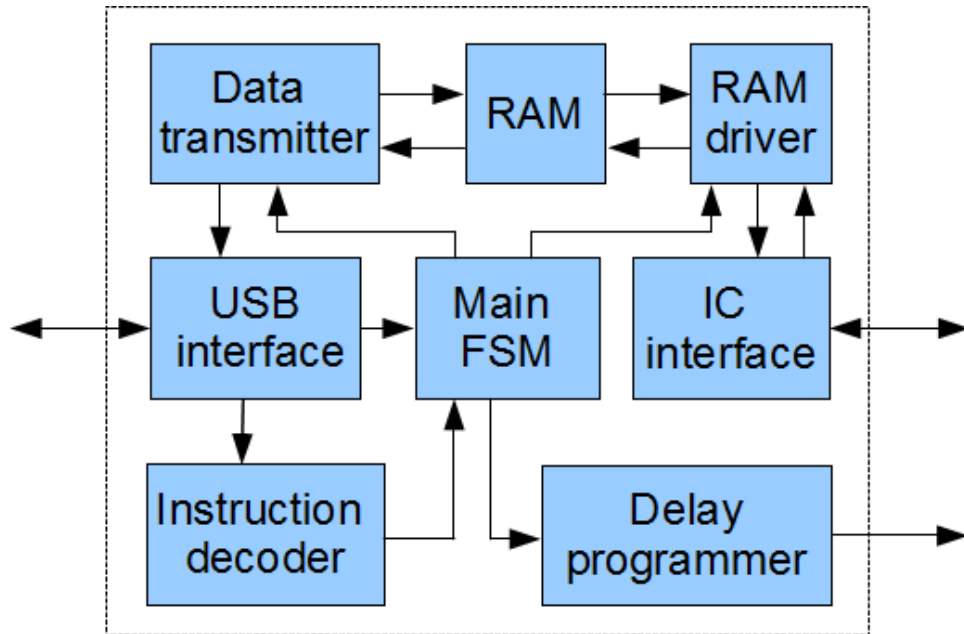


Figure 18. The functional block diagram of the FPGA logic.

as the instruction set of Table 3 on the page 23 is taken into account. There are more instructions to decode than in the previous design but the functionality is similar.

The *MFMS* controls the top level functionality of the logic according to the instructions represented in Table 3. The instruction set consists of two reset instructions, three action instructions and three set instructions. The *RESET* forces the logic into the idle state. The *INTERRUPT* interrupts the measurement without corrupting the data. It allows the logic to finish reading the data of the previous quenching mark. Each action instruction has a corresponding set instruction: the *MEAS* needs a byte-long identifier and four bytes to the number of quenching cycles, the *READ* needs two bytes to the address where to start reading and the *PROG* needs four byte-long delay values that will be programmed into the delay elements. The set parameters are placed in special registers which are led to the responsible blocks. The responsible blocks of the *READ* and the *PROG* actions are the *data transmitter* and the *delay programmer*, respectively. The *RAM driver* and the *IC interface* performs the action of the *MEAS* together. The action is started by a signal of the *MFMS* and the responsible block signifies when the action is taken.

The *delay programmer* is the FPGA interface to the delay elements of the test board. As the *MFMS* asks the *delay programmer* to run, it loads the delay values stored in the registers of the *MFMS*, enables the latches of the delay elements and sends the delay values serially to the delay elements. The serial clock is a divided FPGA system clock. As the programming is done, the *MFMS* is reported.

The *IC interface* and the *RAM driver* forms a complex that handles the FPGA interface to the SPAD matrix IC and the data writing into the *RAM*.

The *IC interface* consists of the TDC initializer at the power up and the data interface. The TDC initialization is done after the programming of the IC and after the asynchronous reset. The software reset transfers the *IC interface* to an idle state. As the TDC is initialized, the *IC interface* is able to read the data from the IC. The data

interface consists of a 7-bit address and 12-bit data bus. The data bus includes four 3-bit codewords that represent the TDC results of one matrix row of four elements. The read process covers all the 128 addresses. In order to begin reading, the *IC interface* needs a permission from the *RAM driver* and the quenching signal after the permission. Waiting for the quenching signal makes sure that the SPAD matrix does not rewrite the results during the readout. When the readout is done as many times as the parameter of the *SET MEAS* defines, the *IC interface* signals the *RAM interface*.

The *RAM driver* writes the *RAM*, receives the *MSFM* signal to start measurement and confirms to the *MFSM* when the measurement is done. The measurement data consists of the cumulative existence of different codewords during the measurement. Each SPAD element of the matrix has eight different options as the length of the codeword is 3 bits. Thus, there are 4096 memory locations in the *RAM*. The *RAM* is initialized by the *RAM driver* at the beginning of the measurement. The *IC interface* is permitted to read the results after the *RAM* initialization. During the measurement, the *RAM driver* receives a codeword from a specific location of the SPAD matrix. The address of the *RAM* is a combination of the address, from where the data is read, and the codeword. The memory content of the address is incremented. As the *IC interface* reads four codewords from each address, the *RAM driver* needs an access to four places of the *RAM* simultaneously. This is implemented dividing the *RAM* into four smaller *RAMs* which have their own write enable, address and data ports. It has to be noted that the *RAM* has both synchronous read and write, which enables the use of the special memory elements of the FPGA resources and saves the logic element resources to other use. Having a synchronous read in the *RAM* generates more latency to the writing process.

The *data transmitter* generates the data protocol of Figure 17 and feeds the frame byte by byte to the *USB interface*. The start signal comes from the *MFSM* and it is responded to when the transmission is complete. The *data transmitter* reads the data from the *RAM* starting from the address that is specified in the register of the *MFSM*.

3.3.4. Computer software

The computer software collects the data the IC generates. It is implemented with c++ in the Visual Studio 2010 environment as a Windows forms application and has a graphical user interface (GUI) represented in Figure 19. The software controls the measurement sending instructions to the FPGA via the USB. The relation between the software application and the Windows driver model is represented in Figure 10 on the page 17. The USB host controller driver handles the USB protocol and the application developer works with macros that make the exporting from the DLL simpler. Four black boxes in Figure 19 manage measurements and data, delay element setup, data representation and user notification.

The *Measurement options*, at the top left corner, are used to measurement and data management. The user can define the number of the quenching cycles the IC performs during the measurement in the *Number of measurements* textbox. The value is sent to the FPGA when the user presses the *Start*. The progress bar illustrates the progress of the measurement. The *Start* becomes *Stop* during the measurement which enables the interrupt of the measurement. The *Laser frequency* means the quenching frequency of the SPAD matrix and helps the program to estimate the duration of the measurement.

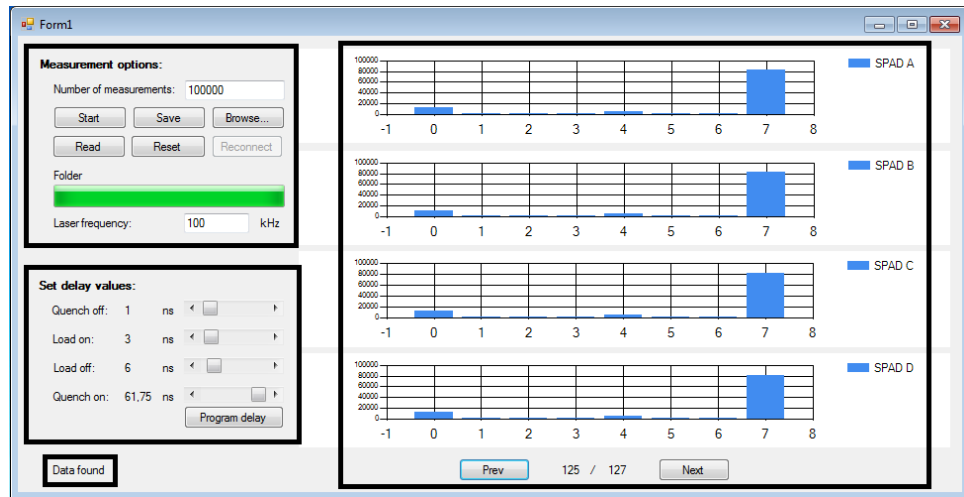


Figure 19. The graphical user interface of the software.

The *Read* button brings the data from the FPGA RAM to the computer. If the user wants to save the measurement results, the *Browse* opens a Windows folder browser to choose the destination folder and the *Save* stores the data into the destination folder. The folder path appears between the buttons and the progress bar. The data is saved in text files that are collected into a folder. The folder is created in the destination folder. If multiple measurements will be saved in the same destination folder, the software will generate different folder names to the consecutive measurements. If there appears any fail in the USB connection, the *Reconnect* button becomes active and enables reconnection. The FPGA logic can be reset by the *Reset* button.

The *Set delay values* on the left of Figure 19 manages the delay values of four delay elements on the PCB. The user can define the delay values setting the scroll bars. Pressing the *Program delay* button sends the delay values and programming instruction to the FPGA.

The right part of Figure 19 is reserved to data representation. As the data is read from the FPGA, it appears in the charts. The chart may need to be refreshed using any of two buttons under the charts before the data is visible. As well as there are four columns in the SPAD matrix, there are four corresponding charts representing the data. The numbers at the bottom signify the SPAD matrix row that is currently shown. *Prev* and *Next* buttons can be used for browsing the rows. Each chart has eight bars as there are eight different codewords in the TDC. The chart represents the codeword distribution of an element. For example, the current view could be measured in weak light condition where the circuit is quenched 100000 times and most of the cycles do not trigger.

The black box at the left bottom corner is a message box. It is used to notify the user about the program condition. For example, it tells if the measurement is running or ready, if the data is correctly received, or if there is a specific problem in the USB connection.

4. MEASUREMENTS

This section is about measurements that were made with the SPAD matrixes. As there were two different SPAD matrixes in research the measurements are represented in two sections here. The first section is about time gated measurements and the latter section represents time interval measurements.

The aims of the measurements were to verify the designed functionalities of the matrixes, evaluate the quality of SPADs and study the timing homogeneity of the control signals. The functionalities to be verified are detailed in sections 4.1 and 4.2 as the operation principle differs between the two SPAD matrixes. The quality evaluation included the measure of the noise level in the SPADs which factor is called dark count rate (DCR). In order to observe delay differences in signal propagation in an IC smaller than 1 cm^2 special measuring setups and large number of measurements were needed. The measuring setups depended on the IC functionalities and are detailed later in corresponding sections 4.1.2 and 4.2.

The measurements were divided into two parts. The first part consisted of DCR measurements where the matrixes were set in active mode several times without any external light source. The detector IC was in a darkness and that is why these measurements are called dark count measurements. The counted hits were due to the thermal and tunnel noise due to the excess bias [9, 2]. The DCR measurements were done only with the SPAD matrix with time gating, that is represented in the section 2.3. The second part of the measurements included laser excitation where the laser pulses at 100 kHz frequency were aimed at the matrix. The timing of the laser excitation was related to the activation and the quenching moments of the SPAD matrix.

4.1. Time-gated measurements

The time gated 128x8 SPAD matrix IC, that was represented in the section 2.3, was used in the time-gated measurements. The IC counts the number of breakdowns over a predefined number of quench cycles. The breakdown can occur once a quench cycle for a SPAD. Each SPAD of 128x2 matrix has four time gates whose values were adjustable. The sizes of the gates were assumed to influence directly the hit counts when the photon detection probability is low.

The quality and the homogeneity of the signal timing were studied in the measurements. The quality of the SPAD matrix was evaluated in the dark count measurements which are represented in the section 4.1.1. When the homogeneity of the signal timing was under the research, the laser pulses were timed relative to the load and the quenching signals of the SPAD matrix. There were a set of measurements where the laser pulses were swept over the active time of the SPADs with 5 - 10 picosecond increments. The laser was pulsed 50000 times at each step.

4.1.1. Dark count measurement

The dark count measurements were done using two different gating windows. The measurement setups are detailed in Table 4. *Load* is the same as *precharge* earlier in

this document. All the gate windows began after the load and the ended as the time windows of Table 4 were over. Thus, the shorter time windows were included in the longer. The gating windows were measured outside the IC using an oscilloscope.

Table 4. DCR measurement setups.

	DCR 1	DCR 2
Load	1100 ps	1100 ps
Gate 1	700 ps	9000 ps
Gate 2	900 ps	9200 ps
Gate 3	1000 ps	9300 ps
Gate 4	1500 ps	9800 ps
Cycles	5 000 000	10 000 000

As mentioned earlier in the chapter 4, the DCR depends on the temperature and the excess bias. Besides, each SPAD has its individual DCR character when a SPAD matrix forms a distribution [10]. This feature was found experimentally and is illustrated in Figure 20.

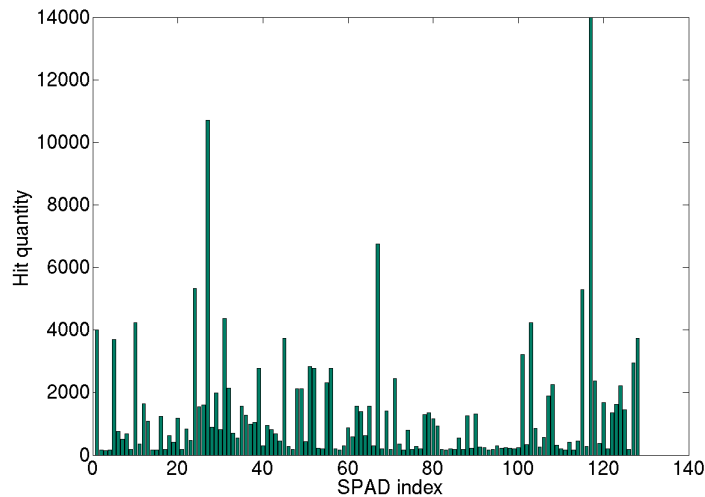


Figure 20. Dark count hit distribution over the SPAD matrix in DCR1 measurement at the gate four.

A factor that affects the number of dark count hits is the active time window size. The active time window can be approximated as the sum of the load and the gate window from Table 4. Experimental reference can be seen in Figure 21 where a gate-wise mean dark count hits are represented for both measurements. The gate-wise mean refers to the mean over 128x2 SPAD matrix for each time gate separately.

4.1.2. Laser pulsed measurement

The overall measurement setup of the laser pulsed measurements is described at the beginning of the chapter 4 and the section 4.1. The values of the load and the gates were

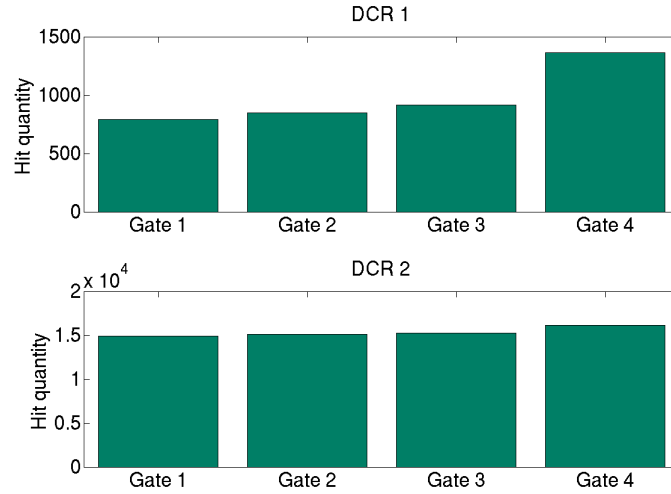


Figure 21. The mean dark count hit at four gates in the DCR1 and DCR2 measurements.

the same as in the first DCR measurement in Table 4. The laser beam was attenuated by placing a dark glass between the laser and the IC in order to avoid saturation. The laser timing began from the active region of the gates and were incrementally delayed until all the gates were quenched. [11]

Figure 22 is a three-dimensional description of the measurement results. The x-axis is the delay, the y-axis is a row from the SPAD matrix and the z-axis depicts the number of the hits. The profile of the laser intensity can be seen as a cross-section along the y-axis. A steep downward trend of the surface at the end of the delay depicts the quenching of the SPAD matrix.

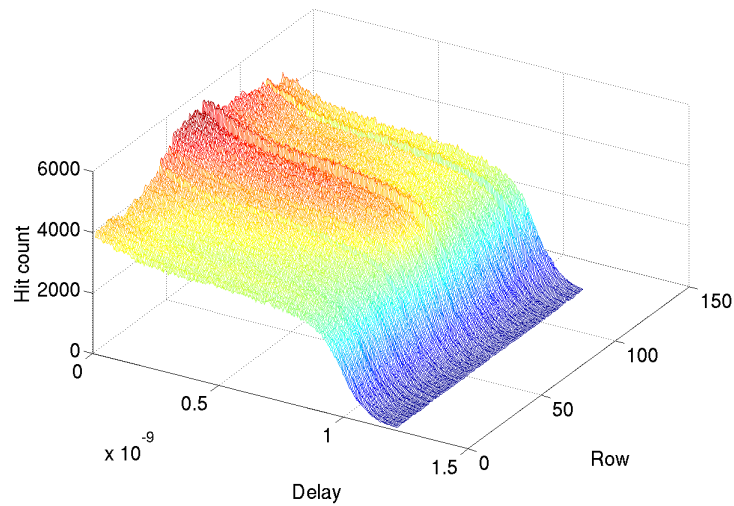


Figure 22. Laser pulsed measurement. The laser was incrementally delayed until the gate was quenched.

The purpose of the measurement was to find out the homogeneity of the signal timing and, especially, how large signal delay differences are between the elements of

the SPAD matrix. The reference was the moment when the laser triggered hits were reduced to a half. Achieving the result needed a processing of the data. At first, the surface of the waterfall in Figure 22 was cut just above the steep fall removing the affect of the side pulse of the laser. The estimated DCR hits were subtracted and the measured values of single SPADs were normalized in order to find the half height.

Figure 23 illustrates how the arriving time of the signals varies along the SPAD matrix. The x-axis describes 128 elements of the SPAD matrix and the y-axis is the delay of the laser compared with the beginning of the measurement. The laser pulse was timed inside four gates at the beginning of the measurement. The angular shape of the plot is due to the 10 ps step shifts of the laser. All the gates had 60 ps homogeneity at the worst.

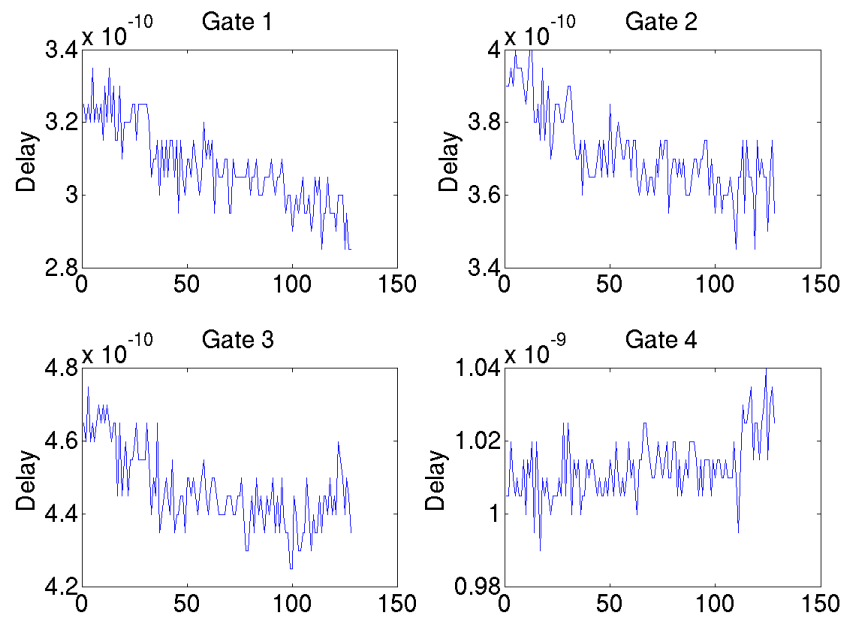


Figure 23. The moment when the counted hits were decreased to a half compared to the hits when the whole laser pulse was inside the gate.

4.2. Time interval measurements

The IC represented in the section 2.4 on the page 10 has a SPAD matrix and a TDC. The SPAD matrix was quenched according to the timing in Figure 1 on the page 8.

4.2.1. Laser pulsed measurements

As the TDC works independent of the quenching, it needed a separate *start TDC* signal. The signals were time so that the *precharge* occurs at the beginning of the *spadoff* and the *start TDC* arrived after the *precharge*. The measurements included laser excitation, which was timed related to the *start TDC*. The *start TDC* was lead through a delay pipe

whose length was changed by 2,5 mm steps which corresponds to a delay of 8,3 ps at the speed of the light. The TDC measured the time interval between the *start TDC* and the breakdown of the SPAD. Depending on the measured time interval, the TDC generated a codeword according to Table 1 on the page 12. The codewords were collected a million times at each delay setup.

Figure 24 shows an example of the hit distribution during the measurement. The x-axis depicts the delay generated by the delay pipe. At each x-axis value, the sum of all the codewords is a million. The cumulation of a certain codeword is represented at the y-axis. As the delay changes, the laser excitation moves from a time window to another according to the codes. The transition between two time windows is gentle which indicates an inaccuracy in the measurements. The measurement model is explained in the section 5.2.

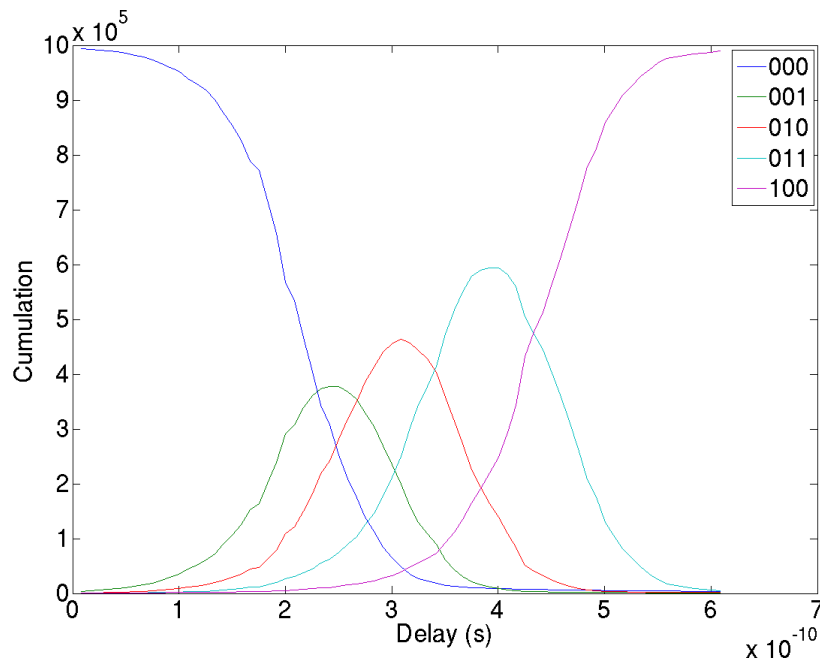


Figure 24. The hit distribution during the laser excitation measurement.

As the data was postprocessed in Matlab, a Gaussian curve was fitted to each TDC code cumulation, like presented in Figure 24, in each element of the SPAD matrix. Gaussian curves have three parameters that are the location, the height and the width. In Figure 24, the top most parts of three curves in the middle depict the locations of the fitting Gaussian curves. Three tops also represent an approximation of the center of time windows. When the fitting Gaussian curves were studied over the SPAD matrix, it was noted that the location of the curves varies between corresponding curves. Figure 25 shows the locations of the curves fitting the TDC code '001' over the matrix as four columns are marked as letters from A to D and the rows represented in the x-axis. The location is referred to the delay that is generated by the delay pipe and showed in y-axis.

The green sink at the row 87 is due to the unexpected behaviour of a single element. If the sink is not taken account, all the locations fit inside a 180 ps time window.

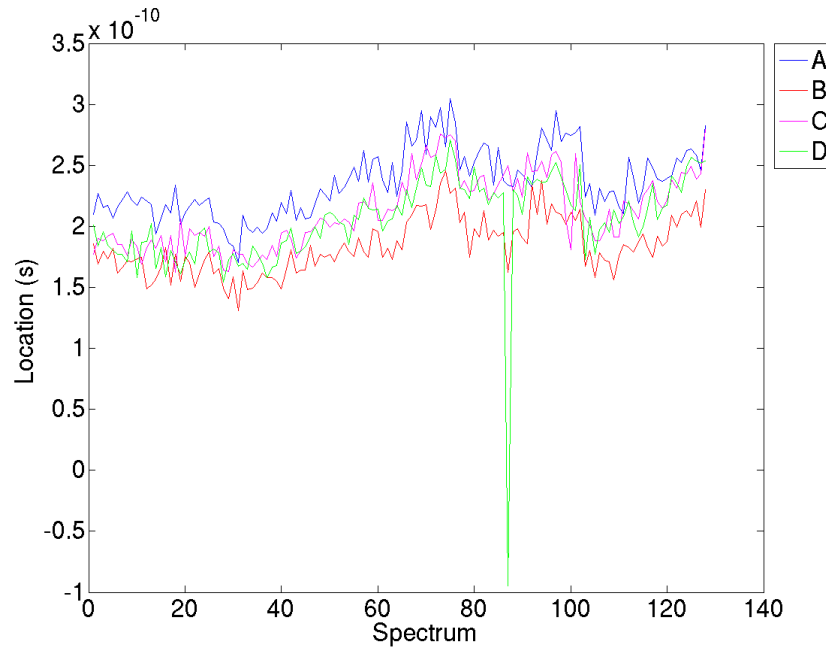


Figure 25. The TDC code '001' spike location.

4.2.2. Crosstalk measurement

As a SPAD breakdowns, it emits a bunch of photons that may launch a breakdown in adjacent SPAD elements. This phenomenon is called optical crosstalk. This subsection represents a measurement that studies the characteristics of the optical crosstalk in the SPAD matrix. The challenge of the measurement is how to determine if a breakdown is caused by the breakdown of the adjacent SPAD element.

The measurement principle was to set a single noisy SPAD element as an emitter and the adjacent SPAD elements were considered as detectors. The measurement was done at the conditions of the DCR measurements that made sure no external photons were absorbed in the matrix. The TDC reference clock was set to 50 MHz which broadened the time windows as the time window size is directly proportional to the reference clock period.

The FPGA logic represented in the section 3.3.3 on the page 24 collects the TDC codes in a statistical way which fades the results of an individual measure and hence the crosstalk information cannot be extracted from the data afterwards. The reader logic was redesigned to temporally store the data of the individual measure. As the readout was complete, the temporal data was compared with the codeword of the emitter SPAD. If the emitter SPAD had the code '001' and the adjacent elements had the same or slightly bigger code, a crosstalk hit would be recorded. The crosstalk information was stored both element and TDC code wise in order to maintain the locational and temporal information.

As each SPAD matrix element has its own DCR, it is statistically expected the emitter and the detectors trigger a couple of times at the same time window without any dependence. In order to estimate the level of the crosstalk, the statistically expected part has to be removed from the measurement results, as it is done in Figure 26. Figure

26 shows a 7x4-element part from the complete matrix at the time window '001'. The emitter element locates between the two highest bars and its value is removed because of the clarity. The crosstalk is represented as the trigger ratio between the emitter and the detector. The noisy element was triggered more than a million times.

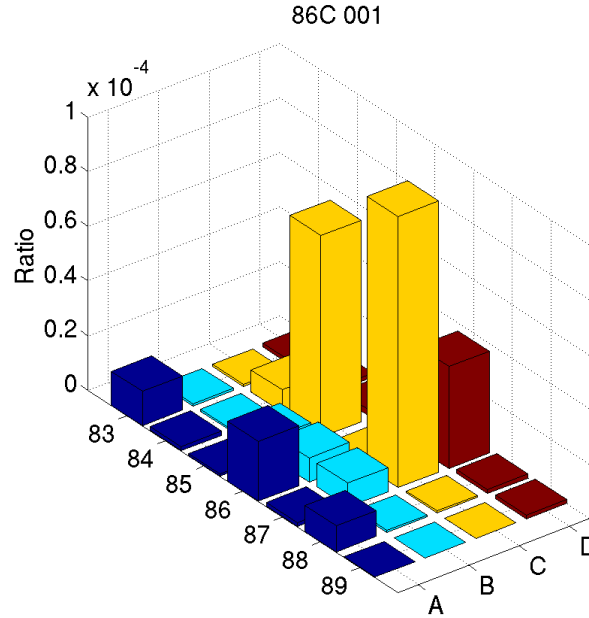


Figure 26. The spatial distribution of the crosstalk.

As it can be seen from Figure 26, the crosstalk is stronger in the spectral direction compared with the spatial direction. This is expected phenomena due to the arrangement of the matrix layout where the gap between the columns is wider than between the rows.

The TDC given temporal information showed the crosstalk was five times stronger in the second window and three times stronger in the third window compared with the first window on average. This gives a sign the crosstalk has not only spatial but also temporal profile.

5. MODELLING THE MEASUREMENTS

5.1. Discrete convolution and deconvolution

The discrete convolution is defined as the equation 1 where the $x[k]$ is the stimulus and the $h[k]$ is the impulse response.

$$y[j] = \sum_{k=-\infty}^{\infty} x[k]h[j - k] \quad (1)$$

When the stimulus and the impulse response have finite lengths n and m , respectively, and are otherwise zero, the discrete convolution can be written in the equation 2.

$$y[j] = \sum_{k=0}^{n+m-2} x[k]h[j - k] \quad (2)$$

The sum of the convolution between the stimulus $x[k]$ and a unit height rectangle impulse response is written in the equation 3, which is proved in the Appendix 6.

$$S = \sum_{j=0}^{n+m-2} y[j] = m \sum_{k=0}^{n-1} x[k] \quad (3)$$

5.2. Measurement model

The section 4.2.1 represents measurements where the time interval between the *start TDC* and the laser trigger was increased by small steps. The TDC measures the time interval between the *start TDC* and the SPAD avalanche. The measured time interval is represented in 3-bit words according to Table 1 on the page 12. A single laser pulse was measured a million times at each delay setup.

The measurement model represents a way to assume the time window sizes of the TDC according to the measurement results. In more detail, the window sizes are directly proportional to the number of the corresponding TDC code words occurring during the whole delay range of the measurement. The rest of this section validates the model.

There are some factors that cause uncertainty to the measurements. When the laser is triggered a million times without any changes in the delay, the distribution of the output codes of the TDC can consist of not only one code but also two or more codes as it is shown in Figure 24 on the page 32. Especially, several code words are given when the delay between the *start TDC* and the laser trigger is close to the range where the narrow TDC time windows locate.

The dominating uncertainty factors are the shape of the laser pulse, the jitter between the laser excitation and the *start TDC* and the jitter of the SPAD. The TDC has also limited accuracy but its uncertainty is minor to these other factors. As the SPAD has a certain photon detection probability, the probability of the avalanche breakdown increases as the number of the photons grows. The shape of the laser pulse depicts the

distribution of the photons at the function of time. If the number of photons is low, the shape of the laser becomes the dominant factor of the uncertainty as it takes longer before the SPAD triggers. The jitter between the laser excitation and the *start TDC* is a feature of the used measurement equipment that limits the accuracy of a single measurement. The jitter of the SPAD depends on the physical size of the detector and the reverse bias. The leading edge of the current signal changes depending on the location of the depletion layer where the photon is absorbed. [12]

As mentioned, the moment of the SPAD triggering changes even though the delay setup stays similar. The triggerings occur a fixed delay after the *start TDC* - considered as mean - and, on the other hand, changes depending on the uncertainty factors mentioned in the previous paragraph - considered as deviation. As the laser is triggered a sufficient number of times, the triggerings forms a distribution at the function of time as it is demonstrated in Figure 27. The beginning of the delay and the window 000 corresponds to the moment of the *start TDC*. When the passive delay is incremented by a step, the average of the distribution moves further from the *start TDC* a corresponding time.

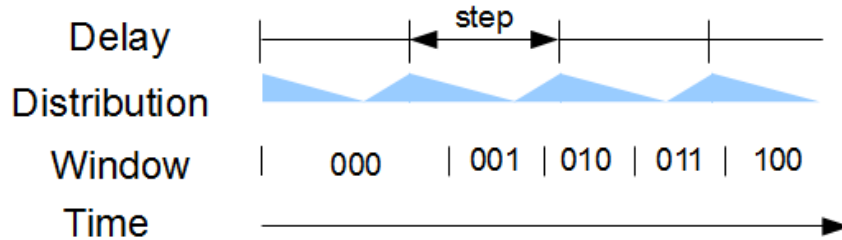


Figure 27. Demonstration of the measurement principle.

At a specific delay setup, the number of particular code words depends on the triggerings that occur inside the particular time window. As the delay is incremented, the number of the code words changes because the distribution moves. Predicting the cumulation of the code words on a specific delay setup needs a model where the distribution is a function of time and integrated from the beginning to the end of the time window.

A discrete model of the measurements is represented in Figure 28, where a single time window is shown unlike Figure 27. The triggering distribution is sampled to a vector of elements. Now, the number of code words can be calculated as a sum of the vector elements that locate inside the time window. As the window is modelled a vector with the same sampling rate as the distribution has, the element value is one inside and zero outside the time window. Hence, the code word cumulation of a time window can be calculated as a sum of the element wise multiplication between these two vectors. When each discrete delay value generates a sum, the measurement range forms a vector.

As the sampling interval of the time window and the distribution is marked as k and the delay step j , the measurement can be modelled by the discrete convolution represented in the section 5.1. At each delay setup, the matrix is pulsed by a fixed times which refers to the sum of the stimulus in the equation 3. When the m is solved from the equation 3, the time window width is multiplication between the m and the

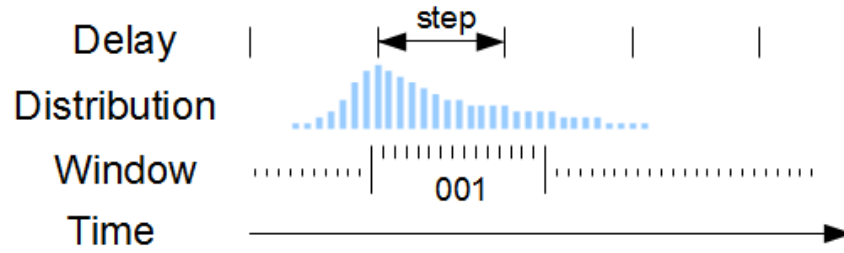


Figure 28. Discrete model of the measurement principle.

time delay interval between two consecutive delay setups. The model gives proper results to those time windows that are fully convolved.

5.3. Model accuracy

This section advises the accuracy of the estimated TDC window sizes that are given from the model of the previous section. The conclusion of the model accuracy is that the larger the statistical uncertainty factors of the stimulus are the more accurate estimations of the average TDC window sizes are. The average is used here as the lengths of the TDC time windows are not temporally constant.

When the model uses the data from the measurements, the accuracy of the measurements reflects in the accuracy of the model. When all the uncertainty factors mentioned in the section 5.2 are lumped together, they form a distribution which has a deviation. As the deviation has several independent components whose origins differs from each other, the model of the deviation is unknown in principle, but it is considered statistical. As a statistical distribution, increasing the number of measurements makes the shape of the distribution more stable and increases the repeatability.

The delay steps of the measurements represented in the section 4.2.1 were adjusted handheld. The adjustment followed a scale that is in the delay element. The accuracy of the handheld adjustment is a millimetre not only between two consecutive delay setups but also between two arbitrary delay values during the measurement as the scale was used as a reference. Thus, the error can be locally significant but reduced when the complete measurement range is observed.

In order to estimate the accuracy of the convolution, let us divide the triggering distribution and the time window into blocks, whose lengths are the time interval of a single delay step, as it is done in Figure 29. The time window consists of m full length blocks (blue in Figure 29) and a residual block (red in Figure 29) at the end of the window. During the convolution, the triggering distribution is moved step by over the time window and these two factors to be convolved are focused on the delay steps which are pictured black vertical lines in Figure 29.

When looking at a single blue block of the time window in Figure 29, the whole distribution goes through it during the convolution. Hence, the sum of the triggerings accumulated in the single blue block equals the sum of the triggerings in the whole distribution. The sum of the distribution answers to a delay length of a single delay step as mentioned at the end of the previous section. If the delay steps are constant

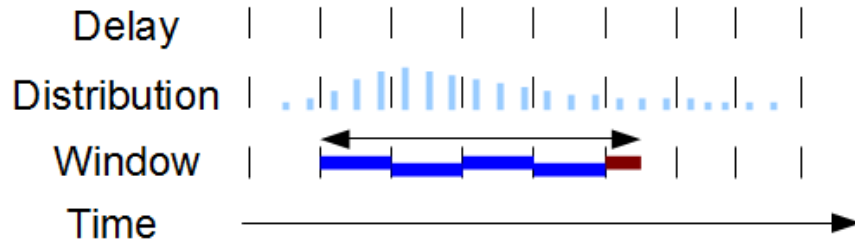


Figure 29. The distribution and the time window are divided into a delay step length blocks.

over the measurement and the distribution is constant, delay-step-length components of the time window affect no error to the window size assumption.

The red block of the time window represented in Figure 29 does not cover the whole distribution during the convolution. The convolution between the residual part of the time window and the triggering distribution can be illustrated as a sampling over the distribution, as it is shown in Figure 30.

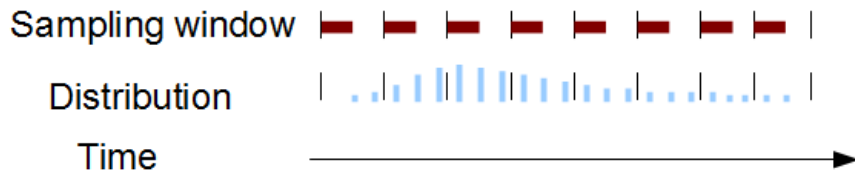


Figure 30. Sampling the distribution with a residual part of the time window.

As the distribution is convolved by the residual block of the time window, the sum of the convolution is smaller than the sum of the distribution. The relation between these two sums is assumed to answer the relation between the residual block (red) and a single whole block (blue) of Figure 29. The error, that is included in the window size assumption, forms from the mismatching of these two relations. The accuracy of the window size suffers from the steep shapes in the distribution. The error can vary also depending on the moment of sampling. On the other hand, a wide distribution compared with the delay step makes this error small and minor to the measurement inaccuracy.

5.4. Simulation

Figure 24 on the page 32 shows the result of the delay stepped laser measurement. This section describes a simulation that models that measurement and especially the simulation of the three small time windows at the middle of the measurement range.

The measurement used a Hamamatsu laser whose profile is presented in Figure 31. The SPAD element was assumed to collect 50 photons in total distributed over time according to the laser profile. The photon detection propability was set to 10%. The statistical fluctuation between the timing mark and the laser excitation, also called timing jitter and caused by the measurement tools, were set to have 70 ps standard deviation. The SPAD jitter caused by the dimensions of the SPAD element were set

to 40 ps standard deviation. Both jitter sources were assumed normal distributed. The normal distributions were skewed in order to resemble the SPAD jitter profile represented in the reference [13]. Different distributions were convolved in order to form a SPAD triggering distribution that takes different statistical inaccuracy elements into account.

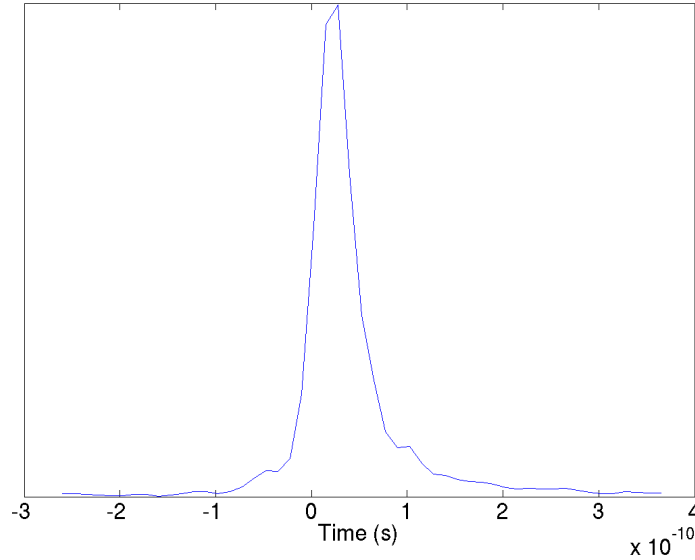


Figure 31. The profile of the Hamamatsu laser output curve.

As the delay pipe was adjusted handheld during the measurement, the delay accuracy is limited. The simulation uses a 2,5 mm delay step which has 1 mm uniformly distributed inaccuracy. The element was simulated a million times at each delay setup.

The time windows were calculated from the measurement represented in the section 4.2.1 exploiting the linear connection between the time window size and the hit cumulation during the whole measurement. The three small windows had sizes 55,9 ps, 69,9 ps and 97,9 ps. The simulation results are represented in Figure 32.

As the window sizes were calculated from the simulation results using the linear connection between the window sizes and the hits, the deviation from the input values were at the magnitude of per mil. Further research showed that the window size calculation got more accurate, when the delay steps were more accurate, the measurement range was widened and the repetition was increased.

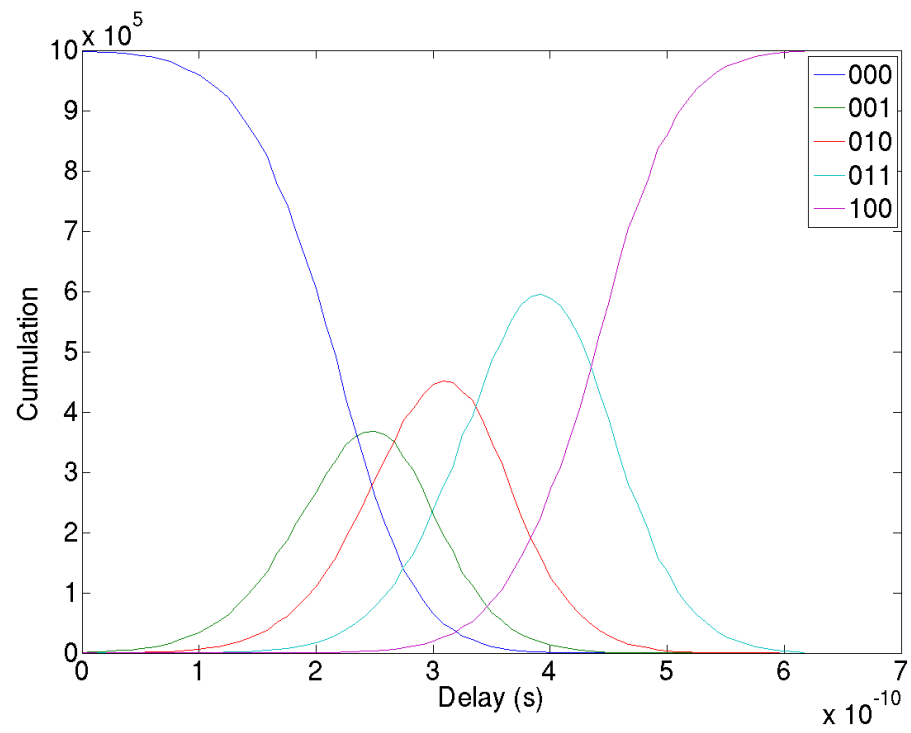


Figure 32. The simulation of the laser excitation measurement.

6. DISCUSSION

This chapter is a review and an evaluation of the work. As the original aim of this work was to build up a test environment for the IC represented in the section 2.3 and to verify its functionality performing some measurements, the topic has got wider. The final work consists of test environments for two different ICs, measurements for both ICs and modelling some measurements.

The performance of the test environment presented in the section 3.2 met mostly the expectations that were specified. The test environment was able to perform measurements, represent the data and store the data in the computer. At the very low number of quenching cycles, the repetition rate of 100 kHz was met. In higher volumes, the speed suffered from the serial reading interface. The IC was designed to support measurements during the readout, but the 3-bit wide counter inside the IC limits the number of hits that can occur during 250 μ s readout. This limits the power of the light source as any SPAD could trigger 7 times at maximum from 25 possibilities as the repetition rate was 100 kHz. Another issue was the reliability of the reading. The IC gave confusing results many times when the measurements and readout were performed simultaneously. Due to the timetable of the project, this issue was solved separating the measurements and the readout, which limited the repetition rate to a few tens of kilohertz at maximum. Developing the reset could have been an alternative way to face the issue.

The other test environment, represented in the section 3.3, fulfilled its specifications. The parallel data interface was fast and flexible. The FPGA was able to support the repetition rate up to 390 kHz. During the crosstalk measurements, the repetition rate could have been even higher as only a part of the SPAD matrix were read. The graphical user interface made the representation of the measurement results more visual and controlling the measurements more intuitive.

Technically, the test environments were practical and they met their purpose. For future developing, there are some improvements that can be made. At first, the interface between the FPGA and the USB converter could be more controlled, especially when the data runs up to the computer. This improvement makes the link between the computer and the FPGA more reliable. Besides, if there is any digital interface to the measurement tools, the measurements can be made more computer driven.

As the measurements prove the functionality of the test environments, the main purpose of the measurements were to find out the characteristics of the SPAD matrix and how it would work as a receiver of the Raman spectroscopy. It is intuitive, that wide spatial matrix of small elements can distinguish different wavelengths from a beam that is diffracted. However, according to the laser pulsed measurements described in the chapter 4, the timing between the laser excitation and the SPAD matrix response varies over the matrix. Besides, reference measurements showed that the SPAD matrix response slightly varies between consecutive measurements.

As the Raman scattering has a different temporal profile from the fluorescence [14] and the fluorescence of the Raman sample has to be minimized, the knowledge of the dependence between the laser excitation and the SPAD matrix response is important. The physical structure of the SPAD matrix and its wiring has an affect to the response. The influence of the device temperature cannot be excluded as the SPAD matrix response varies between independent measurements. The fluctuation of the

SPAD matrix response can be compensated using calibrating measurements if there are data processing resources available.

A part of this work was devoted to the modelling of the measurements. The mathematical model deepened the understanding about the inaccuracy factors of the measurements and how they appeared in the results. The convolution offered a simple and powerful tool for estimating the time window sizes of the TDC.

7. CONCLUSION

Two test environments for two different SPAD matrix ICs were designed and implemented in this work. The work included measurements that verified the functionalities of the test environment. The measurements also studied the characteristics of the SPAD matrixes as a receiver of a Raman spectrometer.

The test environments were divided in four separate parts which were a PCB, an FPGA development board, a USB converter and a computer software. The SPAD matrix IC were placed on a PCB that offered a physical and electrical platform. An FPGA acted as a temporal memory for the data and an interface between the IC and the USB converter. The USB converter was an interface between the FPGA and the computer. The computer software was used to set up the measurements and collect the data.

The measurements were performed in both dark and laser excited conditions in order to find out the delay differences over the matrix and the time window profile of a single element. The delay difference over the matrix was inside 60 ps window in the first matrix and 180 ps window in the second matrix. It has to be noted that the results between the two SPAD matrixes are not directly compatible as the method of defining the delay difference was different and the size of the matrixes were different. Another notice is that the sizes of three small TDC time windows differ from the specification and from each other dozens of percentages.

In future, the measurement results can be corrected using the knowledge of previous measurements if there are computational resources available. The computational correction can be done for the systematic error that is found.

8. REFERENCES

- [1] Field R.M., Lary J., Cohn J., Paninski L. & Shepard K.L. (2010) A low-noise, single-photon avalanche diode in standard 0.13 μm complementary metal-oxide-semiconductor process. *Applied Physics Letters* 97, Article 211111, pp. 1–3.
- [2] Cova S., Ghioni M., Lacaita A., Samori C. & Zappa F. (1996) Avalanche photodiodes and quenching circuits for single-photon detection. *Applied Optics* 35, pp. 1956–1976.
- [3] Lämsmäki A. (2011) Aikaportitettu 128 x 8 pikselin SPAD-matriisi Raman-spektrometriin. Diplomityö, Oulun yliopisto, sähkötekniikan osasto, Oulu.
- [4] Nissinen I., Nissinen J. & Kostamovaara J. (2013) A time-gated 4x128 SPAD array with a 512 channel flash 80 ps-TDC for pulsed Raman spectroscopy. 21st European Conference on Circuit Theory and Design (ECCTD).
- [5] Nissinen I., SPAD matrix interface. Internal report, Oulu University, Electronics Laboratory (2013) .
- [6] FT232H Single Channel Hi-Speed USB to Multipurpose UART/FIFO IC. Datasheet, Future Technology Devices International Ltd. (2011) .
- [7] Software Application Development. D2XX Programmer's Guide, Future Technology Devices International Ltd. (2012) .
- [8] 3.3V, 8-Bit, Programmable Timing Element. Datasheet, Maxim Integrated Products (2007) .
- [9] Veerappan C., Richardson J., Walker R., Li D.U., Fishburn M.W., Stoppa D., Borghetti F., Maruyama Y., Gersbach M., Henderson R.K., Bruschini C. & Charbon E. (2011) Characterization of Large-Scale Non-Uniformities in a 20k TDC/SPAD Array Integrated in a 130nm CMOS Process. In: Solid-State Device Research Conference (ESSDERC), pp. 331–334.
- [10] Karami M.A., Niclass C. & Charbon E. (2009) Random Telegraph Signal in Single-Photon Avalanche Diodes. In: International Image Sensor Workshop, Bergen, Norway.
- [11] Nissinen I., Lämsmäki A.K., Nissinen J., Holma J. & Kostamovaara J. (2013) 2x(4x)128 time-gated cmos single photon avalanche diode line detector with 100 ps resolution for raman spectroscopy. In: ESSCIRC, pp. 291–294.
- [12] Lacaita A. & Mastrapasqua M. (1990) Strong dependence of time resolution on detector diameter in single photon avalanche diodes. *Electronic Letters* 26, pp. 2053–2054.
- [13] Mazzillo M., Condorelli G., Campisi A., Sciacca E., Belluso M., Billotta S., Sanfilippo D., Fallica G., Cosentino L., Finocchiaro P., Musumeci F., Privitera S., Tudisco S., Lombardo S., Rimini E. & Bonanno G. (2007) Single photon avalanche photodiodes arrays. *Sensors and Actuators A: Physical* 138, pp. 306 – 312.

- [14] Martyshkin D.V., Ahuja R.C., Kudriavtsev A. & Mirov S.B. (2004) Effective suppression of fluorescence light in Raman measurements using ultrafast time gated charge coupled device camera. *Review of Scientific Instruments* 75, pp. 630–635.

LIST OF APPENDICES

- APPENDIX 1: Modules of the FPGA logic (Design 1)
- APPENDIX 2: The state transition diagram of the finite state machine (Design 1)
- APPENDIX 3: The schematic of the test card (Design 2)
- APPENDIX 4: The layout of the test card (Design 2)
- APPENDIX 5: The component placing of the test card (Design 2)
- APPENDIX 6: Proofing

APPENDIX 1. MODULES OF THE FPGA LOGIC (DESIGN 1)

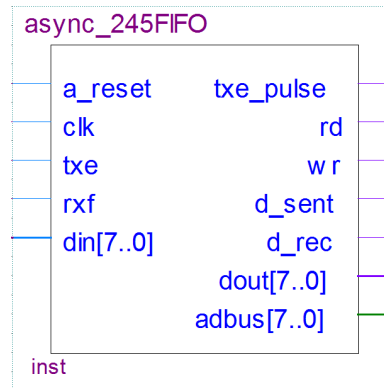


Figure 1. USB interface.

Table 1. Ports of the USB interface.

Port	Direction	From/To	Purpose
txe	Input	USB converter	Write operation invitation handshake
rx	Input	USB converter	Read operation invitation handshake
din[7..0]	Input	Data transmitter	The bus of the data to be transmitted
tx_pulse	Output		Test Port
rd	Output	USB converter	Read operation confirmation handshake
wr	Output	USB converter	Write operation confirmation handshake
d_sent	Output	Data transmitter	Transmission done indicator
d_rec	Output	Multiple	Reception done indicator
dout[7..0]	Output	Multiple	The bus of the received data
adb[7..0]	Inout	USB converter	Bidirectional data bus to the USB converter

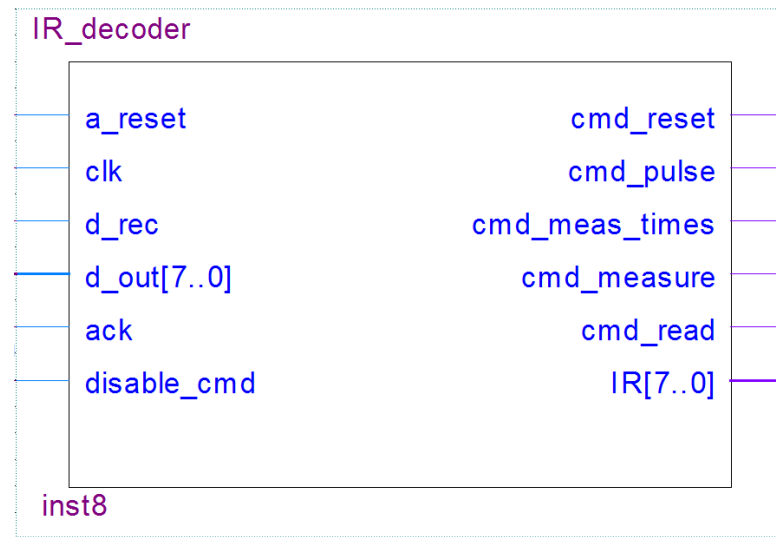


Figure 2. Instruction decoder.

Table 2. Ports of the instruction decoder.

Port	Direction	From/To	Purpose
d_rec	Input	USB interface	Reception done indicator
dout[7..0]	Input	USB interface	The bus of the receipted data for instruction decoding
ack	Input	Finite state machine	Acknowledgement of an instruction
disable_cmd	Input	Finite state machine	Rejects to decode receipted bytes
cmd_reset	Output	Finite state machine	Decoded reset instruction
cmd_pulse	Output	Finite state machine	Decoded SPAD matrix counter limit set instruction
cmd_meas_times	Output	Finite state machine	Decoded SPAD matrix read cycle limit set instruction
cmd_measure	Output	Finite state machine	Decoded start measurement instruction
cmd_read	Output	Finite state machine	Decoded read data to computer instruction
IR[7..0]	Output		Test Port

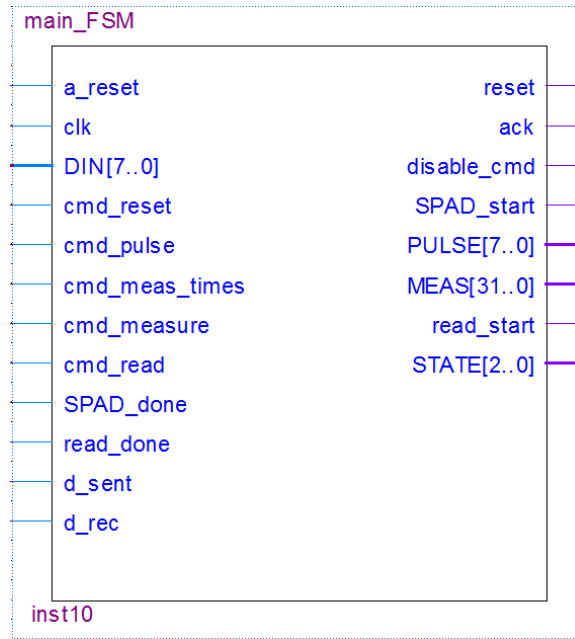


Figure 3. Finite state machine.

Table 3: Ports of the finite state machine.

Port	Direction	From/To	Description
DIN[7..0]	Input	USB interface	The received data for set instructions
cmd_reset	Input	Multiple	Reset instruction
cmd_pulse	Input	Finite state machine	SPAD matrix counter limit set instruction
cmd_meas_times	Input	Finite state machine	SPAD matrix read cycle limit set instruction
cmd_measure	Input	Finite state machine	Start measurement instruction
cmd_read	Input	Finite state machine	Read data to computer instruction
SPAD_done	Input	IC interface	The indicator of successfully done measurement
read_done	Input	Data transmitter	The indicator of successfully done read operation
d_sent	Input	USB interface	TestPort
d_rec	Input	USB interface	A byte from the USB received
reset	Output	Multiple	Resets the running measurement or read process
ack	Output	Instruction decoder	Confirms an instruction
disable_cmd	Output	Instruction decoder	Disables the instruction decoding while the set instruction data is received
SPAD_start	Output	IC interface	Start measurement indicator
PULSE[7..0]	Output	IC interface	The number of the pulses the IC counts before it is read
MEAS[31..0]	Output	IC interface	The number of IC read cycles

Table 3: Ports of the finite state machine. (continued)

Port	Direction	From/To	Description
read_start	Output	Data transmitter	in a measurement Start transmission to the computer indicator
STATE[2..0]	Output		Test Port
disable_cmd	Output	Finite state machine	Rejects to decode recepthed bytes

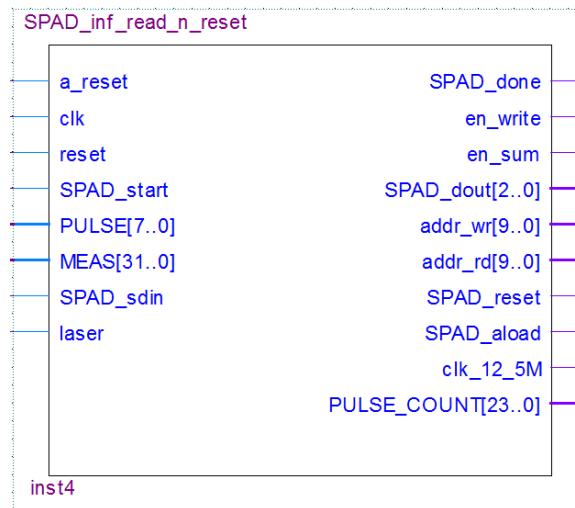


Figure 4. IC interface.

Table 4: Ports of the IC interface.

Port	Direction	From/To	Description
reset	Input	Finite state machine	Sets the interface to initial condition
SPAD_start	Input	Finite state machine	Start measurement
PULSE[7..0]	Input	Finite state machine	The pulse count the IC collects before it is read
MEAS[31..0]	Input	Finite state machine	The number that the IC is read
SPAD_sdin	Input	IC	Serial data
laser	Input	Laser trigger	Indicates the quenching of the SPAD matrix
SPAD_done	Output	Finite state machine	The measurement is done
en_write	Output	RAM	RAM write enable
en_sum	Output	Adder	Addition enable
SPAD_dout[2..0]	Output	Adder	Paralleled data
addr_wr[9..0]	Output	RAM	Write address
addr_rd[9..0]	Output	MUX	Read address
SPAD_reset	Output	IC	Clears the output buffer of the SPAD matrix
SPAD_aload	Output	IC	Begins the read of the IC output buffer
clk_12_5M	Output	IC	Serial 12,5 MHz clock
PULSE_COUNT[23..0]	Output	Data transmitter	The actual number of measured quenching cycles

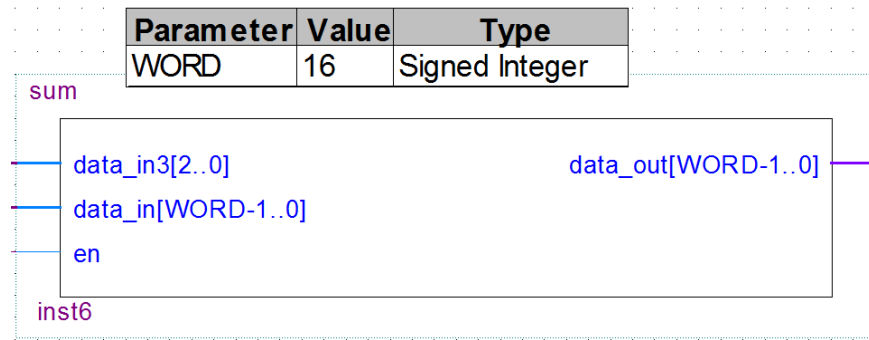


Figure 5. Adder.

Table 5. Ports of the adder.

Port	Direction	From/To	Purpose
data_in3[2..0]	Input	IC interface	Paralleled IC data
data_in[15..0]	Input	RAM	Previous value in the RAM
en	Input	IC interface	Sum enable. When disabled only data_in3 is in the output
data_out[15..0]	Output	RAM	The sum of the inputs or just the data_in3 input

Table 6. Ports of the RAM.

Port	Direction	From/To	Purpose
D[15..0]	Input	Adder	Write data
read_addr[9..0]	Input	MUX	Read address
write_addr[9..0]	Input	IC interface	Write address
WE	Input	IC interface	Write enable
Q[15..0]	Ouput	Multiple	Read data

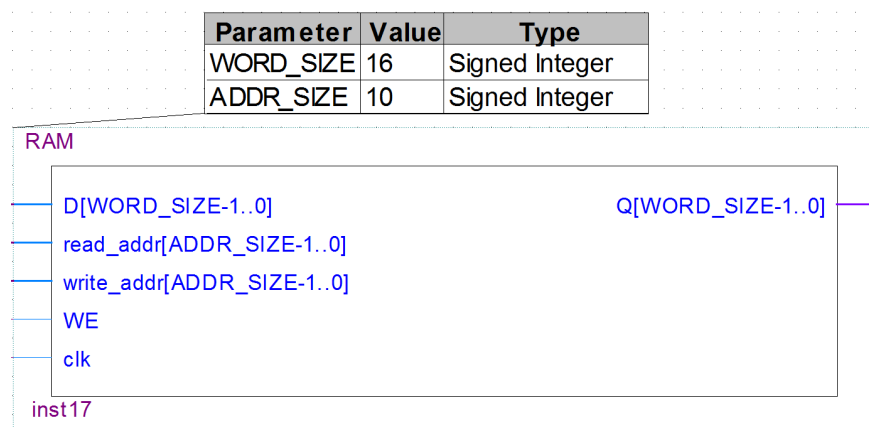


Figure 6. Random access memory.

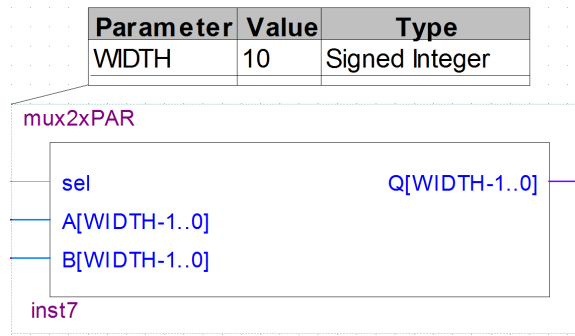


Figure 7. Address multiplexer.

Table 7. Ports of the address multiplexer.

Port	Direction	From/To	Purpose
sel	Input	Data transmitter	Channel selector
A[9..0]	Input	Data transmitter	Read address from the data transmitter
B[9..0]	Input	IC interface	Read address from the IC interface
Q[9..0]	Output	RAM	Read address

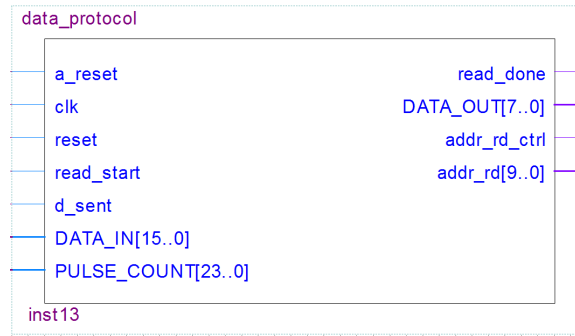


Figure 8. Data transmitter.

Table 8. Ports of the data transmitter.

Port	Direction	From/To	Purpose
reset	Input	Finite state machine	Sets the transmitter to the initial condition
read_start	Input	Finite state machine	Start transmission indicator
d_sent	Input	USB interface	A byte is sent
DATA_IN[15..0]	Input	RAM	Pure data
PULSE_COUNT[23..0]	Input	IC interface	The number of measured laser pulses
read_done	Output	Finite state machine	Transmission successfully done
DATA_OUT[7..0]	Output	USB interface	Framed data
addr_rd_ctrl	Output	MUX	Read address select
addr_rd[9..0]	Output	MUX	Read address

APPENDIX 2. THE STATE TRANSITION DIAGRAM OF THE FINITE STATE MACHINE (DESIGN 1)

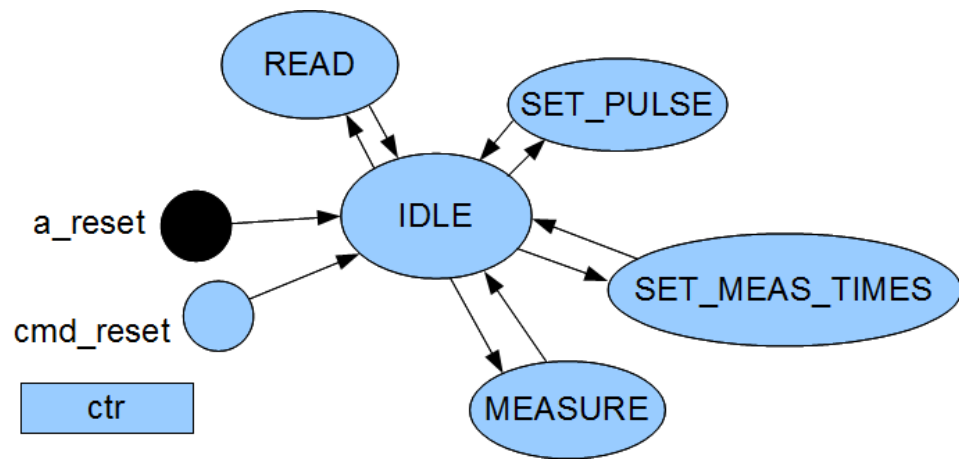


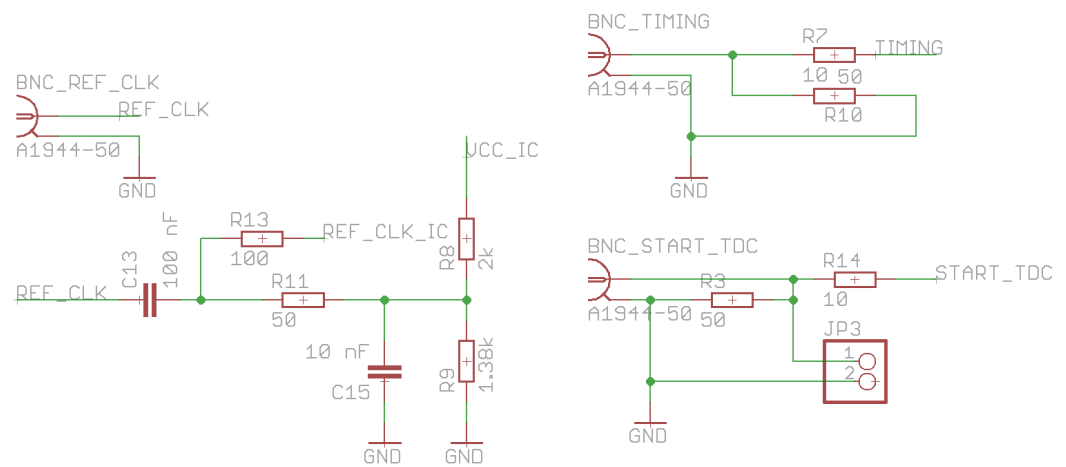
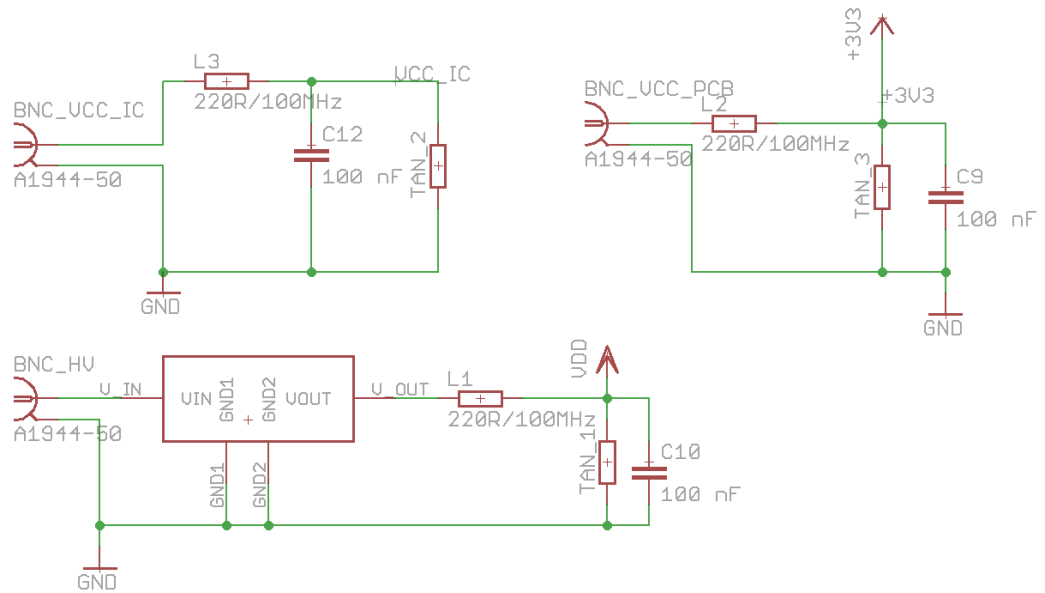
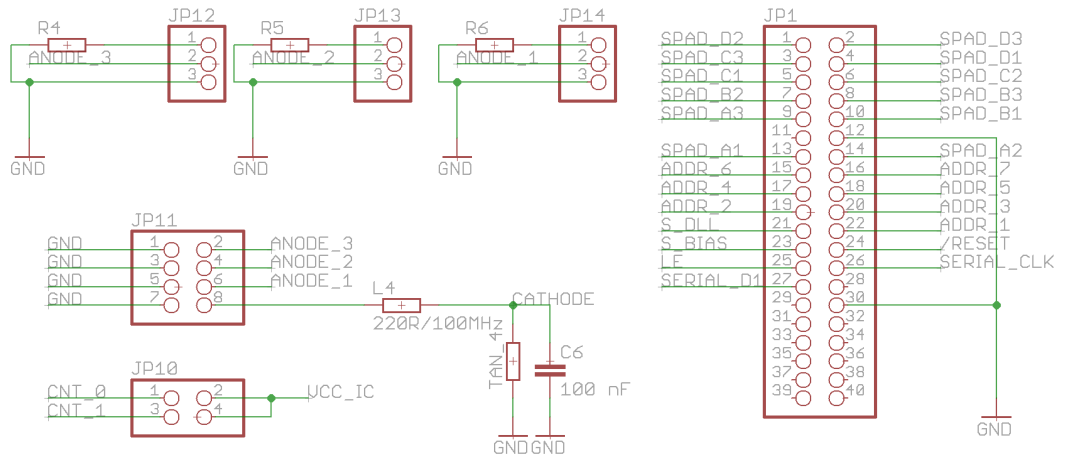
Figure 9. The state transition diagram.

Table 9: The conditions and actions of the state transition diagram.

From	To	Condition	Action
any state	IDLE	a_reset ¹	ack=0 disable_cmd=0 reset=0 SPAD_start=0 read_start=0 PULSE=1 MEAS=1 ctr=0
any state	IDLE	cmd_reset	ack=1 disable_cmd=0 reset=1 SPAD_start=0 read_start=0 ctr=0
IDLE	IDLE		ack=0 disable_cmd=0 reset=0 SPAD_start=0 read_start=0 ctr=0
IDLE	SET_PULSE	cmd_pulse	ack=1 disable_cmd=1

¹Asynchronous reset, active low.

From	To	Condition	Action
IDLE	SET_MEAS_TIMES	cmd_meas_times	ack=1 disable_cmd=1 ctr=0
IDLE	MEASURE	cmd_measure	ack=1 SPAD_start=1
IDLE	READ	cmd_read	ack=1 read_start=1
SET_PULSE	SET_PULSE		ack=0
SET_PULSE	IDLE	d_rec	disable_cmd=0 PULSE=DIN
SET_MEAS_TIMES	SET_MEAS_TIMES	d_rec d_rec AND ctr==0 d_rec AND ctr==1 d_rec AND ctr==2	ack=0 ctr++ MEAS[31..24]=DIN MEAS[23..16]=DIN MEAS[15..8]=DIN
SET_MEAS_TIMES	IDLE	d_rec AND ctr==3	disable_cmd=0 MEAS[7..0]=DIN ctr=0
READ	READ		ack=0 read_start=0
READ	IDLE	read_done	
MEASURE	MEASURE		ack=0 SPAD_start=0
MEASURE	IDLE	SPAD_done	



APPENDIX 4. THE LAYOUT OF THE TEST CARD (DESIGN 2)

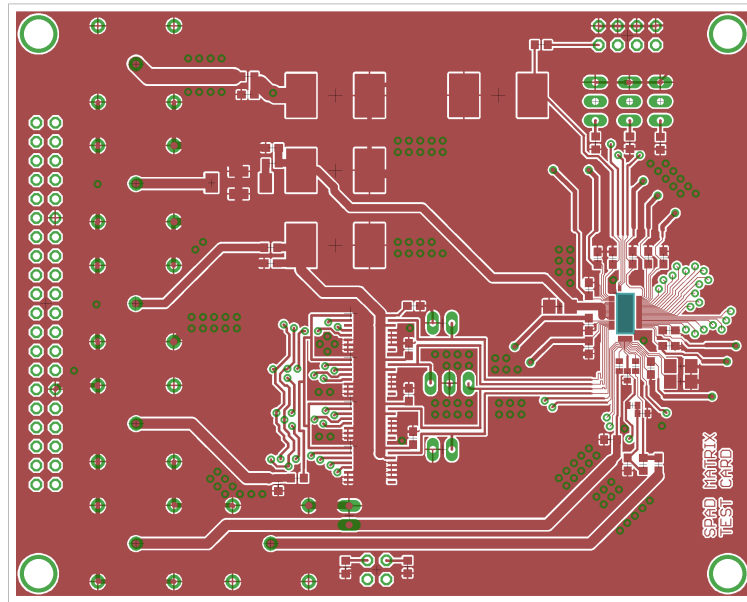


Figure 10. The top layer of the test card PCB.

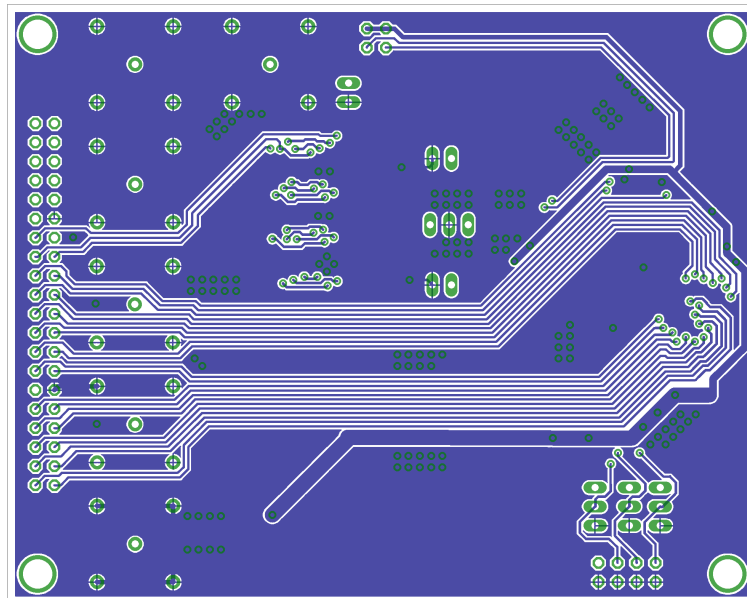


Figure 11. The bottom layer of the test card PCB.

APPENDIX 6. PROOFING

The convolution of the finite length stimulus (n) and impulse response (m):

$$y[j] = \sum_{k=0}^{n+m-2} x[k]h[j-k] \quad (4)$$

When each element of the m length impulse response $h[k]$ has a value 1, the convolution gives

j	y[j]	Number of elements
0	x[0]	1
1	x[0]+x[1]	2
2	x[0]+x[1]+x[2]	3
	...	
n+m-4	x[n-3]+x[n-2]+x[n-1]	3
n+m-3	x[n-2]+x[n-1]	2
n+m-2	x[n-1]	1

Lets define the sum of the convolution over the range of the j .

$$S = \sum_{j=0}^{n+m-2} y[j], \quad (5)$$

The definition is divided in three phases: $m < n$, $m = n$ and $m > n$.

When the $m < n$, the convolution elements are formed by the following way:

j	y[j]	Number of elements
0	x[0]	1
1	x[0]+x[1]	2
	...	
m-2	x[0]+...+x[m-2]	m-1
m-1	x[0]+...+x[m-1]	m
m	x[1]+...+x[m]	m
	...	
n-1	x[n-m]+...+x[n-1]	m
n	x[n-m+1]+...+x[n-1]	m-1
	...	
n+m-3	x[n-2]+x[n-1]	2
n+m-2	x[n-1]	1

The sum of the convolution is gives

$$\begin{array}{rcl}
 & S & \\
 \hline
 & x[0]+x[0]+x[1]+\dots+x[n-2]+x[n-1]+x[n-1] & \\
 = & x[0]+x[0]+x[0]+\dots+x[n-1]+x[n-1]+x[n-1] & \text{Reordering the elements} \\
 = & mx[0]+mx[1]+\dots+mx[n-1] & \text{Grouping} \\
 = & m(x[0]+x[1]+\dots+x[n-1]) &
 \end{array}$$

Thus, the sum S of the convolution $y[k]$ can be defined now as

$$S = \sum_{j=0}^{n+m-2} y[j] = m \sum_{k=0}^{n-1} x[k] \quad (6)$$

In the second phase, the lengths of the stimulus and the impulse response are equal ($m=n$).

j	y[j]	Number of elements
0	x[0]	1
1	x[0]+x[1]	2
	...	
m-2	x[0]+\dots+x[m-2]	m-1
m-1	x[0]+\dots+x[m-1]	m
m	x[1]+\dots+x[m-1]	m-1
	...	
n+m-3	x[m-2]+x[m-1]	2
n+m-2	x[m-1]	1

The sum S of the convolution $y[k]$ is now

$$\begin{array}{rcl}
 & S & \\
 \hline
 & x[0]+x[0]+x[1]+\dots+x[m-2]+x[m-1]+x[m-1] & \\
 = & x[0]+x[0]+x[0]+\dots+x[m-1]+x[m-1]+x[m-1] & \text{Reordering the elements} \\
 = & mx[0]+mx[1]+\dots+mx[m-1] & \text{Grouping} \\
 = & mx[0]+mx[1]+\dots+mx[n-1] & n=m \\
 = & m(x[0]+x[1]+\dots+x[n-1]) &
 \end{array}$$

The sum S is equal to the equation 6.

In the last phase, the impulse response $h[k]$ is longer than the stimulus $x[k]$ ($m > n$).

j	y[j]	Number of elements
0	x[0]	1
1	x[0]+x[1]	2
	...	
n-2	x[0]+...+x[n-2]	n-1
n	x[0]+...+x[n-1]	n
n+1	x[1]+...+x[n-1]	n
	...	
m-1	x[0]+...+x[n-1]	n
m	x[1]+...+x[n-1]	n-1
	...	
n+m-3	x[n-2]+x[n-1]	2
n+m-2	x[n-1]	1

Now, the convolution $y[j]$ has a constant value, when the j is between the closed range from n to $m-1$.

$$\begin{aligned}
 & S \\
 & \hline
 & x[0]+x[0]+x[1]+...+x[n-2]+x[n-1]+x[n-1] \\
 = & x[0]+x[0]+x[0]+...+x[n-1]+x[n-1]+x[n-1] && \text{Reordering the elements} \\
 = & mx[0]+mx[1]+...+mx[n-1] && y[j] \text{ is constant } m-n \text{ times} \\
 = & m(x[0]+x[1]+...+x[n-1])
 \end{aligned}$$

The sum S has the similar form as the equation 6. As all the three phases give the same result, the sum S of the convolution between a limited range stimulus $x[k]$ and a unit height rectangle impulse response $h[k]$ is defined in the equation 6. Besides, the sum S has a linear dependence with the length m of the impulse response $h[k]$.