

ULRR

Reconfigurable modular optical fibre resonance sensor system using field programmable gate array

Item Type	Thesis
Authors	Ong, Yong Sheng
Download date	2026-05-20 13:42:42
Item License	https://creativecommons.org/licenses/by-nc-sa/1.0/
Link to Item	https://hdl.handle.net/10344/8168



UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH

**Reconfigurable modular optical fibre
resonance sensor system using field
programmable gate array**

Submitted by Yong Sheng Ong

For the award of Doctor of Philosophy

Supervisors

Dr Ian Grout

Professor Elfed Lewis

Dr Waleed S. Mohammed

Department of Electronic and Computer Engineering

Submitted to the University of Limerick

January 2019

Abstract

The optical fibre sensor (OFS) has been studied for the last few decades and has found extensive use in many scientific and engineering fields including major application areas such as environmental, chemical, and biomedical. In spite of its popularity for use in a range of sensor applications, the OFS system is still facing hurdles for effective deployment in a range of environments. An OFS is a sensor that utilises an optical fibre as the sensing element or as a medium for light to propagate. A typical OFS system is physically large in size and is required to be used only within a fixed laboratory setting which limits its use in portable, on-site applications. To gain widespread use, a user friendly and portable optical sensor system is required and available for wide use. Such a system should be a device that is simple to use and the results easy to obtain and understand for personnel in the field.

The work discussed in this thesis considers the development of a modular field programmable gate array (FPGA) based OFS system. The use of the FPGA in an OFS design is explored as the enabling technology. This leads to a portable sensor system design that can perform data processing functions in the field without the need for a personal computer (PC). For sensor data classification, the k-nearest neighbour (kNN) machine learning algorithm has been adopted for use in this system to achieve embedded and real-time system operation. This work has used a surface Plasmon resonance (SPR) sensor to demonstrate the feasibility of the system. The sensor is connected to a light source and photodetector using a plastic optical fibre (POF), allowing the detection of refractive index that has its resonance wavelength fall within the visible wavelength range. A tricolour red, green and blue (RGB) light emitting diode (LED) was used as the light source with single photodiode used as the photodetector. Time domain intensity modulation of individual colour light was used to interrogate three bands of interest in the SPR spectrum using the photodiode. The SPR sensor was developed for use in chemical sensing applications. To test and evaluate the system operation, concentrations of different glucose solutions were classified using the kNN algorithm in this sensor system to demonstrate its feasibility. Two approaches were used in this work to reduce both memory requirement and time complexity through pre-processing and hardware acceleration.

This thesis is structured as follows. Chapter 1 will introduce the work and provide a rationale for the approach undertaken. The novel aspects of this contribution will be identified and discussed. Chapter 2 will review the OFS system considering both the SPR sensor and the portable, FPGA based system architecture. Chapter 3 will introduce the FPGA with its internal architecture that identifies the usefulness of the FPGA in embedded sensor system designs. Chapter 4 will introduce the kNN algorithm with different improvement techniques. The improvement technique is to lead towards an embedded classification implementation. The development of the sensor system will be discussed in Chapter 5. Chapter 6 will present the test results and embedded classification. Chapter 7 will conclude the thesis.

Declaration

A research project submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy from the University of Limerick. I declare that this research is my effort. It has not been presented for any other degree, part of a degree at this or any other university. I have made a reasonable effort to ensure that the work is original and where information was obtained from other sources, the sources have been cited and recognised in the text.

Name Yong Sheng Ong

Signature _____

Date _____

Acknowledgements

This work would not be complete without the contribution and help from numerous persons and firstly, I would like to express my gratitude to my supervisors, Dr. Ian Grout, Dr. Waleed S Muhammed and Prof. Elfed Lewis. They, as my supervisors, have been provided me with guidance in those numerous discussions. Their support and understanding also allowed me to explore different research directions and produce this piece of work.

I would like to extend my gratitude to friends in Optical Fibre Sensor Research Centre in the University of Limerick, Ling Xia, Ma Yu, Dinesh, Hong Wei, and Dr Zhu for sharing their ideas throughout this research journey and spending great time together. This gratitude also goes to friends in Bangkok University Center of Research in Optoelectronics, Communications and Computational Systems, Sunil, Hironmay, Romeo, Kevin and Manjunath. This has been a great adventure. See you guys again.

I would also like to thank staff in the electronics laboratories within the Department of Electronic and Computer Engineering for their support that has been essential for our research.

Dedication

To my mother, to my father, to my sister Wen Pei and to my brother Yao Sheng for their love and support throughout this life.

Table of Contents

Abstract	i
Declaration.....	ii
Acknowledgements	iii
Dedication	iv
Table of Contents.....	v
List of Figures	ix
List of Tables	xiii
Abbreviations.....	xiv

Chapter 1 Introduction

1.1	Introduction	1
1.2	Portable optical fibre sensor system	2
1.3	Application of the FPGA in a portable optical fibre sensor system	3
1.4	Aims and objectives of the research.....	5
1.5	New and novel aspects of the research.....	5
1.6	Sensor system prototype hardware.....	6
1.7	Structure of the thesis	7

Chapter 2 Optical fibre sensor system

2.1	Introduction	9
2.2	Optical fibre sensor system basics	10
2.2.1	Optical fibre sensor	10
2.2.2	Light source	11
2.2.3	Optical detector	12
2.3	Optical sensor modulation techniques	13
2.3.1	Intensity	13
2.3.2	Phase.....	13
2.3.3	Wavelength	14
2.4	Surface Plasmon resonance (SPR) sensor	15
2.4.1	Interrogation methods for optical fibre based SPR	17
2.5	Optical fibre system architecture review	18
2.6	Portable optical fibre sensor system	19
2.7	Summary.....	20

Chapter 3 Field programmable gate array

3.1	Introduction	21
3.2	FPGA basics	22
3.2.1	Architecture of the FPGA.....	22
3.2.2	Programming technology.....	23
3.2.3	FPGA Design flow	24
3.2.4	Commercially available FPGAs and CPLDs	25
3.3	Auxiliary On-Chip Components	27
3.3.1	Built-in ADC.....	27
3.3.2	DSP slice.....	27
3.3.3	Block RAM.....	28
3.3.4	IP core (common use component).....	28
3.4	Applications of the FPGA in a sensor system.....	28
3.5	Digilent® Arty FPGA development board	30
3.5.1	Xilinx® ADC	31
3.5.2	DSP slices - DSP48E1	32
3.5.3	7-series FPGAs Block RAM.....	32
3.6	Summary.....	33

Chapter 4 k-nearest neighbour classification

4.1	Introduction	34
4.2	k-nearest neighbour	35
4.2.1	Application of kNN	37
4.3	Methods to improve the kNN algorithm.....	38
4.3.1	Data dimension reduction	38
4.3.2	Partial distance	39
4.3.3	Pre-structure.....	39
4.3.4	Instance reduction method	40
4.4	Implementation of the kNN in an embedded system	42
4.4.1	kNN implementation architecture	43
4.5	Summary.....	46

Chapter 5 Experimental sensor system design

5.1	Introduction	47
5.2	SPR sensor.....	48
5.3	Sensor system electronic design	51
5.3.1	Power supply and regulator	51
5.3.2	Light source circuit.....	52
5.3.3	Photodetector circuit.....	53
5.3.4	XBee wireless communication.....	59

5.3.5	Printed circuit board development	60
5.4	FPGA hardware component design.....	61
5.4.1	Universal Asynchronous Receiver/Transmitter (UART)	62
5.4.2	XADC controller design.....	63
5.4.3	Light pulse generation	67
5.4.4	LCD display controller	69
5.4.5	Data processing unit	70
5.4.6	System summary	71
5.5	Interfacing with a PC.....	73
5.6	System performance.....	77
5.6.1	Optical source output.....	77
5.6.2	System response assessment	79
5.7	Summary.....	82
Chapter 6 Results and discussion		
6.1	Introduction	83
6.2	Sensor response.....	84
6.3	Classification validation using personal computer.....	87
6.3.1	Classification validation on Sensor A	90
6.3.2	Classification validation on Sensor B.....	92
6.4	FPGA-implemented classification component design and performance.....	94
6.4.1	Development of the kNN algorithm components.....	95
6.4.2	Synthesis results	100
6.4.3	Real-time classification	101
6.5	Summary.....	102
Chapter 7 Conclusions		
7.1	Thesis summary	104
7.2	Contribution of the work	105
7.3	Future work.....	106
References.....		108
Appendix A	Publication	A-1
Appendix B	Software and hardware used.....	B-1
Appendix C	Xilinx® Vivado.....	C-1
Appendix D	Digilent® Arty board.....	D-1
Appendix E	Electronic, optoelectronic and optical components.....	E-1
Appendix F	Printed circuit board	F-1

Appendix G Register transfer level (RTL) design description schematicsG-1

List of Figures

Figure 1-1 Simplified architecture of an OFS system.	2
Figure 1-2 Sensor system prototype hardware: a) Digilent® Arty 7 FPGA Development Board. b) Printed circuit board. c) Optical fibre sensor (SPR) and LED-fibre coupler (circled in red dashed line).....	6
Figure 2-1 Optical fibre structure.	10
Figure 2-2 Spectra reading technique.	12
Figure 2-3 Intensity based OFS using a reflective surface.....	13
Figure 2-4 Fabry-Perot sensor for pressure sensing.	14
Figure 2-5 Fibre Bragg grating.....	15
Figure 2-6 Kretschmann configuration. (I is incident light)	15
Figure 2-7 Typical responses for intensity and wavelength interrogation.	16
Figure 3-1 Simplified FPGA architecture.	23
Figure 3-2 Simplified FPGA design flow.	24
Figure 3-3 Arty FPGA development board.....	31
Figure 3-4 Simplified XADC block diagram.	31
Figure 3-5 Simplified DSP48E1 architecture.....	32
Figure 4-1 kNN classification example for $k = 3$ and 5 (boxed question mark is the new unknown data).	35
Figure 4-2 An example of kd-tree (2d-tree).	40
Figure 4-3 Adder tree (D input) for Euclidean distance calculation.....	44
Figure 4-4 General two-dimensional systolic array.....	45
Figure 5-1 Sensor system breakdown.	48
Figure 5-2 BU-SPR sensor: a) schematic diagram. b) image.....	49
Figure 5-3 Sensor response with different measurand (Refractive Index).....	50
Figure 5-4 Sensor system and connection.....	51

Figure 5-5 LED driver – current source and potential divider. Input pulse represent FPGA output.	52
Figure 5-6 LED driver output - FPGA output in yellow line (first channel) and voltage across current sensing resistor (second channel).	53
Figure 5-7(a) Feedback circuit block diagram. (b) Feedback loop is broken to find out the loop gain.	54
Figure 5-8 Feedback analysis circuit (loop is broken to inject test signal).	55
Figure 5-9 Bode Plot of loop gain signal.	56
Figure 5-10 Transient analysis of optical receiver circuit.	57
Figure 5-11 Transient behaviour (with 220 fF effective feedback capacitance) of transimpedance circuit in current system.	58
Figure 5-12 Transient behaviour(with 220 fF effective feedback capacitance) of amplified transimpedance circuit in current system.	58
Figure 5-13 Transient behaviour of shielded amplified transimpedance circuit in current system.	59
Figure 5-14 Developed PCB with the FPGA development board (underneath).	61
Figure 5-15 Block diagram for the FPGA.	62
Figure 5-16 UART data packet.	62
Figure 5-17 RTL code schematic for the UART module.	63
Figure 5-18 Simulation waveform of the UART module (RX and TX).	63
Figure 5-19 XADC sampling capacitor charging simulation circuit.	64
Figure 5-20 XADC sampling circuit simulation result.	65
Figure 5-21 Simplified XADC controller module.	65
Figure 5-22 XADC controller a) state timing and b) state diagram.	66
Figure 5-23 Simplified light pulse generator module.	67
Figure 5-24 Light pulse generator a) light channel selection mechanism and state timing. b) state diagram.	68
Figure 5-25 Simplified LCD controller module.	69
Figure 5-26 LCD abstract state diagram.	69
Figure 5-27 LCD controller state diagram.	70

Figure 5-28 Simplified data processing unit module.....	70
Figure 5-29 LCD controller state diagram for a) UART transmission and b) LCD display.	71
Figure 5-30 Resources usage for the sensor system.	71
Figure 5-31 Simplified digital design system module connection.	72
Figure 5-32 Data acquisition and termination process.....	74
Figure 5-33 Incoming data packets and its content.	75
Figure 5-34 GUI for system control and display.....	76
Figure 5-35 Output intensity (a.u., arbitrary unit) as function of different pulse width modulation level (min – max = 0 – 15) for red, green, blue LED indicated by their line colour.	78
Figure 5-36 Peak wavelength of red, green, and blue LED for different PWM levels.....	79
Figure 5-37 Time series of LED output data (a.u. = arbitrary unit) measured using the system (dash line and solid line represent averaged data and raw data, respectively).	80
Figure 5-38 Zoomed-in plot for time series of the red LED output.....	81
Figure 5-39 Output repeatability assessment. (a.u. = arbitrary unit)	82
Figure 6-1 Embedded data processing architecture.....	83
Figure 6-2 Sensor (50 nm gold thickness) response(arbitrary unit, a.u.).	85
Figure 6-3 Sensor (70 nm gold thickness) response (arbitrary unit, a.u.).....	86
Figure 6-4 Representatives generation process schematic diagram.	89
Figure 6-5 Representatives generation from Sensor A training set for scenarios included centroid, and J=2, 4, 6 from CURE approach. Training set is shown as black dot and representatives are shown as enlarged colour dots.....	91
Figure 6-6 Representatives generation from sensor B training set for scenarios included centroid, and J=2, 4, 6 from CURE approach. Training set is shown as black dot and representatives are shown as enlarged colour dots.....	93
Figure 6-7 Architecture of the kNN classification in FPGA.....	95
Figure 6-8 kNN module's hierarchy.....	96
Figure 6-9 a) Systolic array architecture of distance calculator. b) Enable signal flow.	97
Figure 6-10 Data flow process in distance calculator.....	98
Figure 6-11 Architecture of distance sorting.....	99

Figure 6-12 Majority voting process.	100
Figure 7-1 Proposed quasi-distributed sensor system based on FPGA.....	107
Figure C-1 Design flow [104].	C-1
Figure D-1 Digilent® Arty Board.	D-1
Figure E-1 Common anode tri-colour LED.	E-1
Figure E-2 IF-D91 photodiode.	E-2
Figure E-3 GH4001 plastic optical fibre.....	E-3
Figure E-4 LCD2004 20x4 character LCD display module.....	E-4
Figure F-1 PCB- top surface.	F-1
Figure F-2 PCB- bottom surface.	F-1
Figure F-3 Schematic of the sensor system’s electronic circuit.	F-2
Figure G-1 Light pulse generation module RTL design description schematic.	G-2
Figure G-2 LCD controller module RTL design description schematic.....	G-3
Figure G-3 XADC controller module RTL design description schematic.....	G-4
Figure G-4 Data processing module RTL design description schematic.....	G-5
Figure G-5 Top level RTL design description schematic.	G-6

List of Tables

Table 3-1 Xilinx® products.	26
Table 3-2 Intel® (previously Altera) products.	26
Table 3-3 Lattice Semiconductor products.	26
Table 3-4 Microsemi products.	27
Table 5-1 Phase margin for sweep simulation.	56
Table 6-1 Solution's refractive index.	84
Table 6-2 Number of collected samples for Sensor A.	90
Table 6-3 Validation success rate for different representatives points (Sensor A, 50nm gold thickness).	91
Table 6-4 Test success rate for sensor A at optimum case, J=4.	92
Table 6-5 Number of collected samples for sensor B.	92
Table 6-6 Validation success rate for different representatives points (Sensor B, 70nm gold thickness).	94
Table 6-7 Test success rate for sensor B at optimum case, J=4.	94
Table 6-8 Implementation results for part and component.	101
Table 6-9 Confusion matrix for the real-time embedded kNN based refractive index classifier. Predicted classes are on the columns and actual classes are on the rows.	102
Table B-1 Software.	B-1
Table B-2 Hardware.	B-1
Table E-1 IF-D91 characteristics.	E-2
Table E-2 GH4001 characteristics.	E-3
Table G-1 List of module RTL.	G-1

Abbreviations

ADC	Analogue-to-digital converter Analog-to-digital converter
ARM	Advanced RISC Machine
ASIC	Application specific integrated circuit
a.u.	Arbitrary unit
BRAM	Block random access memory
BU-SPR	Bangkok University-SPR
CAN	Controller area network
CCD	Charged coupled device
CLB	Configurable logic block
CMOS	Complementary metal oxide semiconductor
CPLD	Complex programmable logic device
CSV	Comma separated values
CURE	Clustering using representatives
DAC	Digital-to-analogue converter Digital-to-analog converter
DAQ	Data acquisition
DI	Deionized

DIP	Dual in-line package
DSP	Digital signal processing Digital signal processor
EEPROM	Electrically erasable PROM
FBG	Fibre Bragg grating
FIFO	First-in first-out
FP	Fabry-Perot
FPGA	Field programmable gate array
FSM	Finite state machine
GOF	Glass optical fibre Glass optical fiber
GPU	Graphics processing unit
GUI	Graphical user interface
HDL	Hardware description language
I/O	Input/Output
IC	Integrated circuit
IDE	Integrated development environment
IV	Current-to-voltage
IP	Intellectual property
kNN	k-nearest neighbour

LED	Light emitting diode
LCD	Liquid crystal display
LDO	Linear dropout voltage regulator
LUT	Lookup table
LSB	Least significant bit
MAC	Multiply accumulate
NA	Numerical aperture
Near-IR	Near-infrared
OFS	Optical fibre sensor Optical fiber sensor
PC	Personal computer
PCB	Printed circuit board
PE	Processing element
PL	Programmable logic
PLD	Programmable logic device
POC	Point of care
POF	Plastic optical fibre Plastic optical fiber
PROM	Programmable ROM

PWM	Pulse width modulated Pulse width modulation
RAM	Random access memory
RGB	Red-Green-Blue
RI	Refractive index
RISC	Reduced instruction set computer
ROM	Read only memory
RTL	Register transfer level
SoC	System on chip
SPI	Serial peripheral interface
SPR	Surface Plasmon resonance
SRAM	Static random access memory
TE	Transverse electric
TM	Transverse magnetic
UART	Universal asynchronous receiver transmitter
USB	Universal serial bus
UV	Ultraviolet
VHDL	VHSIC hardware description language
VHSIC	Very high speed integrated circuit

VI	Voltage-to-current
WTA	<i>“winners take all”</i>
XADC	Xilinx® ADC
μC	Microcontroller
μP	Microprocessor

Chapter 1 Introduction

1.1 Introduction

A sensor is a device that detects events or changes in the targeted parameters. The sensor has become an indispensable device as it provides much needed information that includes force, chemical composition, temperature and other parameters that may affect human behaviour. The information obtained, for example, enables a greater understanding into environmental changes and the impact of human activity on the environment. Sensor types, that include the optical sensor, continue to be studied and developed. The focus of the work presented in this thesis is on the optical fibre sensor (OFS) and over the last few decades, advancements in OFS design and use have been rapid. It has been developed not only as a substitute to an electrical counterpart, but sometimes as a unique solution to an application. The OFS has found extensive use in many scientific and engineering fields including major application areas such as environmental, chemical, and biomedical. The demand for optical fibre sensors is growing with the need to acquire more knowledge concerning the environment.

To measure a physical quantity, the OFS modifies the properties of transmitted light through an optical fibre under the influence of a measurand. The sensing element can be part of the optical fibre itself (intrinsic) or external to the optical fibre (extrinsic). The advantages of fibre optic sensing lies within its material and light properties that include light-weight, small size, high sensitivity and intrinsic safety in hazardous environments. An OFS system is composed of multiple subsystems that implement system control, optical signal generation, data acquisition and analysis, communication and a human user interface. The OFS system's performance depends on the sensor design as well as supporting instrumentation and analytical tools used for data processing [1]. Figure 1-1 depicts a simplified architecture of an OFS system. The optical fibre guides the light from the light source to the sensing element and then towards the optical receiver. On the sensing element side, properties of the guided light will be modified under influence of the parameters to be monitored. The optical receiver, and sometimes the light source, is connected to a computing device that provides control and data analysis. A suitable electronic circuit hardware interface is required to establish the electrical connection to the computing

device and, where a suitable software programmable device is used, suitable software program(s) to perform necessary system operations.

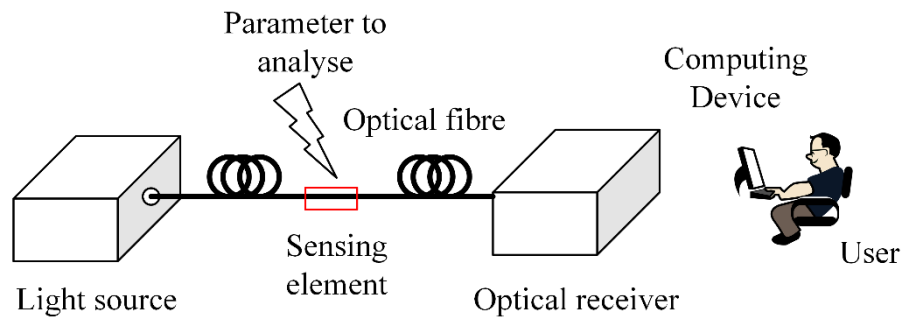


Figure 1-1 Simplified architecture of an OFS system.

Various ideas have been proposed and designs have been created to enhance the OFS in terms of sensitivity and selectivity, and new techniques have been developed for undertaking measurements. Some OFS systems have been successfully commercialised whilst others are still facing hurdles for effective deployment in a range of environments. To date, the OFS system is still largely reliant on a personal computer (PC) and static laboratory instrumentation for deployment. These requirements are limiting factors in some situations such as the need for low-cost, portable applications.

1.2 Portable optical fibre sensor system

The conventional approach to acquire sample data requires personnel to manually handle and collect a sample on site and then to send it to a laboratory for analysis. Such an approach is laborious, time consuming, costly and the integrity of sample can be compromised at each stage of the process [2]. The design of an OFS system that addresses the equipment portability and on-site analysis issues is gaining attention to realise a “bringing the laboratory to the sample” concept instead of the other way round. In addition, a portable design can also be beneficial in a stationary environment such as a factory or in the home due to ease of installation and requiring less planning for use.

The feasibility of a portable optical sensing system requires careful consideration of factors that include cost, sample characteristics, user technical skills and operation constraints. The operation constraints include system maintenance, availability of consumables and power requirements [1]. To achieve a small physical size and a lightweight OFS system design, optoelectronic components such as the light emitting diode (LED) and photodetector are often utilised in a portable system as they offer low power operation and compact size. The LED is a preferred option for a portable sensing application as it provides a stable spectral output and is available in wide range of wavelengths from ultraviolet (UV) to near-infrared (near-IR) (247 to 1550 nm) [2].

This combined with widely available and small physical size computing hardware and electronic parts can advance OFS system design toward portability. Apart from being portable, these parts can also reduce the total cost of system ownership to an affordable level and this allows more user adoption of this type of system.

Regardless of the size of the device, the basic architecture of an OFS system remains largely unchanged as shown in Figure 1-1 and consists of several key parts: light source, optics setup, optical receiver and processing unit for signal processing, system control, communication and readout. Different studies have proposed a portable system design that utilise a software programmed device including the microcontroller (μC) as the processing unit instead of standard PC arrangement to coordinate the operation of the subsystems and to ensure the system is running efficiently whilst only requiring to be battery powered. This implementation provides a lower cost option over the PC and smaller physical size. Such implementations have been pursued to meet the growing demand of the optical sensor.

1.3 Application of the FPGA in a portable optical fibre sensor system

A software programmed device based system can be easily programmed as its operation is described in software code and a wealth of experience exists in the use of these devices. These software programmed devices are the microcontroller (μC), microprocessor (μP) and digital signal processor (DSP). Software programmed devices are capable of handling sensor inputs that rely on a simple working principle. However, the majority of the software programmed device's operation needs to be implemented in sequential fashion due to the sequential operation nature of the software code, typically written in the ANSI-C structured programming language. The sequential nature of the system operation could however limit the potential capabilities of the system. In addition, the system operation could be more effective if concurrent (parallel) operation was possible by executing multiple operations in parallel and the completion of operations in a reduced time.

A concurrent circuit operation capability is readily offered by a hardware-configured counterpart such as the field programmable gate array (FPGA) and complex programmable logic gate device (CPLD). Hardware configured devices such as the FPGA also allow for the control of the digital system operation at the register level so that user can manipulate the timing of individual operations, or at higher level using behavioural modelling approaches. Concurrency, combined with the ability of controlling system operation at the register level, can be of benefit to the implementation of an algorithm that requires synchronisation or temporal information [3]. This has been proven in applications such as fluorescent measurement [4], real time classification [5] and lock-in amplification [6, 7]. Modern FPGAs are also provided with digital signal processor

(DSP) features that support multiply accumulate (MAC) operations and internal block random access memory (BRAM) to store intermediate data [8]. These features allow users to solve complex signal processing tasks in single integrated circuit (IC). This allows the FPGA to create high levels of integration by eliminating the need of external ICs such as RAM for some applications. This can result in a cost reductions and a lower power consumption. Therefore, the FPGA is a strong candidate as a processing unit that is capable of facilitating a portable OFS system to conduct real-time monitoring. This is especially true for circumstances that require the handling of large amounts of data [8]. Hardware configured devices (similar to μC) can readily accommodate different communication requirements such as the UART (universal asynchronous receiver transmitter), CAN (controller area network) and SPI (serial peripheral interface), and hence it can interface with other systems with a different communications interface. The configurability and large number of I/O available with the FPGA also allow the system to adopt new interface requirements and therefore have a longer lifetime instead of becoming obsolete when new requirements occur. The system could be updated according to user needs by loading the required system update configuration [9].

To recognise the change in a monitored parameter in a meaningful way rather than an electrical signal (current or voltage), the obtained sensor data is required to undergo some form of data processing and analysis. Depending on the nature of the measured signal, different approaches for data analysis can be used including regression [10], classification [8] and clustering [11]. The first two techniques are supervised learning methods that require well-labelled data. Regression, in its simplest form, attempts to fit a straight linear relationship (this being linear regression) between variables whilst classification is an approach for predicting variables categories. The third technique, clustering, forms data clusters based on inherent structures within a dataset. These approaches also subjected to different design and implementation issues that determine which hardware/software platform they would be best implemented on.

One of the popular intuitive data analysis techniques, the k-nearest neighbour (kNN), is suitable for use in a range of different situations. It is an instance-based learning method that categorises input data using stored information without the use of model and is sometimes used as a regression method without a model. This classification method is capable of achieving a low classification error that is no greater than twice the Bayes error rate [12]. The Bayes error rate is commonly used as a classification benchmark. A low error rate requires the use of a large amount of stored information. It is however inefficient to perform the kNN algorithm in software within a μC due to large amount of sequential operations that are required to be executed. Different approaches are used to address the complexity and time issues such as creating a space-partitioning data structure for neighbour searching, data reduction by removing redundant or noisy data, and hardware acceleration using a graphics processing unit (GPU) or FPGA. There

have been studies that have employed the FPGA to accelerate the computation process to achieve a reasonable computation time and circuit power consumption. The FPGA, as a powerful processing unit, is useful in resource-limited situations including field studies where on-site and complex analysis is needed.

1.4 Aims and objectives of the research

The research discussed in this thesis aimed to investigate the design of OFS systems and to develop a cost efficient, modular, and portable OFS system using the FPGA. The system design is based on a Xilinx[®] Artix-7 (XC7A100TCSG324-1) FPGA to utilise hardware advantages such as the ability to design at the register level and to utilise its built-in DSP capabilities. The sensor used to demonstrate the feasibility of the system was an extrinsic surface Plasmon resonance (SPR) sensor that measures refractive index. A small footprint additional circuit was also included to integrate the sensor with the FPGA. This enabled a modular system to be developed where the sensor and electronics were interchangeable.

The aims of this work were achieved through implementation of the following objectives:

1. To develop a small footprint electronic system for a resonance based optical fibre sensor.
2. To design an algorithm to be implemented within the FPGA for LED pulse detection and extraction of information regarding the LED pulse amplitudes.
3. To develop a software script with a graphical user interface (GUI) for communications and to allow for wireless data transfer between the sensor system and the PC.
4. To improve the computational time requirements for the k-nearest neighbour classification using parallelism.
5. To develop a hardware implementation of the k-nearest neighbour classification algorithm for real time classification.

1.5 New and novel aspects of the research

To fulfil the aims and objectives stated above, the novel aspects of the work are as following:

1. Implementation of refractive index classification based on the transmission response of a tri-colour LED.
2. The use of a simple instance reduction method, CURE (clustering method) was adapted, as a supervised learning method.
3. Dimension reduction for the kNN algorithm using the difference of a colour's intensity, which also provided an inherent self-referencing capability.

4. A flexible architecture based on systolic array for Euclidean distance calculation (implementation using the FPGA).
5. An embedded kNN algorithm implementation using the FPGA for resonance type optical sensing.

1.6 Sensor system prototype hardware

As discussed in the previous section, an OFS system design consists of four main components, (i) an OFS, (ii) a controller unit, (iii) a light source, and (iv) an optical receiver. In this work, the sensor system design consisted of a Digilent® Arty A7-35T Development Board, a self-developed printed circuit board (PCB) and the OFS as depicted in Figure 1-2. The OFS (SPR) is connected to the optoelectronics using a plastic optical fibre (POF). A PCB that consists of a tri-colour LED as light source and photodiode as an optical receiver was developed. The FPGA acts as a controller to coordinate sensor interrogation and processes the sensor data.

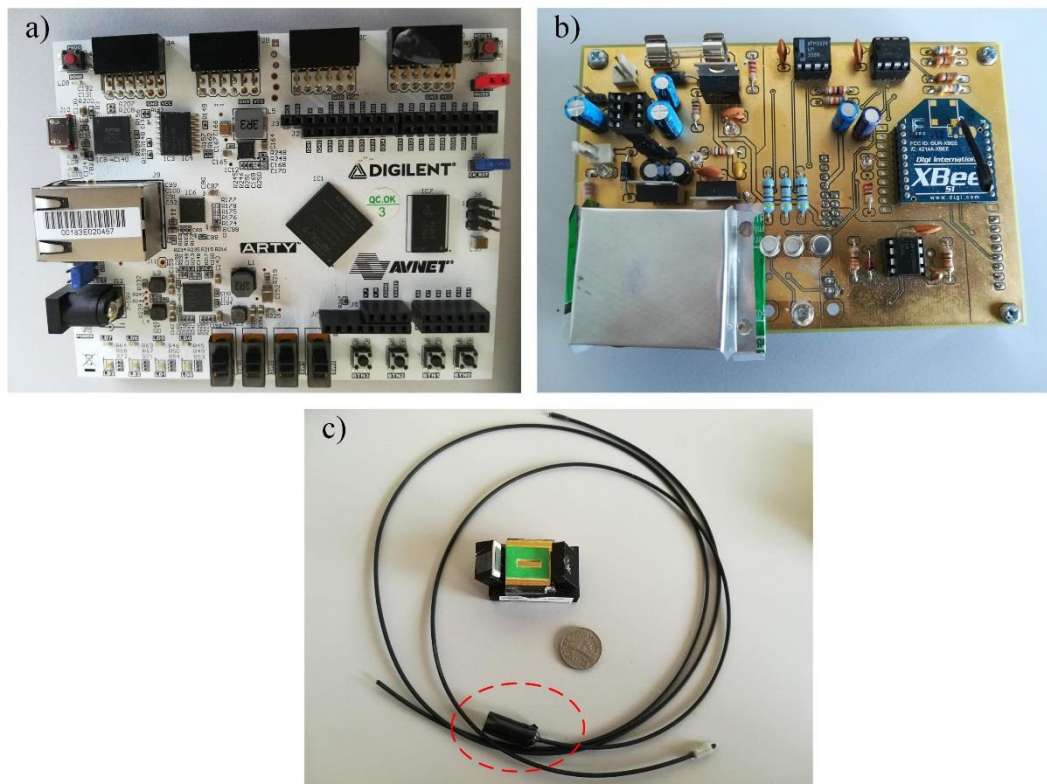


Figure 1-2 Sensor system prototype hardware: a) Digilent® Arty 7 FPGA Development Board. b) Printed circuit board. c) Optical fibre sensor (SPR) and LED-fibre coupler (circled in red dashed line).

1.7 Structure of the thesis

This thesis comprises seven chapters and focuses on the development of a portable, modular and cost efficient sensor system. The thesis is structured as follows:

- **Chapter 1: Introduction**

This chapter has discussed the growing need for sensors and the importance of portable OFS sensor systems. Different implementations of portable OFS sensor system have been considered and an implementation based on the FPGA explored.

- **Chapter 2: Optical fibre sensor system**

This chapter reviews the architecture of an OFS system along with a brief introduction of optical fibres and a discussion into common light modulation techniques for application in an OFS. SPR sensing methods are also highlighted.

- **Chapter 3: Field programmable gate array**

This chapter describes the architecture of the FPGA and reviews currently available vendors and their technologies. The FPGA design flow is also described along with an introduction to the built-in logic blocks provided by vendors. Application of the FPGA in a sensor system is reviewed and its strengths highlighted.

- **Chapter 4: Embedded data classification**

This chapter reviews data classification with a focus on the kNN algorithm. kNN algorithm background and practices for kNN efficiency enhancement are also highlighted. Implementation of the kNN in an embedded system (including the FPGA) is discussed.

- **Chapter 5: Experimental sensor system design**

In this chapter, the subsystems including: light source driver circuit, optical receiver, FPGA and SPR sensor are discussed in the creation of an OFS system. The subsystems' build and integration are discussed along with SPR sensor's feature. The required characteristics of the subsystems are focused on the design of these subsystems. The design of the software using Python and its interface to support the system operation are also discussed.

- **Chapter 6: Results and discussion**

This chapter demonstrates the operation of the prototype OFS system in a laboratory environment. The function of whole system as a readout unit is investigated using different weight ratios of glucose and water. The results obtained are compared to the test result using a spectrometer. The implementation of classification in both software

and hardware is discussed along with real time classification executed within the FPGA. An extended period of testing in a latter part of the section demonstrates the sensor performance in laboratory environment.

- **Chapter 7: Conclusions**

This final chapter summarises the development of a modular FPGA based OFS system aimed for mobility. At the end of chapter, recommendations for potential future work into system performance enhancement and extension are suggested.

Chapter 2 Optical fibre sensor system

2.1 Introduction

It has been a few decades since the study of optical fibre sensor (OFS) began. Different techniques and ideas have been proposed and studied for different applications including environmental sensing, structural health monitoring and biomedical [13-15]. The OFS has advantages over other sensor types that include immunity to electromagnetic interference, lightweight, and high sensitivity. Compared to a free space optical sensor, the OFS requires minimal alignment for light coupling and signal collection. However, an optical sensor cannot operate by itself and hence requires supporting elements to perform the required sensing operations. A typical OFS system comprises a sensing element, optical source, matching optical receiver, processing unit and a readout (communications) unit. Any OFS can be categorised by the sensing element as well as the optical modulation mechanism that includes intensity, phase and wavelength interrogation. The modulation mechanisms depend on the nature of the light source and the matching optical receiver used. However, the general architecture of most of the OFS systems is generally the same.

Surface Plasmon resonance (SPR) is a phenomenon that have been used for a wide range of sensing applications including biochemical [16], and environmental monitoring [17]. This optical sensing technique is capable of performing real-time and label-free monitoring (without using any molecular marking technique such as fluorescent labelling or isotope labelling) as the measurements are based on refractive index (RI) changes. There are continuous research and development activities aimed at developing SPR based sensing. Different optical configurations have been studied to excite the SPR including prism based, grating based and optical waveguide based [18]. The optical fibre sensor system configurations have proved to be simple and favoured by different applications.

In this chapter, the first two sections discuss the architecture and operation of a general optical fibre sensor system. Section 2.2 reviews OFS basics. Section 2.3 reviews OFS modulation techniques. The basics of SPR sensing are discussed in Section 2.4 and Section 2.5 presents OFS

system architecture examples for the intensity and wavelength interrogation. Realisation of a portable setup for the OFS is also explored in the section. Section 2.7 summarises this chapter.

2.2 Optical fibre sensor system basics

2.2.1 Optical fibre sensor

The optical fibre is a cylindrical dielectric waveguide. A step index fibre, for example, is made of two different materials where a lower refractive index cladding surrounds a higher refractive index core as shown in Figure 2-1. Light is guided inside the optical fibre in the form of discrete modes. The allowed modes' propagation constants satisfy two conditions: (i) total internal reflection, and (ii) sustaining constructive interference when propagating inside the fibre [19]. This results in a low transmission loss for the optical fibre. Additional external layers are usually added to provide mechanical strength and protection from environmental effects. The thin structure of the optical fibre makes it flexible and light weight. There are two major optical fibre materials, polymer and silica. Silica optical fibre typically has two main low loss transmission windows around 1550 nm and 1330 nm [19]. It is also inert to most chemicals. The polymer (plastic) optical fibre (POF) on the other hand has a low transmission loss in the visible light region. It is also easier to handle. POF usually comes with large core diameter (the typical diameter is around 1 mm). Both materials offer high resistance to electromagnetic interference and have no cross-talk between close proximity fibre optics. The silica optical fibre is usually used in communications and serves as the network backbone with the need for fewer signal repeaters whereas POF caters for short distance transmission applications due to its ease of handling. Their capabilities have led to their application being extended from communication to the field of sensors.

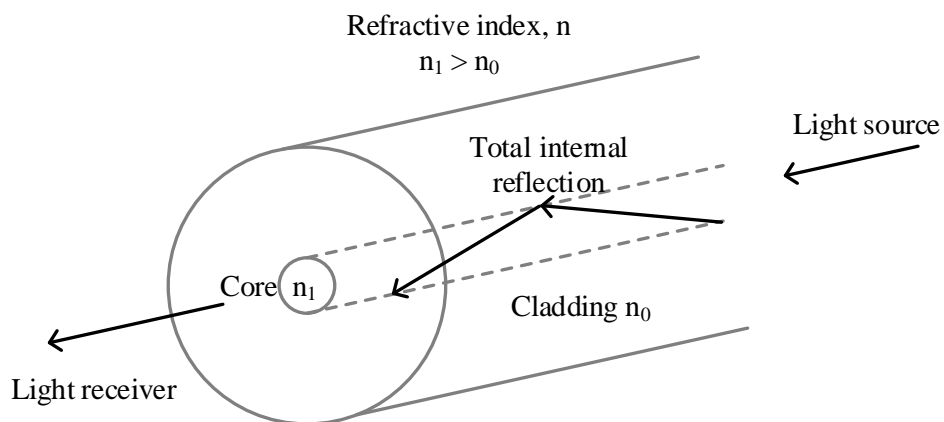


Figure 2-1 Optical fibre structure.

Typically, optical fibres are used to sense the targeted parameters in two distinct ways: extrinsic or intrinsic. The key difference between the two types of sensor is that the sensing mechanism takes place within or external to the optical fibre, respectively. The OFS modifies the guided light's properties in response to the targeted parameter: physical, chemical, or biological. The optical properties such as intensity, wavelength, and phase of the guided light are then interpreted using the corresponding device [20]. For an intrinsic OFS, the light interacts with the element of the optical fibre or built-in structure within the optical fibre such as a fibre Bragg grating [21]. The extrinsic OFS requires that the guided light is coupled to an external device to interact with sensing element such as an extrinsic Fabry-Perot interferometer [22]. The optical fibre hence functions as signal carrier. An OFS can be further categorised in term of its light modulation technique as discussed in section 2.3.

In order to implement the OFS system, a number of components such as light source, optical detector and computing unit are required [20]. Depending on the sensing mechanism, different arrangements of the light source and optical detectors are required for the necessary characteristics to modulate the OFS and extract the desired output. There are a wide range of devices for both light source and optical detectors. However, only devices that are related to the OFS system are discussed in the next two sections.

2.2.2 Light source

There are generally two main categories of light source, coherent and non-coherent [20]. The non-coherent source includes the incandescent lamp (e.g., Tungsten-Halogen lamp), arc lamp (e.g., deuterium lamp) and solid state source (e.g., light emitting diode (LED)). The incandescent lamp produces a light that covers a large band of wavelengths due to blackbody radiation. The arc lamp produces light by an electric arc that is formed under a high potential difference. The LED emits light in the wavelength that is proportional to its band gap energy and covers a narrow band of wavelength (a few tens of nm). For the incandescent source, it requires time to reach thermal equilibrium to produce a stable spectral output. On the other hand, the arc lamp and the LED have a relatively stable output. The LED produces light in a cone shape that is limited in some angle while both arc and incandescent lamps produce light equally in all directions [23]. The coherent source, the laser, is available in different forms including gas laser, crystal laser and semiconductor laser. The laser produces highly monochromatic and directional output [24]. Directional light sources such as the LED and the laser are favoured in some applications as no light focusing lenses are required for light coupling. This places the laser as a strong candidate as a light source but the wider adoption of it is hindered by requirements such as operation safety and power consumption (for battery powered applications) [20].

2.2.3 Optical detector

The optical detector (photon detector) can be categorised as being either intensity detection or spectral detection [20]. The intensity detection device can be further classified by its operating principle such as photoemissive, photoconductive, and the PN junction [24, 25]. A photoemissive device emits an electron when its conducting surface absorbs a photon. The free electrons are then collected in an external circuit and thus produce a detectable electric current that corresponds to the number of emitted electrons (which also corresponding to the number of incoming photons). A photoconductive device such as photoresistor is a semiconductor device that changes its electrical conductivity under the influence of light. The device is sensitive to photons with energies higher than a certain amount and its sensitivity is dependent on its material doping. It is prone to be influenced by ambient temperature and a long response time is needed. A PN junction device is also a semiconductor device but its working principle involves a PN semiconductor junction such as photodiode or phototransistor. A wavelength encoding signal is required to resolve the signal into its spectral components and usually the spectral component separation is performed using a diffractive element as shown in Figure 2-2. Subsequently, an imaging device (typically a semiconductor device such as a CMOS (complementary metal oxide semiconductor) image sensor) is used to perform the intensity measurement on the individual spectral components. The commercially available spectrometer is well capable of perform in this operation in a single enclosed device, but comes with a higher cost and maintenance is required.

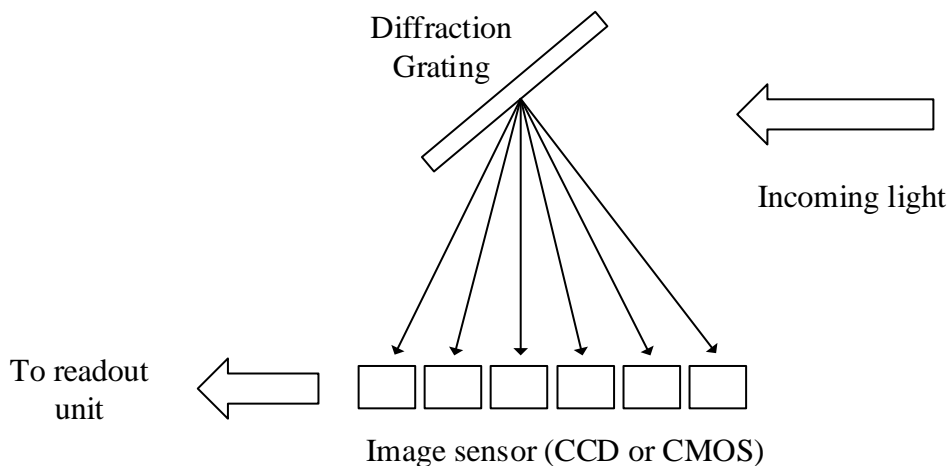


Figure 2-2 Spectra reading technique.

2.3 Optical sensor modulation techniques

2.3.1 Intensity

Intensity modulation is a widely used technique in the OFS due to its simplicity. Under this scheme, an intensity loss is induced by a change of the targeted parameter such as refractive index, temperature, pressure, and bending angle at the sensing region. The output intensity change is then translated into a measurement. The measurement setup can be realised using only inexpensive components such as the LED and an optical detector (e.g., the photodiode and phototransistor). The sensor can be implemented with different methods including the reflection method as shown in Figure 2-3. As an example of a reflection application, Borecki *et al.* demonstrated an OFS based on the intensity modulation [26]. The OFS working principle utilises the Fresnel reflection from the optical fibre's end (the sensor head). The sensor head's reflectivity is manipulated by the refractive index of the medium it is immersed in. This system is implemented using a monochromatic light source. The sensor head is mounted on a motorised arm that can move in a vertical direction to lift the sensor out of, or submerge the sensor into, a solution. The reflected light intensity is measured and recorded over the process of sensor head submerging, submersion, emerging and emergence. A small neural network is fed with information of the reflected power and the sensor head position over a specified period of time. The neural network produces the viscosity, surface tension, turbidity or rate of evaporation of the solution. This system is based on the PC for the neural network implementation. The computer interfaces with a control card and data acquisition (DAQ) card for motorised arm control and data acquisition.

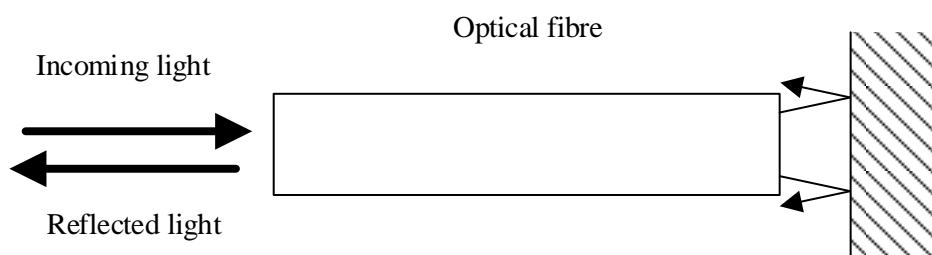


Figure 2-3 Intensity based OFS using a reflective surface.

2.3.2 Phase

The phase modulation technique depends on the phase difference between the two signals that results in constructive or destructive interference. The intensity change resulting from the interference can be detected using an imaging method or a simple photodiode. Under this technique, one signal is usually used as the reference signal and the other as the sensing signal.

The phase shift is produced by an optical path length change that is influenced by the physical distance, or the optical properties of, the medium the light is propagating. The common strategies to induce a phase shift include a moving surface, material dimension change under temperature or medium's effective index change [27]. One of the popular implementation methods, Fabry-Perot, utilises interference from reflections between two surfaces as shown in Figure 2-4. Poeggel *et al.* presented an optical fibre Fabry-Perot pressure sensor for In Vivo urodynamic analysis [22]. This sensor is designed to measure bladder and abdominal pressure in real time. The Fabry-Perot cavity is an all-glass design that was achieved by a double splicing operation. A single mode optical fibre and a capillary tube were spliced together first. The open end of the capillary tube is then spliced with a multimode fibre. The multimode fibre is then cut and thinned (polished and etched) to form a flexible thin surface. This creates the required closed cavity to sense external pressure.

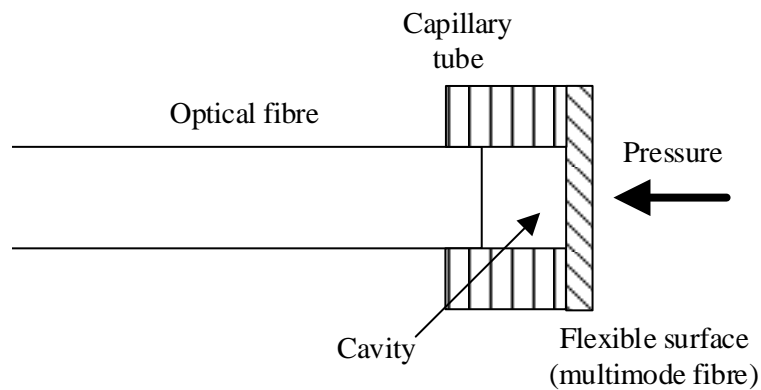


Figure 2-4 Fabry-Perot sensor for pressure sensing.

2.3.3 Wavelength

The wavelength modulation technique detects a change in the targeted parameter from the sensor output spectra [28]. A broadband or a tuneable light source is usually used to provide the needed spectra. For some applications, the sensor could be the light source itself (e.g., a laser and fluorescence material) that reacts to an environmental change such as temperature. To identify the change in the output spectra, the modulation technique usually requires a device that is capable to resolve the spectra. There are a variety of sensing mechanisms that fall under this modulation technique including the popular fibre Bragg grating (FBG) sensor as shown in the Figure 2-5 [27]. A periodic variation of the refractive index is created in the optical fibre and the higher refractive index region is formed using high intensity laser. The formed Bragg grating reflects a particular wavelength light back to the source and a different periodic spacing corresponds to a different wavelength. The sensor is primarily used to sense temperature or strain

that causes the periodic spacing change by a minute amount. This results in change of reflected wavelength.

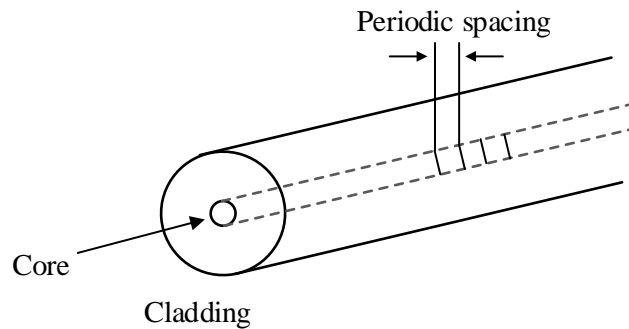


Figure 2-5 Fibre Bragg grating.

2.4 Surface Plasmon resonance (SPR) sensor

SPR is a well-established research phenomenon that has found many applications since the pioneering work by Kretschmann *et al.* [29]. The configuration demonstrates the optical excitation of surface Plasmons through attenuated total internal reflection through a prism. Today, the Kretschmann configuration (shown in Figure 2-6) is still a popular choice for the SPR study. The surface Plasmon is an electromagnetic excitation that exists at the interface between a metal and a dielectric layer and its propagation is confined to the perpendicular direction. SPR occurs when the excitation photon propagation constant is matched with the surface Plasmon and has the same transverse magnetic polarisation (which can be isolated using a polariser). During the phenomenon, part of the energy from the incident light is transferred to the surface Plasmons and results in a reduced intensity of the reflected light

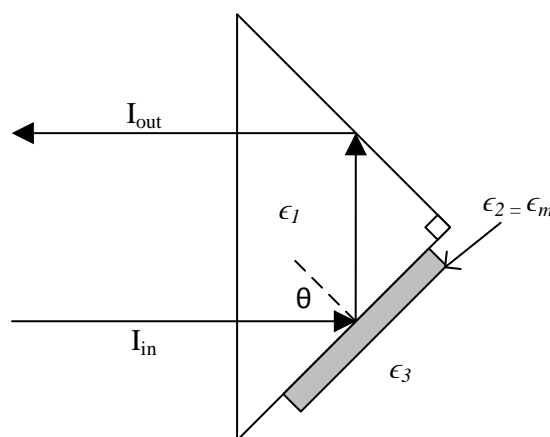


Figure 2-6 Kretschmann configuration. (I is incident light)

The resonance condition depends on the optical properties, the relative dielectric constant of the metal (ϵ_m), the relative dielectric constant of the prism that supports the interface (ϵ_1) and the medium within the SP evanescent field (ϵ_3). Both the metal and prism are fixed for sensing purpose and

therefore the resonance condition effectively depends on the sensing region's refractive index. The sensing region usually consists of a selective layer that only reacts to the desired parameter (i.e., chemical) whilst without the selective layer, the setup acts as a transducer that provides information of refractive index only. The selective layer only binds with specific molecule (targeted) and the refractive index change due to present of the targeted molecules is then detected using the SPR sensor. The underlying mechanism is identical for most of the sensor configurations. The propagating constant of the photon is required to match the surface Plasmon's propagating constant, k_{sp} [30].

$$\text{Re}(k_{SP}) \approx k_0 \text{Re} \left(\frac{\epsilon_m n_3^2}{\epsilon_m + n_3^2} \right)^{1/2} \quad (2.1)$$

Where $\epsilon_m = \epsilon_{mr} + i\epsilon_{mi}$ is the relative dielectric constants of the metal, n_3 is the refractive index of the measurand ($n_3^2 = \epsilon_3$) and $k_0 = 2\pi/\lambda$ is the free space wave number, where λ is the light wavelength. The matched propagation constant is founded by locating the decrease in reflected light intensity for different interrogation methods.

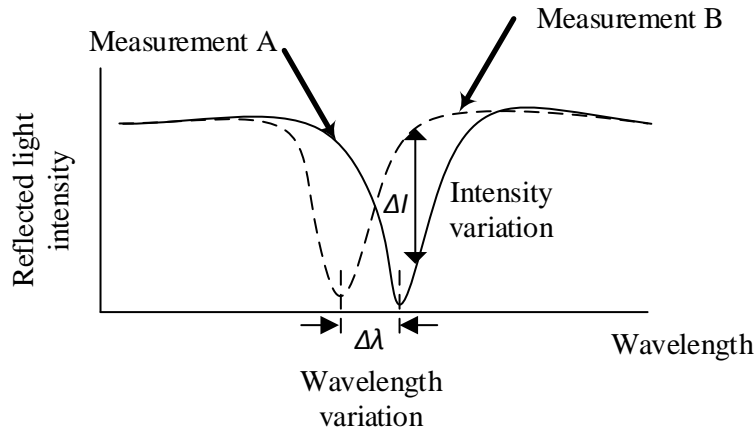


Figure 2-7 Typical responses for intensity and wavelength interrogation.

There are four main types of SPR interrogation method, these being (i) angular interrogation, (ii) spectral interrogation, (iii) intensity interrogation and, (iv) phase interrogation [31]. The Kretschmann configuration is one of the best examples for angular interrogation where the light wavelength is kept constant and incident angle is varied. The configuration for the wavelength interrogation keeps the incident angle constant and has a light source that covers broadband of wavelength shone at the interface. For spectral components that fulfil the resonance condition, the photons' energy is transferred to surface Plasmons and therefore result in a lower reflected intensity whilst the other spectral components remain unchanged. This results in a dip formed in the spectral output. For an intensity interrogation configuration, a monochromatic light source is used as the excitation source and its incident angle is fixed. Due to the fact that the sensing medium affects the resonance condition, the intensity of the reflection light is varied. Typical

responses for both intensity and wavelength interrogation are shown in Figure 2-7. The phase interrogation method relies on two signals (reference and sensing signals) to perform the interrogation. The SPR sensing implementation could be realised using both polarisations and the phase difference produced after Fresnel reflection [32]. The transverse electric polarisation signal and transverse magnetic polarisation signal are used as reference and sensing signals, respectively.

To find the matched propagation constant, an appropriate algorithm for particular interrogation method is required. For example, the centroid algorithm is a straightforward method and yet a very useful algorithm to locate the minimum of a curve for spectral interrogation implemented using imaging method [33]. In Pereira *et al.* work, an analysis region was extracted from the captured image during the sensing operation. The minimum intensity location in term of pixel number (which corresponded to the light wavelength), p_{min} , was determined by using the following equation:

$$p_{min} = \frac{\sum_{p=p_{initial}}^{p_{final}} p I_B - p I[p]}{\sum_{p=p_{initial}}^{p_{final}} I_B - I[p]} \quad (2.2)$$

Where $p_{initial}$ and p_{final} are the first and last pixel numbers of the analysis region, I_B is the base line (found using a predetermined specific solution), p is pixel number, and $I[p]$ is the intensity of the p^{th} pixel. This method assumes that the minimum reflection is the same position as centroid. There is a small discrepancy due to the asymmetric shape of the SPR dip. However, the discrepancy may not have large impact for certain applications if the sensing mechanism is only interested in its relative movements [30]. If the shape of the dip stays constant between measurements, the movement of the centroid is equal to the movement of the minimum and the offset would be cancelled. For other applications, improvements such as the weighted centroid method helps to minimise the discrepancy [30].

2.4.1 Interrogation methods for optical fibre based SPR

SPR based sensor systems are now available in the market and there are several manufacturers who are exploiting the Kretschmann configuration. Given the cost effectiveness and miniaturisation potential offered by the optical fibre, more people are willing to design a sensor system based on the optical fibre. The underlying principle of the SPR excitation is still the same but limited by some inherent constraints for the optical fibre based sensor (e.g., the optical fibres have fixed incident angles at the interface). Due to the fixed incident angle constraint, the angular interrogation method is not suited to an optical fibre based system. Spectral interrogation is one of the common methods to implement SPR optical fibre sensors that can be achieved by the

transmitted or back reflected signal measurement [34]. Intensity interrogation can be implemented using a monochromatic light source when its wavelength fits with the resonant conditions that include incidence angles. This technique implementation is simple yet low cost. The phase interrogation is a less common method to be implemented. The requirement for the more complex optical instruments and data processing methodologies can restrict its application.

2.5 Optical fibre system architecture review

In light of the fact that the optical fibre based SPR sensor is usually implemented using intensity or wavelength interrogation, this section presents examples of OFS system implementations that are based on intensity and wavelength interrogation. This review focuses on the system build and the components used. The following section is then dedicated to the implementation of a portable OFS system.

One example of an OFS system based on intensity modulation technique was reported by Di *et al.* The authors reported an electric current measurement using a fiber-optic curvature sensor system that makes use of intensity modulation [35]. The system utilised the LED and photodiode as a fixed current light source and optical receiver. The optical receiver output was sampled using a DAQ and PC. The system utilised linear regression to calibrate the system and depended on the use of the Wavescan software for live data plotting. Krehel *et al.* presented an OFS for respiratory monitoring based on intensity interrogation [36]. The authors utilised a red LED as the light source, a photodiode as optical receiver and a data logger to sample data. The data analysis was performed using MATLAB[®] and a PC.

For a wavelength interrogation example, Beniamino Sciacca *et al.* reported a SPR based on OFS to detect apolipoprotein E protein using wavelength interrogation [37]. The sensing mechanism made use of the re-scattering of light from the surface Plasmons (from a high surface roughness interface). The sensing architecture utilised a white LED as the light source and a spectrometer as the optical receiver. The data analysis was performed using MATLAB[®]. Yanasea *et al.* reported an optical fiber SPR sensor for living cell activation using wavelength interrogation [38]. The SPR sensor system was composed of a white LED as light source and a spectrometer as the optical receiver.

The aforesaid examples required the use of PC or rather expensive and large instruments as part of the OFS system. PCs are widely used as the processing unit in an OFS system due to the vendor provided user friendly software and the abundance of compatible components. Such systems are accurate but tend to be costly and stationary, which shows the necessity for a portable and flexible sensor system for use in a remote area or a field application (without the use of a PC). With

advances in circuit design and miniaturisation of different computing devices, different system realisations can be proposed and studied to develop a low cost and portable solution.

2.6 Portable optical fibre sensor system

There has been a growing demand for portable systems that are simple to use, low-cost and provide rapid detection for applications related to point of care (POC) and environmental monitoring. These system made use of components that available for consumers or integrated platforms to realise the portable OFS system.

Off-the-shelf optoelectronic components and microcontrollers have enabled the realisation of the OFS system according to a specific application requirements. Laskar *et al.* reported an OFS system based on the Atmel® ATmega32 microcontroller for moisture sensing in transformer oil [39]. A portion of the optical fibre had its outer coating stripped away and bent into a fixed radius of curvature. This was to induce power loss if a higher refractive index medium covered the bare part. The transmitted optical power was measured using a light dependent resistor and a laser diode was used as the light source. The analysis was performed on-board through a light weight artificial neural network. Bram Van Hoe *et al.* reported an integrated OFS system based on the Atmel® ATmega644 microcontroller for fibre Bragg grating (FBG) [40]. The microcontroller controlled a current source to drive the laser diode and manipulate or fix the laser wavelength through control of the driving current. A photodetector was used to sample the FBG output at a fixed wavelength to achieve intensity modulation or wavelength sweep to achieve wavelength modulation. The system could be connected using an Android software app and read out of the data. The authors took one step further to integrate the opto-electrical components in a thin, flexible package using a multi-step process.

The smartphone is a highly integrated device that is provided with numerous sensors and easily connected to other devices. It is a good platform for optical sensors as it normally has a high resolution imaging sensor, high intensity light source and powerful processing unit. This device can however be easily identified as a valuable item. The ease of creating an application also suits the smartphone as a user friendly platform for an OFS system [41]. Liu *et al.* demonstrated a fibre optic SPR biosensor based on smartphone platform [41]. The system utilised three channels: measurement, control and reference channels to perform the intensity interrogation. The reference channel was to compensate for intensity fluctuation of the smartphone's flash light and control channel was for a sensing element kept in a control environment. The SPR sensor was an extrinsic sensor and a smartphone case was used to connect the lead in and lead out fibres of the OFS to the smartphone camera and LED to ensure the optical alignment. Lenses were used to provide efficient coupling. A narrow-band filter was placed in front of the flash unit to provide a

near monochromatic light output. A self-developed Android app was used for the camera exposure control, image extraction, data processing, data storage, and data transmission. Bremer *et al.* reported a single channel SPR based on OFS-smartphone system [42]. The system utilised a smartphone as light source and imaging device. The proposed system relied on a PC to process the data. The optical fibre coupling was implemented using a 45° angled fibre end for both input and output and a diffraction grating placed before the camera resolved the spectra.

2.7 Summary

This chapter has developed an understanding of the SPR based OFS system that builds on related literature. The chapter started with a general discussion of key sensor system components including the optical fibre, light source and optical detector for an OFS system. This was to understand the key characteristics that are commonly used in the system realisation. Different OFS modulation techniques were then explored along with their corresponding system needs discussed. The sensing mechanism, SPR, in current system and its interrogation techniques were discussed. The related interrogation techniques were identified and implementation examples reviewed to identify the key components that are required. Review of associated literature further discussed the need for a portable sensor system architecture for such sensing method.

Chapter 3 Field programmable gate array

3.1 Introduction

Prolific use of the software programmed processor (microcontroller (μC), microprocessor (μP) and digital signal processor (DSP)) has been driven by the ability of programming its operation for different applications and also to reprogram the same device for different target applications. This ability allows a variety of applications to be implemented using the same device in an economical way. The μC has been favoured for extended periods of operation due to its power consumption control (e.g. through clock frequency adjustment). Some modern μC designs have adopted a multi-core architecture for concurrent operations. However, the software programmed processor is not energy efficient for tasks that have complex data dependency and large amount of data [43]. A dedicated hardware for implementing computational-intensive algorithms (parallel operations) shows the improvement especially if the data dependency is complex. Dedicated hardware can be implemented by an application specific integrated circuit (ASIC) or programmable logic device (PLD). ASICs have a fixed functionality that is set during manufacturing. However, the PLD, such as the field programmable gate array (FPGA) and complex programmable logic device (CPLD), is hardware programmable/configurable and reprogrammable/reconfigurable by the user for a particular requirement. The PLD is a low-cost device to obtain a customised digital logic function and requires only minimal time to create a functioning device compared to developing an ASIC solution. The designer is able to describe their digital circuitry using a hardware description language (HDL) such as VHDL (VHSIC (very high speed integrated circuit) HDL) or the Verilog HDL before synthesising the description to target a particular device. The FPGA is of particular interest in this work due to its architecture and ability to implement high-speed and complex digital signal processing (DSP) operations. Today, FPGAs have specific built-in DSP blocks to support typical DSP operations and some devices also provide an integrated analogue-to-digital converter (ADC) that supports sensor data acquisition for signal analysis. The FPGA can also be designed at physical level without any design abstraction that can be important guarantee the timing of the operations in high speed, time sensitive applications.

This chapter is structured as follows. Section 3.2 begins with a discussion into FPGA basics, focusing on the technology behind the reconfigurable device operation. The programming technology and FPGA design flow are summarised to provide the basics into how to configure the device. Section 3.3 discusses the common components that can be built using the FPGA and the available on-chip resources. Section 3.4 reviews the application of the FPGA in a sensor system. Section 3.5 discusses the FPGA in current use and section 3.6 summarises this chapter.

3.2 FPGA basics

3.2.1 Architecture of the FPGA

In an ASIC, the fixed function is built from wiring together a number of digital logic gates. The development of the ASIC is costly and time consuming, but a single-unit cost during manufacture is low if the ASIC design is fabricated in high volume. The FPGA, as an alternative, is comprised of a matrix of configurable logic blocks (CLBs) that are connected together via programmable interconnect. The pattern is known as “island” style with large blocks of logic grouped together and connected to other blocks. This allows rapid and economic development of new designs in limited quantities for applications such as instrumentation. The FPGA uses input/output (I/O) blocks to access external signals via the device pins as depicted in Figure 3-1. The figure depicts a simplified FPGA architecture. The CLB consists of fixed number of look-up tables (LUTs) that store Boolean functions to perform logic operations and functions as the basic building block of the FPGA. The CLB also incorporates other functions such as small memory elements referred as distributed RAM. The exact build of the CLB is different from one vendor to another and even between different generations of FPGAs from the same vendor.

Figure 3-1 shows a simple example in which each LUT is paired with a flip-flop to facilitate pipelined circuit designs. This ensures signal integrity is maintained with high clock frequency operation. The flip-flop can be bypassed through a multiplexer depending on user requirements. To implement some functions that require multiple bits of data to be processed, such as adders and multipliers, multiple LUTs need to be combined using carry chains. Carry chains are dedicated resources to link LUTs. However, a carry chain is not optimised to implement wide input or output arithmetic functions as large propagation delays add up and require a large amount of the available resources. Dedicated DSP slices are more suitable to implement this type of function. The majority of FPGAs are equipped with dedicated blocks included such as the ADC and block random access memory (BRAM) to have more efficient function implementation.

The modern FPGA is capable of implementing an entire processor, multiple processors or a complete system on chip (SoC). Such complex circuitry is required to be formed using large

number of logic gates that span over a large physical area on the FPGA layout. Flexible connections, the programmable interconnect, are used to make the links between different parts of a design. These interconnects are also responsible for connecting CLBs with I/O blocks to interface with external circuitry. All these resources can be reconfigured according to application needs using VHDL or Verilog HDL.

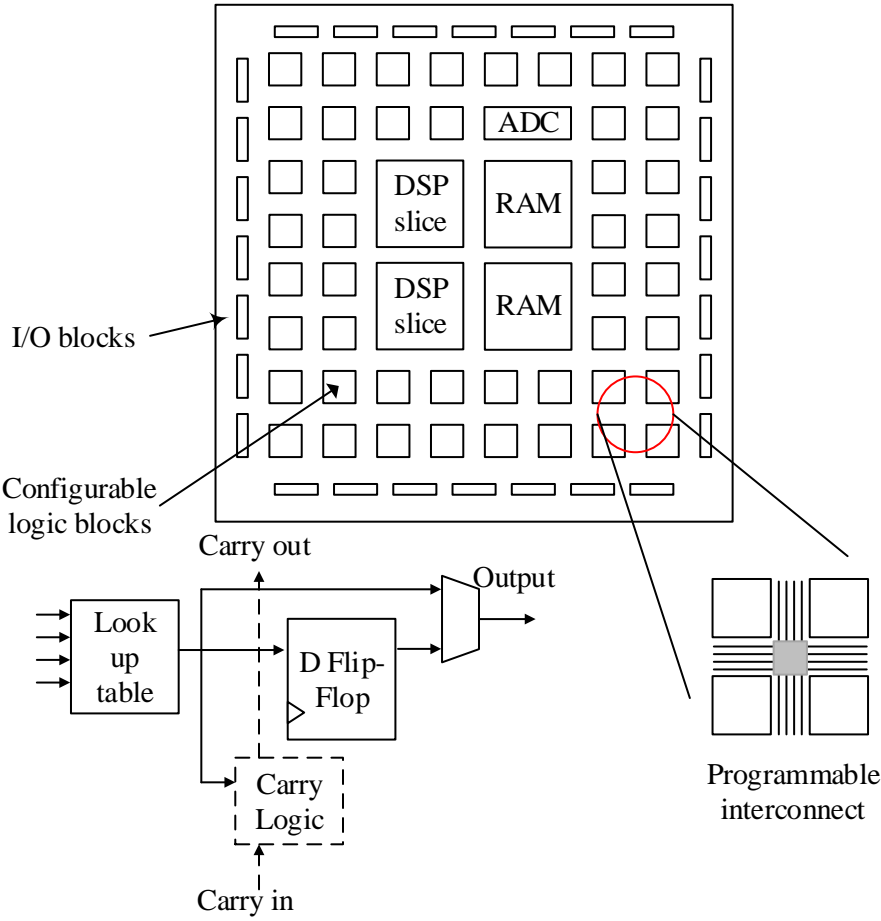


Figure 3-1 Simplified FPGA architecture.

3.2.2 Programming technology

There are two main FPGA technologies available, static RAM (SRAM) which is volatile memory and flash memory, which is non-volatile, to store the circuit configuration [44]. They have different characteristics that suit different needs. Most commercially available FPGAs are based on SRAM technology including Xilinx® and Altera®. The SRAM-based programming technology is the dominant technology as it is relatively more mature than flash memory technology. The SRAM cells are volatile in nature and therefore the FPGA configuration needs to be loaded each time power is applied to the device. In addition, external permanent storage is required to hold the configuration data. On the other hand, the flash memory based FPGA is non-

volatile. It requires no external configuration device to load the configuration at start-up as the configuration is stored within the FPGA itself and can be rebooted in short time after an event such as a power failure. It also consumes less area per cell as it requires less components for single logic cell construction. However, the device cannot be reconfigured an infinite number of times. For current technology, the flash based FPGA can provide competitive power consumption compared to a conventional μC , but only SRAM-based FPGAs can meet the requirements imposed by intensive computational application [45]. However, the high performance comes at a cost of high power consumption.

3.2.3 FPGA Design flow

The FPGA provides a reconfigurable architecture that is easily configured for a range of different circuits. The designer however needs to go through a series of steps, which commonly known as a design flow, to implement the final configured device [46]. Simulation can be undertaken to verify the design operation at several points in the design development [47]. Details of a particular design flow would be different between vendors and different tools is used. The vendor tools would be integrated into an integrated development environment (IDE). A simplified design flow is shown in Figure 3-2.

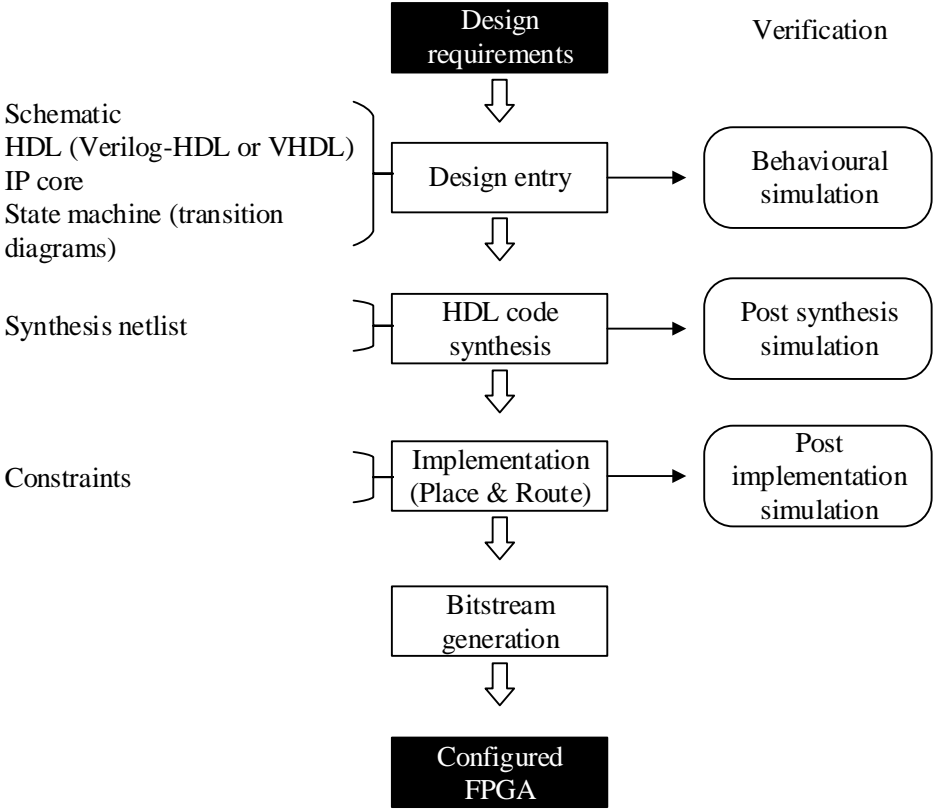


Figure 3-2 Simplified FPGA design flow.

Initially, the user is required to have a general understanding of their application needs. These needs are then translated into a design in the form of a HDL design description, a circuit schematic, a provided IP core or a finite state machine (FSM) state transition diagram. The HDL design entry method is the common design tool as it is independent of vendor tools. This allows the described components to be reused in most different configurable devices if no specific part (e.g., the Xilinx[®] MicroBlaze) inference is used.

An efficient HDL design should be created to utilise the available FPGA hardware resources and be structured as a hierarchical design [48]. The hierarchical design would break a large and complex design into small and manageable parts. The design is then verified using behavioural simulation to ensure it functions as intended. At this stage, the design can be simulated as a module and the design is not architecture-specific unless a part from specific vendor is inferred. A test bench (with user defined stimulus input) is used in the simulation to examine the input, internal and output signals. The design is then synthesised for a target FPGA into a netlist that uses the available components in the FPGA to satisfy the behaviour described by the HDL description. Some vendor tools provide logic optimisation for I/O pin, area or timing requirements. The synthesised design can be simulated to verify if the functional requirements were met and behaves as expected. Next, the synthesised netlist would undergo place and route (implementation) that place the required components in suitable location in the targeted FPGA. The components are then connected using programmable interconnect by the tool. The designer can specify the constraints such as pin assignment and timing requirements (e.g., clock period and maximum signal delays). After implementation, a timing simulation can be performed to ensure the additional signal delays due to logic and interconnect propagation delays are within the timing requirements. The post implementation timing simulation is the closest emulation to the configured FPGA performance, but may require a longer simulation time when compared to behavioural simulation. During the design stage, the designer can perform power and utilisation analysis to optimise their design. Finally, the designer can generate a bit stream for the targeted FPGA. The designer can then configure the targeted FPGA with the bit stream or download it into external non-volatile memory for SRAM based FPGAs.

3.2.4 Commercially available FPGAs and CPLDs

Over the last couple of decades, FPGA functionality has changed from simple glue logic to becoming a major component within a complex system [49]. Different FPGA products are available today for different computational needs. The products benefit from both standardisation of the system components and the ability to customise for the needed application. For different products, the provided toolchain by the vendor also varies. Higher abstraction programming tools are provided for higher performance FPGA. There exists a few major vendors such as Xilinx[®],

Intel®, Lattice Semiconductor and Microsemi. Both Xilinx® and Intel® are the main FPGA manufacturers that cater for applications from low to highest computational power. The following tables (3-1, 3-2, 3-3 and 3-4) list the products from commercial vendors (only FPGAs and CPLDs are identified whilst SoC (system on a chip) devices have been omitted). The descriptions were extracted from both the vendor website and product description.

Table 3-1 Xilinx® products.

Type	Product family	Brief description
FPGA	Spartan®-7	Suited for low cost, lowest power, and high I/O performance application.
FPGA	Artix®-7	Suited for low power, high DSP and logic throughput applications.
FPGA	Kintex®-7	Optimised for best price-performance.
FPGA	Virtex®7	Provides highest system performance and capacity.
CPLD	CoolRunner™-II	High-performance and low-power CPLD

Table 3-2 Intel® (previously Altera) products.

Type	Product family	Brief description
FPGA	Stratix®	Combines high density and high performance with a rich feature set to enable more functions and maximise system bandwidth
FPGA	Arria®	Delivers optimal performance and power efficiency in the midrange.
FPGA	Cyclone®	To meet low-power, cost-sensitive design needs
CPLD	MAX®	Delivers advanced processing capabilities in a low-cost, instant-on, small form factor programmable logic device.

Table 3-3 Lattice Semiconductor products.

Type	Product family	Brief description
FPGA	ECP	Connectivity & Acceleration
FPGA	iCE	World's Smallest FPGAs.
FPGA	Mach (part of the family is flash based)	Bridging & Expansion FPGAs.

Table 3-4 Microsemi products.

Type	Product family	Brief description
FPGA	PolarFire	Cost-optimised, lowest power, mid-range density FPGAs
FPGA	IGLOO2	Feature rich low density devices.
FPGA	ProASIC3 FPGA	Low-power FPGAs for demanding (environment) high-volume applications.
FPGA	IGLOO	The industry's low-power FPGAs.

3.3 Auxiliary On-Chip Components

The logic resources (i.e., the LUTs) within the FPGA are technically sufficient to implement a wide range of circuits. However, to have better usage of the FPGA fabric area, different custom block components such as DSP slices and block RAM (BRAM), have been included into the FPGA's fabric [50]. To interact with an analog input signal, the FPGA vendor may also have included an ADC. In some high performance FPGAs, traditional processing units (mostly based on the ARM (Advanced RISC Machine) core) are included to provide software flexibility.

3.3.1 Built-in ADC

The ADC is the key component to enable the digital domain to interact with analog signals. Some vendors have incorporated dedicated ADCs into the FPGA. The integration minimises both system footprint and system power usage. The built-in ADC also usually fits the common flexibility theme within the programmable fabric and large I/O counts. The design can accommodate large numbers of sensors and different needs.

3.3.2 DSP slice

Modern FPGAs are becoming more attractive for digital signal processing tasks using dedicated DSP slices that can be found within the programmable fabric. The DSP slice, combined with programmable logic, is able to support different algorithm needs within a customised design. The embedded DSP slice provides better performance and a more efficient arithmetic operation implementation than is possible with the LUT. If a wide arithmetic operation such as multiplication or addition is implemented using the LUTs, a large number of LUTs and general purpose routing are going to be used. This consumes more energy than the dedicated DSP slices. The functionality provided by DSP slices are different from one vendor to another and also from one generation to another of a particular device.

3.3.3 Block RAM

A BRAM is a dedicated memory element that can typically hold few kilobytes of data. It is common to have a few hundred instances of BRAM within a FPGA. It is placed between the CLBs to provide efficient read and write operations with minimal routing. This positions the BRAM as an intermediate density memory element where lower density and higher density are provided by LUTs and external RAM respectively.

3.3.4 IP core (common use component)

The FPGA is a powerful device to create a customised core, but it requires significantly more design effort than software programmed processor approach. However, certain functions/components are commonly found in different designs repeatedly. These reusable functions are commonly generated as an IP core that is parameterisable. The usefulness of IP cores for HDL is similar to libraries for software languages. IP cores are available in both open source and commercial forms, and the functionality could range from simple storage functions to entire microprocessors. The IP core could be a hard IP core that depends on dedicated hardware components embedded within the FPGA, or a soft IP core that utilises programmable logic for implementation. It is common to find one or more microprocessors within a modern high performance FPGA. For example, Xilinx[®] latest Zynq series incorporates an embedded dual core ARM Cortex-A9.

3.4 Applications of the FPGA in a sensor system

Currently, sensor systems demand more computational resources due to an increase in the requirements for different applications including security, machine learning and signal processing. A sensor system traditionally is implemented using a sequential architecture of a software programmed processor and limited to provide simple functions. To address these intensive computational challenges, it usually results in a system that depends on an external computing device for the processing power. Recently, some systems have adopted the FPGA, given the concurrent nature of the configurable devices hardware, to mainly target parallelisable algorithms and tasks [45]. For example, when the application requires the implementation of a standalone device, this results in smaller allocation of the battery resources for the data transmission. [51]

The use of the FPGA has shown that it can reduce system latency through a lower level interface as reported by Pereira *et al.* [33]. In the work, the FPGA was used to implement control, acquisition and data processing for a SPR sensor system. The SPR device was implemented using

wavelength interrogation through an imaging method. The image sensor's parameters including integration time, the gain of the ADC and the desired area for image acquisition are configured using an SPI interface. The selection of the sensing areas benefits multi-spot sensing. However, the communication with other components (such as a liquid crystal display (LCD)) using a lower level description also means that is not as convenient as with a processor that would have available to existing high-level library functions for common components.

For a machine learning application, the FPGA has been useful in data pre-processing, feature extraction, and the algorithm itself [52, 53]. The FPGA is commonly used in real time applications given its superior parallel processing capability. Hervás *et al.* utilised a Zynq FPGA to provide audio input feature extraction (Mel Frequency Cepstral Coefficients) and machine learning (Gaussian Mixture Model) [52]. The extraction of 13 features was performed using programmable logic (PL) part of FPGA and the output features were feed into the classifier. This was used to ensure that the execution time is fit the requirement. Martinez-Figueroa *et al.* utilised a Cyclone IVE FPGA to compute the statistics values including, mean, variance, skewness and kurtosis from signals [53]. These statistic values are then feed into an artificial neural network (which was also implemented using the FPGA) as features.

Some machine learning depends on a large number of I/O and requires a computationally demanding processing algorithm. The FPGA is a well-suited to handle such applications as it commonly equipped with large number of I/O combined with its concurrent nature. Shi *et al.* implemented a gas identification system using an FPGA as a processing core that received 8 sensor inputs [54]. The system employed a committee machine (CM) classifier which comprised of five different classifiers: k-nearest neighbour (kNN), multilayer perceptron (MLP), radial basis function (RBF), Gaussian mixture model (GMM), and probabilistic principal component analysis (PPCA). The authors evaluated the implementation of the CM using different FPGAs. The author found that if a less powerful FPGA was used, the processing operations were required to be partitioned into a number of stages. The partition was implemented using the dynamic reconfiguration. This function is common to be found in FPGAs and suited for applications that requires intensive processing or have different needs in different time or situation.

The FPGA is also a useful intermediate stage for the designer toward an ASIC implementation given the expensive development cost for every design iteration in ASIC fabrication. Viseh *et al.* utilised the FPGA as a proof of concept of an on-board processor for a tongue drive system [51]. The designer analysed the system on different aspects such as the number of training samples, the data-path word width and bit resolution impact on the system accuracy and hardware complexity. The processing on board design successfully reduce the data transmission need by a factor of 1500x, from 12 kb/s to ~8b/s

FPGAs have also proven to be useful in timing critical applications. Optical sensor system designers have utilised the concurrent nature of FPGA to modulate the light source and demodulate the measured intensity signal at the same time. Wang *et al.* and Zhang *et al.* utilised the FPGA combined with a Fabry-Perot (FFP) tuneable filter to tune the light source wavelength and acquired the intensity output in parallel [3, 55]. Yao *et al.* manipulated laser wavelength using distributed feedback laser instead [56]. Both schemes produced a high wavelength resolution scanning system. The signal processing utilised in both schemes were also implemented using the FPGA. These examples have very stringent requirement on timing and the FPGA is suited for this type of application.

For sensor communication, the FPGA has been used for a data compression application [57]. Kaddachi *et al.* studied the FPGA as a co-processor for a wireless camera sensor node to reduce the energy consumption due to data transmission. The study showed a substantial reduction of processing time and energy consumption compared to the original node.

3.5 Digilent® Arty FPGA development board

The development board used in this project was the Digilent® Arty 7 Development Board with Xilinx® Artix-7 (part no: XC7A35TICSG324-1L) FPGA. It is a ready-to-use development platform as shown in Figure 3-3. Xilinx® provide different tools for designing with their FPGAs. They can be configured through the following approaches.

1. Configure the FPGA by creating components through conventional HDL design method.
2. Embedding a processor (e.g., the MicroBlaze 32-bit RISC processor) and required components within the FPGA and program the processor using C. This can be implemented using IP Integrator within the Vivado IDE and creating a C program using the Software Development Kit (SDK) to perform the required functions.
3. Writing a program in C and creating the HDL code using Vivado HLS (High Level Synthesis).
4. Using HDL Coder tool within MATLAB® software to create the HDL code and this also can be achieved using Xilinx® System Generator.

In the reported work, the circuit design created using the conventional HDL (VHDL) design method. This configuration method was chosen as the produced design is transferable to other devices and the VHDL language required details to be explicitly stated (reduced the potential for an ambiguous design description). The following subsections focus on the FPGA features that are crucial for the current system operation. The features are mostly available across the Xilinx® series 7 FPGA portfolio.

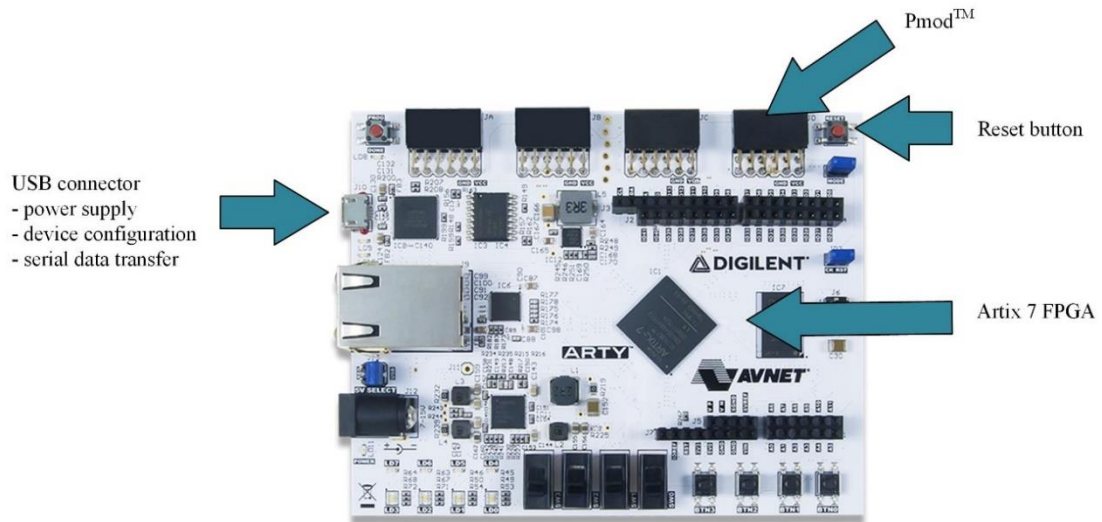


Figure 3-3 Arty FPGA development board.

3.5.1 Xilinx® ADC

The built-in ADC within the Artix-7 FPGA is Xilinx® ADC (XADC) as depicted in Figure 3-4. It can access up to 17 external analog inputs and monitors on-chip parameters including temperature and supply voltage levels. It is a dual 12-bit 1MSPS analog-to-digital converter. The ADC readings are stored in status registers whilst the control registers store the configuration settings of the XADC. Both control registers and status registers can be accessed from the FPGA fabric. The control registers can be also initialised during the design process. The XADC can be configured into different sampling modes included event driven operation, continuous sequence sampling and single pass mode through the control registers.

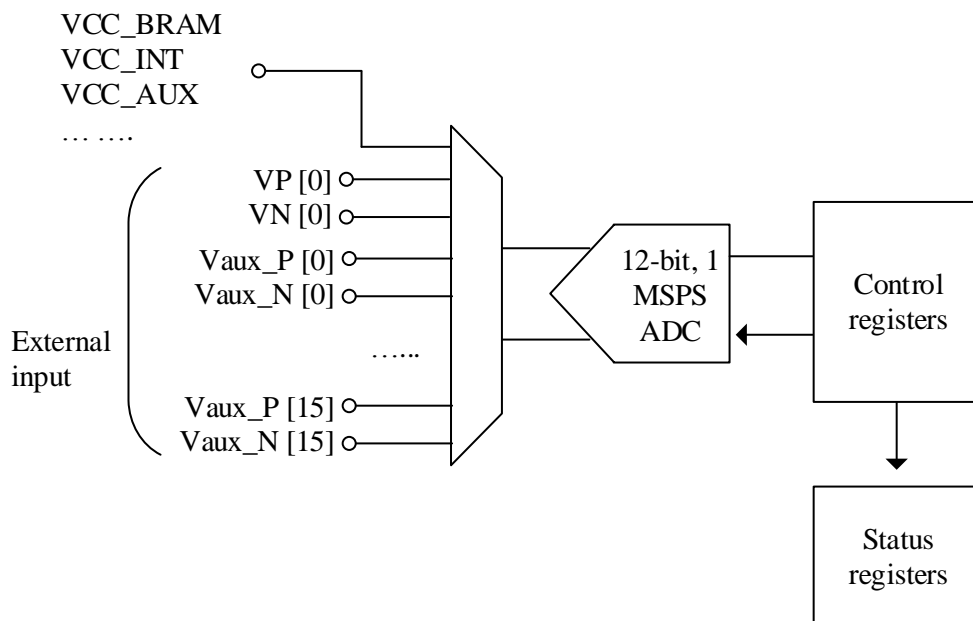


Figure 3-4 Simplified XADC block diagram.

To sample a voltage level across the desired measuring points, a capacitor (mounted on the Arty board) is used to sample the voltage across the two analog inputs. The capacitor circuit needs a transition time to settle to a stable voltage level that fulfils the required accuracy. Given that the XADC is 12 bits resolution, the final voltage error is required less than one half of the least significant bit (LSB) of the 12-bit value [58]. The acquisition time depends on the circuitry used due to its unique transient response. Proper data acquisition time must be assigned to avoid signal transient sampling. The conversion phase can be initiated once the acquisition is done. The XADC allows the acquisition of the next channel to start during the current conversion as there is a separate track-and-hold amplifier.

3.5.2 DSP slices - DSP48E1

The DSP slices, DSP48E1, are embedded within the Artix-7 FPGA’s programmable fabric. A DSP slice consists of three stages including pre-adder, multiplier and accumulator/logic unit. These stages can be pipelined using registers. The DSP slices are configurable as the other components in the FPGA. The design can just utilise part of the DSP functionality and configure it to perform task combination such as multiply, multiply& accumulate, add/subtract & multiply, and other combinations as per the user needs. Accumulation is performed using a dedicated link between the adjacent components to form an adder cascade. Accumulation using the dedicated link is efficient as it does not consumes the general routing path.

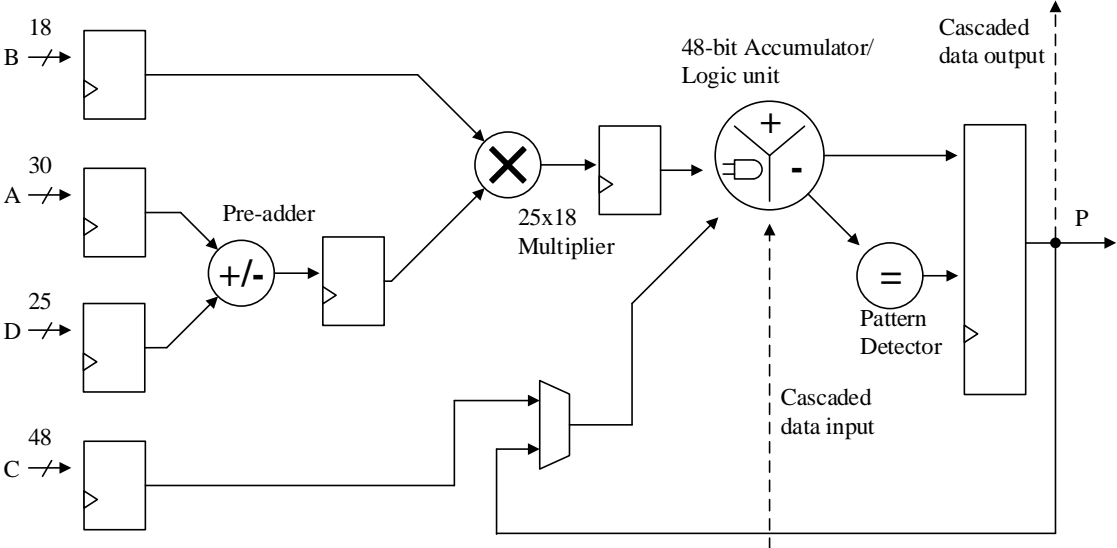


Figure 3-5 Simplified DSP48E1 architecture.

3.5.3 7-series FPGAs Block RAM

A single Block RAM (BRAM) has a fixed amount of memory but it can be configured into different data bus width (the memory depth is also changed as the memory size is fixed) for more

efficient use. To cater for a larger memory requirement, multiple BRAMs can be stacked together to form a larger memory unit. The BRAM can be connected in parallel to increase the data bit width. The BRAM can be configured into dual ports to provide the access to the same memory for different components at the same time. The BRAM can also be configured as a dedicated FIFO if the large amount of memory is needed for the FIFO. The configuration requires no extra logic resource usage. In a high performance FPGA family, the FIFO configuration can support dual clocks for dual inputs. This eases the data transfer between different clock domains.

3.6 Summary

The early part of this chapter covered the FPGA basics. It started with the FPGA architecture identifying the key components of the programmable fabric that included the configurable logic block and programmable interconnect. This developed the understanding on how certain digital logic designs can be accomplished using the FPGA. To obtain the final circuitry, the designer is required go through a series of steps (the design flow) to create, simulate, verify, synthesise and translate the design into a circuitry for a targeted FPGA. A list of vendor products were tabulated to summarise their target application. Modern FPGAs have included different dedicated hardware components for computational efficiency and better use of programmable fabric space. To interact with an analogue signal, FPGAs have been embedded with an ADC. The available components and the architecture of a particular FPGA may vary for different generation of a particular device [50].

To have better design reusability, the common functionality for both dedicated hardware and those built using logic resources are generated as IP cores. FPGAs have been employed in different applications to implement parallel algorithms within sensor systems in order to process data efficiently. This facilitates the implementation of the real time application as the use of FPGA help to satisfy the timing requirement. Some reported works have implemented machine learning using the FPGA for different sensor systems applications. These literature examples demonstrate the processing, control and programmable capability posed by the FPGA. The currently used development board, the Digilent® Arty 7 Development Board, was introduced to develop in-depth understanding of its available features.

Chapter 4 k-nearest neighbour classification

4.1 Introduction

Machine learning is a process taking in raw data to extract information, for example, to classify data through its features by using a well understood training set. This process is similar to how humans perceive things. It has gained popularity in various fields that include machine vision, voice recognition and search engines. Machine learning can be categorised into two types of learning algorithms that include the eager learning and lazy learning algorithms. The eager learning algorithm is a two-step process consisting of the training phase and the classification phase [59]. Firstly, a well identified dataset (the training set) is used to create and validate a model. Next, new raw data is fed to the model to be classified. On the other hand, the lazy learning algorithm that includes the k-nearest neighbour (kNN) does not require the creation of a model from a training set for classification but instead needs to memorise the training set. The lazy learning algorithm is required to go through all the data within the data set for every classification.

Machine learning can also be categorised as being either supervised or unsupervised learning. The unsupervised learning algorithm makes an estimation or prediction on structure and pattern from datasets without any data labelling. This algorithm is useful in situations where the data is too complex or too large for humans to make sense of and where patterns or structures could be found by running the algorithm on the data in order to ‘teach’ the human. Clustering is a common tool in unsupervised learning to explore data structures or patterns. Supervised learning requires a training set with data labelling that is performed by a human. Regression, neural networks and classification are common tools for supervised learning algorithms.

This chapter is structured as follows. Section 4.2 discusses the kNN data classification and its variant. Section 4.3 discusses methods to improve the kNN in term of computational speed and classification accuracy. Section 4.4 discusses the kNN implementation based on an embedded system and its architecture. Section 4.5 summarises this chapter.

4.2 k-nearest neighbour

k-nearest neighbour (kNN) classification has been implemented in many areas of science and engineering due to its performance. The kNN's prediction is based on the majority of k-nearest neighbours' classes. The kNN is able to achieve low classification error that is not worse than twice the Bayes rate provided the training set is large enough [60]. Bayes error rate is commonly used as a classification benchmark. kNN classification is conceptually simple and can be implemented in a straightforward way. It is a lazy learner algorithm that does not make any assumptions about the underlying data distribution. The process to attain a result (as illustrated in Figure 4-1) also provides a simple graphic for user. Visualisation of the algorithm operation is simple and intuitive rather than confusing with multiple layers of neural network that are hidden.

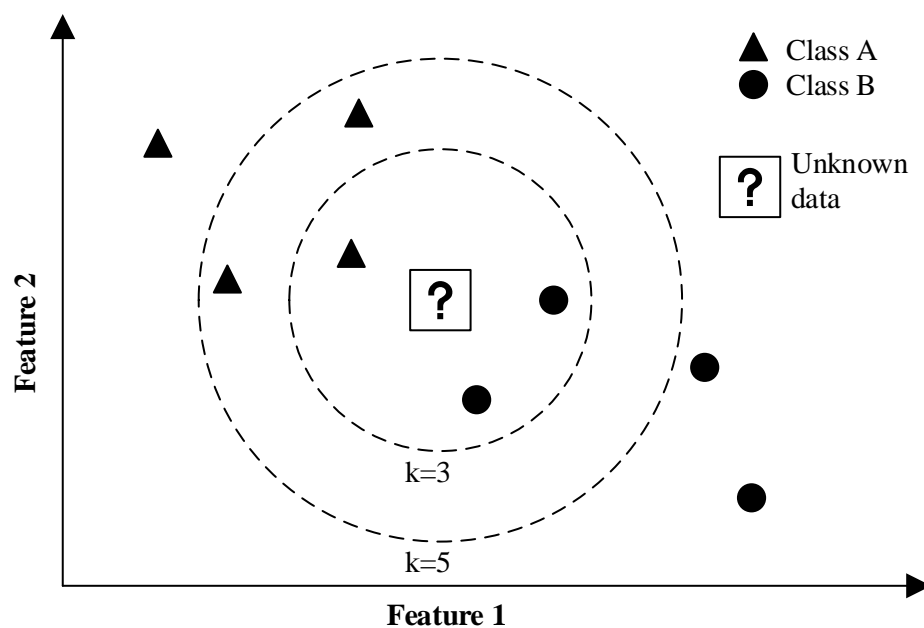


Figure 4-1 kNN classification example for $k = 3$ and $k = 5$ (boxed question mark is the new unknown data).

Figure 4-1 depicts an example of the kNN classification process of a new unknown data for different scenarios, $k = 3$ and $k = 5$ using the Euclidean metric. The training sets are filled symbols while the new unknown data is a boxed question mark. There are two classes, A and B, within the training set and the instances have 2 features represented by the x and y axis. The dashed circle encloses the k-nearest neighbour for both cases and shows that the outcome can be different with different k parameter values. The test data is assigned to class B when $k = 3$ while class A when $k = 5$. Too large a value of k can change the locality of the estimation as farther instances are included to determine the class assignment if the amount of the training instances are comparable with the k parameter. However, given large numbers of training instances, setting the parameter k higher than 1 will decrease influence of noisy data on the algorithm and smooth the decision boundary most of the time [61]. This can lead to better generalisation. This

classification could fail if data size is imbalanced and causes variation of density in the data distribution [62]. This is due to the larger sample size usually having a higher density that normally outvotes the smaller sample size in the border area. To translate this into an executable program, the kNN's algorithm is implemented as the following pseudocode.

k-nearest neighbour (kNN) algorithm

Input: training set (T), number of nearest neighbour (k), test data (X)

Output: class prediction

Algorithm pseudocode:

```
k-nearest neighbour (kNN)
BEGIN

Distance between all vector from training set and the test data
are measured;
Rank the distance;
Find the k number of smallest distance (k closest instances);
Record the label of the k closest instances;
The highest count of label is used as class prediction.

END
```

From the algorithm, the test data is assigned a class label using the k most similar training data. The similarity between data is determined using distance and the distance metric is not specified. Depending on the application, a different metric may be used including Manhattan, Hamming, Euclidean [63]. Euclidean and Manhattan metric measure distances in the spatial term while Hamming metric is commonly used in digital communication to detect changes in a message. The Hamming distance is defined as number of different symbols in a message. The Manhattan distance accumulate differences in all dimensions. To calculate the Euclidean distance, the square of the difference in every dimension is accumulated and then the sum undergoes square root operation to produce the distance as shown in the equation below.

$$d(a,b) = \left[\sum_{n=1}^D (a_n - b_n)^2 \right]^{1/2} \quad (4.1)$$

Where d is the distance between the test instance (a) with the stored training instance (b), D is the total number of features.

There are different ways for kNN to reach a prediction such as consensual nearest neighbour, weighted in feature or distance. A modification would be made to the algorithm for each of these variants. For consensual nearest neighbour, the prediction on the test vector is only made if all

the nearest neighbours belong to the same class and that particular class would be assigned to the test vector [64]. For the distance-weighted variant, the classification process still depends on the k nearest neighbours, but the closer neighbour's vote weighs more than the farther ones [65]. A weight, w_j , is attributed to the j^{th} closest instance and can be defined as:

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1}, & d_k \neq d_1 \\ 1, & d_k = d_1 \end{cases} \quad (4.2)$$

The weight of the neighbour vote ranges from the maximum, one, for the nearest neighbour (1^{st}) to the minimum, zero, for the farthest instance (k^{th}) in k closest instances. The voting is then performed using the accumulated weight as a vote for every class. The conventional kNN can be viewed as a variant of distance-weighted with all distance having weight of one. For the feature-weighted case, the modification would be made on the distance calculation. Features that have stronger influence will be given a large coefficient and the product for every feature will be accumulated as a distance as conventional distance (Euclidean and Manhattan) calculation [66].

4.2.1 Application of kNN

kNN as a classification method is practical in a wide range of fields other than sensors including but not limited to:

- **Bio informatics**

kNN classification is important in this field to analyse and maintain the data. A study by Tahir *et al* proposed kNN to implement cancer classification [67]. It was tested with two cancer databases, prostate and breast, and showed a satisfactory result with an error less than 5%. It can maintain the data quality by filling in missing values using a value obtained from related cases. This is essential as it may have impact on the descriptive and inferential statistics. It provides a better solution than simply ignoring the missing value [68].

- **Computer vision**

kNN is commonly used to find the similarity between an unknown query image and a stored image. Like other classification problems, feature extraction is one of the most important steps [69]. In the work of Kumar *et al.*, the authors proposed fruit grading based on machine vision. The classification identified fruit based on features like morphology and colour.

- **Document/ information retrieval**

Here, kNN is used to retrieve documents from storage based on query keyword(s). Manne *et al* reported such application based on kNN [70]. The stored documents were conceptually represented by keywords extracted from the documents. The keywords are given weights that represents their importance within the text.

4.3 Methods to improve the kNN algorithm

As illustrated in section 4.2, the algorithm's time complexity for each Euclidean distance calculation (equation(4.1)) would reach $O(D)$, where D is the number of data dimensions (or attributes). The "big-o" notation (O) indicates the upper limit of the algorithm computational time. The time complexity to find a single nearest neighbour is $O(n)$, where n is training data set size. Consequently, the algorithm is required to go through all instances to find nearest neighbour and this results in time complexity of $O(Dn^2)$. The kNN algorithm execution would be not practical in higher dimensions. Different methods have explored improvements in the algorithm execution in terms of total execution time and complexity. These methods include data dimension reduction [71], partial distance [72], pre-structuring [73], and editing[74, 75]. The algorithm operations are also commonly implemented in parallel to reduce the algorithm execution time.

4.3.1 Data dimension reduction

For classification applications, instance features (dimensions) are essential for the algorithm to find the difference between classes and reach to a prediction. However, the system is only effective if the number of dimensions of each instance does not overwhelm the system computational power. Typical applications that require dimension reduction include image classification and genome sequences modelling due to the large input data dimension. In practice, images are commonly downscaled before being fed into a classification algorithm. If, for example, an image is downscaled to 256 x 256 (rows x columns) pixels, the classification is still required to process 65536 dimensions (256 x 256) for each greyscale image. Therefore, it is common to perform feature extraction that is an equivalent of dimension reduction in this case. The dimension reduction is possible for some cases due to the fact that an event that surfaces in high dimensional space can actually be governed by a few simple variables [71]. In these cases, other dimensions are redundant for different reasons. First, the dimensions could have negligible variation and thus render itself irrelevant. Second, the dimensions sometimes could correlated with each other and therefore only one is needed. In many cases, the produced results reduce the algorithm complexity and also reduce the memory requirements. If the dimension is reduced lower than four, this can aid a person to visualise the data in plots as humans usually perceive

things in space that have lower or equal to three dimensions. A common practice to reduce the data set dimension is utilising the principal component analysis (PCA). This algorithm find the directions that data varied the most as the principal axis with the assumption that this direction has larger signal to noise ratio compared to the others. The number of principal axes that been found can be smaller than the original dimension and thus achieve dimension reduction. This method reduces the classification algorithm complexity. However, there are some weakness as in this method that it does not work well for principal components with nonlinear data structure and the total number of principal components to keep is unknown [71].

4.3.2 Partial distance

The partial distance method reduces the computational time by terminating a distance calculation once its partial distance is greater than the current closest candidates [72]. This method takes advantage of the distance computation that required accumulation of different dimensions (Euclidean distance in equation(4.1)). To make it more effective, the designer could identify the features that have a larger magnitude and start the comparison from these features. The partial distance (for Euclidean distance) based on r selected dimensions out of full D dimensions is:

$$d_r(a,b) = \left[\sum_{n=1}^r (a_n - b_n)^2 \right]^{1/2} \quad r < D \quad (4.3)$$

Where d is the distance between the test instance (a) with the stored training instance (b), D is the total number of features.

4.3.3 Pre-structure

The pre-structure method partitions the space that the data points reside [73]. This is to reduce the points to be queried by skipping unlikely regions. Figure 4-2 depicts a popular candidate of pre-structure method, the kd-tree, in two dimensions. The tree is constructed hierarchically by recursively splitting the data space at the median as shown in the figure. The splitting is binary using single line position at the median and, subsequently, defines two regions that are roughly balanced in term of data size. The process is usually started by list sorting to ease the process to find the median of the data [76] and makes an assumption that no two points have the same x or y coordinate. The median (splitting value) is stored at the root and the two subsets stored in the two subtrees. The splitting process starts at one dimension and subsequently with next dimension (cycling through the dimensions by returning to the first dimension when the process reaches the last dimension). This process continues until no data points remain outside of the tree. The tree can accommodate multi-dimensional data as indicated by its name, kd-tree.

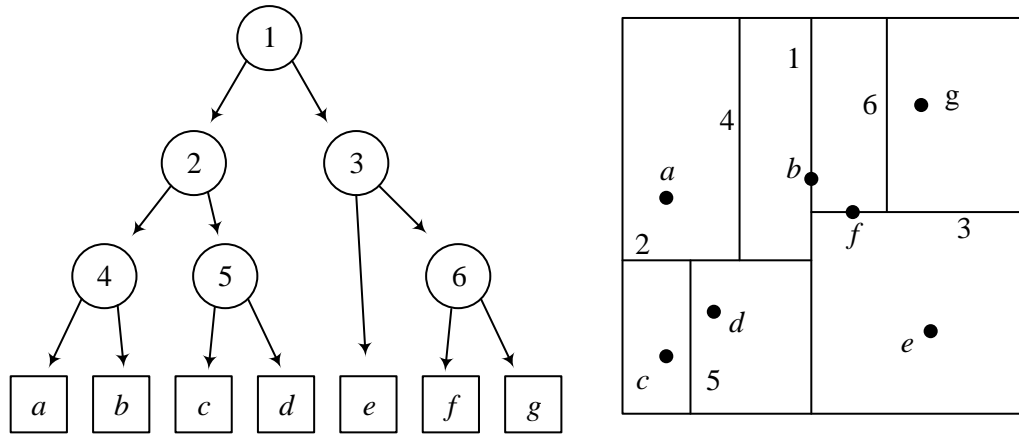


Figure 4-2 An example of kd-tree (2d-tree).

To query a test instance's neighbours, the process starts from the root node to determine the subtree that confined the test instance. The instance's corresponding coordinate is compared with the node value and the process continues recursively until the query reaches a leaf node. To find the k number of nearest neighbour, once the leaf node is encountered, the algorithm back-tracks to its root node. The number of back-track steps depends on the number of neighbours that are required. This method accelerates the search for nearest neighbours of the test instance but still retains the same storage requirements and other kNN weakness. In addition, the structure become less effective as the number of dimensions grows [61].

4.3.4 Instance reduction method

Different training set size reduction methods have been proposed to alleviate the k -nearest neighbour (kNN) computational requirements. There are two major types, (i) prototype selection and, (ii) prototype generation, to address this issue. In the prototype selection method, a subset of the training set is retained as the representatives and the original training set discarded. In the prototype generation method, new artificial instances are generated as the representatives to replace the original training set. The representative generation process is based on the information gathered from the original training set. There are different variants for both types and depending on the method, it can remove noisy instances, redundant instances or both together.

For prototype selection, there are different strategies for implementation that include condensation, edition and hybrid [75]. The condensation strategy tends to retain the points that are close to the class boundary. These points are also known as border points. The intuition behind border points keeping is that the internal points have smaller effect on the classification result compared to the border points. Therefore, there is little consequence in removing them. This results in a very high data set compression rate as the border points typically only constitute a small part of the whole training set. However, in some cases this method keeps the outliers that

degrade the generalisation accuracy. On the other hand, the edition strategy tends to remove borders points and keep internal points. The noisy instances mostly happen to be the border points as the internal points are surrounded by the instances from the same class. Typically, following the removal of the border points, the class boundaries become smoother and the generalisation accuracy is improved. The last strategy, hybrid, removes both border and internal points based on criteria used by previous two strategies. This strategy aims to give a small subset of the data while maintaining or increasing the generalisation accuracy.

For prototype generation, there are also implementation strategies including, class relabelling, centroid based, space splitting and position adjustment [74, 77]. The class relabelling is to improve the generalisation accuracy. The strategy changes the class label of questionable instances from the training set. This is to cope with flaws within the training set such as mislabelled, noisy, and atypical cases. The reduction capability is mostly negligible as the strategy only aims to find instances and relabel its class without removing any instances. The centroid based strategy generates an artificial prototype by merging the instances from similar classes. The merging process typically involves an average or a weighted average of attributes. The space splitting strategy finds approximate class boundaries and then generates prototypes that fulfil the boundaries. The reduction rates are usually dependent on the number of regions that are needed to represent the training set. The positioning adjustment is accompanied with other reduction methods to produce initial prototypes. This strategy aims to correct the position of the initial subset of a training set to optimise the generalisation accuracy. The correction is achieved through movement in the m -dimensional space.

The majority of the pre-process strategies presented in this section present two advantages, speed and storage, for the kNN implementation. For some cases, the prototype selection strategy is a better candidate especially for real physical data handling as the prototypes from the generation method sometimes do not make sense [75]. This is due to the fact that the generated prototypes might be not suitable for any real physical situation. However, one of the prototype selection method disadvantages is that most of them report a prohibitive runtime or even cannot be applied over large size data sets. This is due to the fact they tend to go through all instances and have to iterate through the subset during its selection process.

The kNN parameter, k , might need to be changed after the reduction method took place. For an aggressive reduction method, the training set is reduced to only a single instance to represent what was formerly a cluster of instances. Therefore, it is sensible to use $k = 1$ rather than $k > 1$ in this particular case. The k number is allowed to be reduced after noisy data have been eliminated from training data during reduction process [61]. For other cases, the k number should remain unchanged.

4.4 Implementation of the kNN in an embedded system

The embedded system design offers the potential for creating a portable system that can be deployed in various environments. kNN, as a simple and powerful algorithm, has found many applications as discussed in section 4.2.1. However, the majority of the applications that require kNN implementation use a host PC. On the other hand, there are a number of cases in the literature aimed at improving the kNN implementation through parallel processing. Different strategies (architectures and platforms) have been used to implement parallel processing to accelerate the classification process.

Tou *et al.* proposed a texture classification using kNN using an embedded platform. [78] The system was evaluated using offline images (64×64 pixels). To facilitate the system deployment, the authors evaluated different feature extractions first before being included into the system. The evaluation was performed in terms of computational time and accuracy. The classification execution time for two PCs (running Linux Ubuntu 8.04 and Debian Linux (virtual machine) using Intel T7500 Core 2 Duo 2.20GHz with 4GB of RAM) and two ARM processing boards (running Debian Linux using ARM920T ECV Platform and ARM926EJ-S ECV Platform at 200MHz with 64MB of RAM) were evaluated in this study. The results showed that the fastest time of the ARM processing board was 86 times slower than the fastest time of the PC platform. The authors also argued that the classification using the PC platform was implemented using a better optimised MATLAB[®] code as compared to the self-developed C code.

Attarn *et al.* designed a real time personalised stress detection system using a local processor [43] implementation in different platforms including the Raspberry Pi 3B (baseline), Jetson TX1 GPU, Jetson TX2 GPU, ASIC and Artix-7 100T FPGA. These were evaluated using two classifiers, support vector machine (SVM) and kNN. The authors selected four most sensitive signals from 16 features from multiple physiological signals, such as electrocardiogram (ECG), respiration, oxygen saturation (SpO₂) and acceleration. The average classification accuracies of the KNN and SVM were 95.8% and 96.7%, respectively. In terms of energy efficiency for both classifiers, both the ASIC and FPGA implementations outperformed the other processors by at least four orders of magnitude. The ASIC performance, in term of energy efficiency, was better by 42x and 12x over the FPGA platform for SVM and KNN. The authors also opined that the high development cost and time constraints may not be practical to deploy the system based on an ASIC.

Chen *et al.* proposed a 16-channel smart brain sensor system on chip (SoC) integrated with the integrated digital EEG/ECoG processor (DEEP) to perform multi-dimension feature space analysis algorithms [79]. The pre-processing and feature extractions were performed by dedicated programmable hardware unit to facilitate real time detection. A reduced instruction set

computer (RISC) processor was embedded to preserve fully programmability for the classification algorithms. In this study, kNN was utilised to perform the classification.

Viseh *et al.* proposed a tongue drive system using a local processor as discussed in section 3.4 [51]. The prototype system based on the FPGA only occupied a very small footprint that could be placed inside the user's mouth. The authors evaluated different classification combinations (all combinations included the kNN classifier), such as single- and double-stage classifiers. The single kNN with 6.68% misclassification rate was chosen. The classifier is fed with four sensor values at 50 samples/s. Each sensor read three-axis values (X, Y, and Z), 12 bits each. The authors also aimed to implement the design in ASIC to further reduce the energy consumption by a factor of 15x. This was to eliminate the need for a bulky battery and achieve longer operation hours. This also reduced the system footprint and improve the user experience.

4.4.1 kNN implementation architecture

Different works based on the kNN were introduced in the previous section and the limitations of implementations using a sequential processor discussed. However, some works preferred the kNN implementation using a sequential processor as its application has a lighter workload or a lower throughput rate. For predetermined regular algorithms, a dedicated hardware would be the most efficient approach when it is applicable. There are a few different parallel computing architectures to implement the distance calculator for the kNN including systolic array [80] and adder tree [81].

The adder tree is a simple and fast method to sum all inputs as shown in Figure 4-3. The distance between the input data and stored data is found by accumulating the partial squared distance in all dimensions. The partial squared distance calculation, $(a_r - b_r)^2$, is performed using a series of processing elements (PEs) of subtraction and square operation (where r is the current dimension, a and b are the incoming test instance and the stored training instance, respectively). PE is a component that is capable of performing arithmetic functions and logic functions and sharing its information with its neighbour immediately after processing. The PE lacks the central processing unit's key components such as program counter and control unit to be an independent processing unit. The tree has a time complexity of $O(\log_2 D)$ but this is at the expense of space complexity of $O(2^{\log_2 D})$. This build can be pipelined and produce one output for every clock cycle.

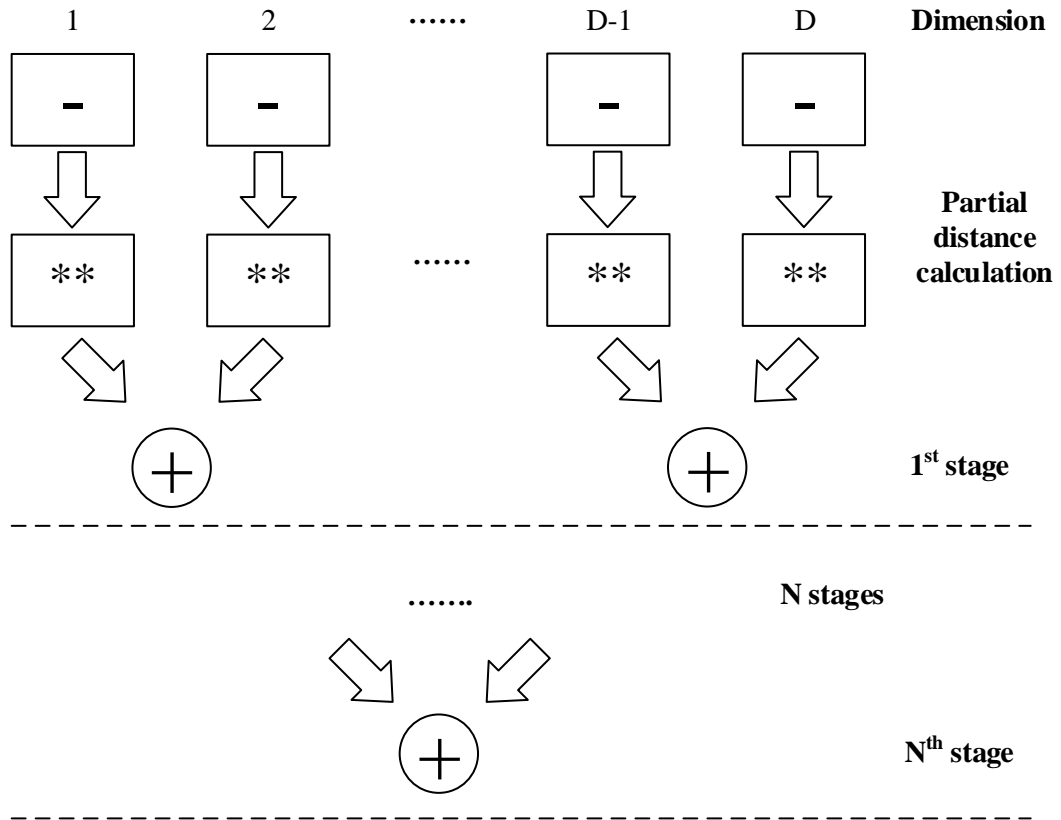


Figure 4-3 Adder tree (D input) for Euclidean distance calculation.

A systolic array is an arrangement of PE where data flows synchronously across the array between neighbours, usually with different data flowing in different directions. Figure 4-4 depicts an example of a general 2-dimensional systolic array. The array is to be a general methodology for mapping high level computation into hardware structures instead of depending on ad hoc approaches [82]. The design accommodates different dimensions and effective usage of I/O's. The array allows adjacent PEs to reuse the input data and reduces I/O requirements [83]. The systolic array presents a highly parallel and modular approach for complex signal processing. It has found different applications including matrix multiplication [84].

Stamoulias *et al.* proposed an IP core design for kNN implementation using an FPGA [80]. Within the systolic array, a memory element is incorporated into every PEs to store the training set. This allowed the training set to break into few portions and parallel processing in the PEs across the row. The local memory removes the input from one boundary. The design was a modification of a traditional systolic array by including an enable controller. The distance calculation was based on the Manhattan distance metric. There were two variants derived from the design to accommodate a single dimension with a large training set and a large number of dimensions with a small training set. The systolic array design also includes the k-nearest neighbour finding and the class majority voting.

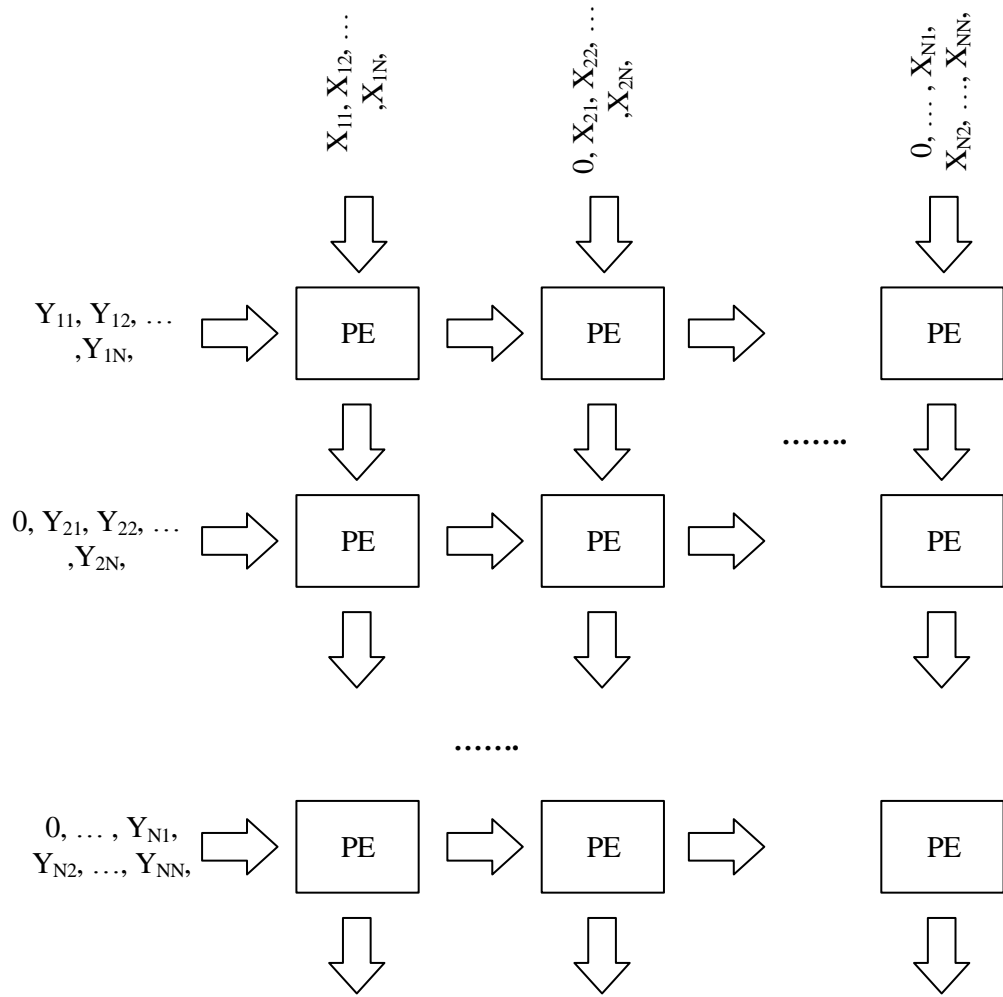


Figure 4-4 General two-dimensional systolic array.

4.5 Summary

This chapter started with a discussion into the kNN classifier basics. The kNN was further discussed with the influence of its parameters on its algorithm and its complexity. From there, different variants of the kNN were discussed that included different distance metrics and ways to reach the prediction. Examples were introduced and discussed to understand the application of kNN in real world scenarios. Different methods were discussed to improve kNN in term of computational speed and classification accuracy. Four methods were introduced and the instance reduction method was emphasised. Different instance reduction strategies were discussed to understand its pros and cons.

Different examples of the kNN implementation using an embedded system were discussed. The examples included different architectures and platforms. A comparison on the implementation approaches using different platforms was provided. This was followed by the kNN implementation using parallel processing architecture and its features.

Chapter 5 Experimental sensor system design

5.1 Introduction

Different sensor system designs and their underlying architectures were discussed in previous chapters. In this work, the working principle of the sensor used was based on surface Plasmon resonance (SPR) and utilised multi-wavelength information to determine information on the measurand. The SPR sensor was used within the optical fibre sensor system for measuring the chemical composition of a liquid. This chapter will discuss the detailed system design for interfacing the electronic to the SPR sensor and the subsystems required to achieve the multi-wavelength interrogation. The sensor system consists of four main modules included a SPR sensor (provided by Bangkok University), as shown in Figure 5-1. The four main modules are the:

1. Optical sensor – SPR sensor connected to the transmitter and receiver modules using plastic optical fibre (POF).
2. Transmitter module – light emitting diode (LED) light source.
3. Receiver module – photodiode based receiver circuit.
4. System digital core – field programmable gate array (FPGA) based configurable core for system control and communications.

In following sections, the understanding of the subsystem behaviour is developed in order to create a functional optical sensor system. The optoelectronic components, the LED and photodetector driving circuit behaviours are discussed in detail in order to design the corresponding modules in the FPGA. The section as well discusses the utilisation of supporting modules, included in the FPGA (Xilinx[®] Artix-7), to facilitate user interfacing to a personal computer (PC). The PC interface allows user to perform system control and parameter setting.

The design of the digital system implemented in the FPGA is as well elaborated. The printed circuit board (PCB) designed and fabricated to connect the optoelectronic circuits to the FPGA is shown in Appendix F. The performance of the system is presented along with test results in order to validate the system operation.

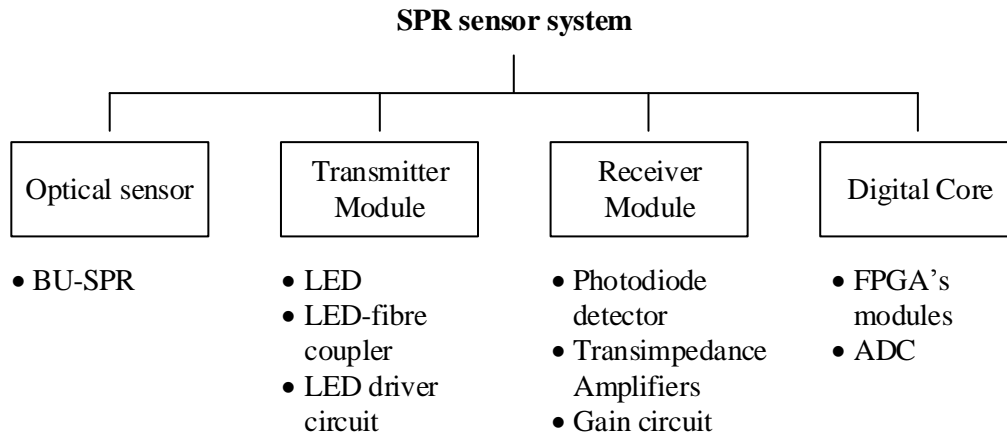


Figure 5-1 Sensor system breakdown.

This chapter is structured as follows. Section 5.2 will discuss the BU-SPR sensor. Section 5.3 will discuss the electronic design for the sensor subsystems. Section 5.4 will discuss the FPGA modules created to regulate the system operation. Section 5.5 will discuss the system interface on host computer that was built using the Python open-source scripting language. Section 5.6 will discuss the performance of the integrated system. Section 5.7 summarises this chapter.

5.2 SPR sensor

The SPR sensor used in this system was provided by Bangkok University as reported by Somarapalli *et al.* [85]. The reported sensor is robust, compact ($5 \times 3 \times 3$ cm) and requires few optical components as depicted in Figure 5-2. The sensor housing is 3D printed to provide mechanical strength for optics alignment. The SPR chip was fabricated by sputtering a thin gold film (50 nm and 70 nm thickness) on a glass slide. A 5 nm chromium layer is pre-coated to provide an adhesion layer for the gold on the glass slide surface. It should be noted that the sensor is not strictly a sensor as identified in section 2.4, but more of a refractive index transducer as it does not include a selective layer at the sensing region (on the gold layer). However, in this thesis it is always referred to as a sensor. In this sensor, the SPR signal is excited by the guided light bouncing inside the glass slide at the proper angle. The sensor housing is designed such that the input and output fibres are fixed at an angle of 26° . This angle provides an SPR reflectance spectrum dip around 600 nm wavelength (for both 50 nm and 70 nm gold film) when using water ($n = 1.33$) as a superstrate. The ends of both fibres are placed at close proximity to the glass slide

in order to maximise light coupling and collection. It was shown in the literature that the experimental SPR curve has a broader dip in contrast to the simulation work. This response difference is due to the assumption (in simulation) of each ray of light shone onto the gold surface has the same incident angle. However, in the experimental work, the incident light that illuminates the gold surface does not have parallel incoming rays, but an angular distribution (cone shape) due to the LED (illumination distribution, cone shape) and the optical fibre (multi-mode) characteristics.

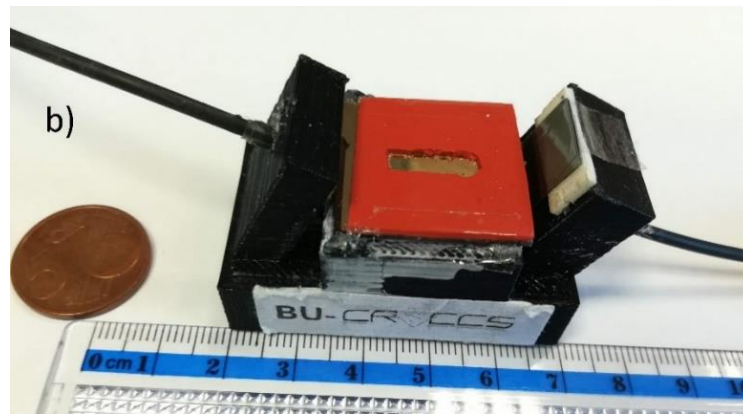
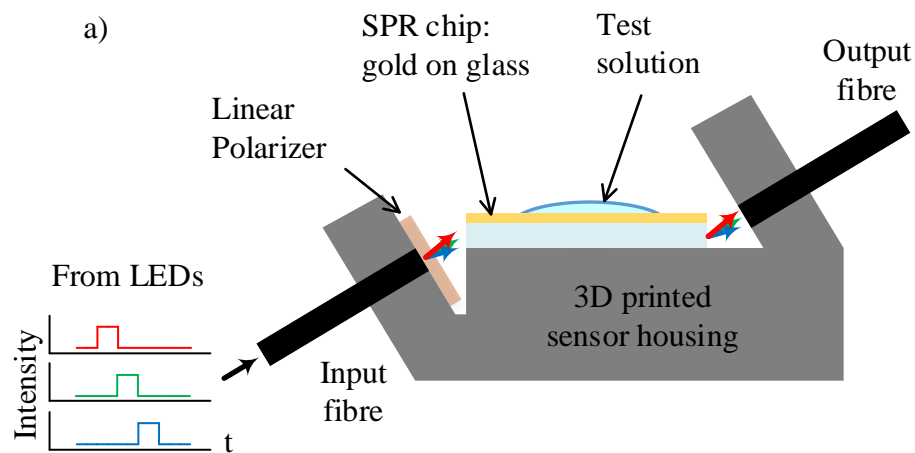


Figure 5-2 BU-SPR sensor: a) schematic diagram. b) image.

Intensity modulated light is generated using a red-green-blue (RGB) LED light source and coupled to the glass slab through a large core plastic optical fibre to maximise the coupling efficiency. The optical fibre centre is aligned to the LED's centre using 3D printed cap. The cap provides better repeatability in the measurements when the fibres are unplugged from and re-plugged to LED. The cap is mounted to the fibres by the application of hot glue. Black tape is used to cover the connection in order to minimise ambient light influence (shown in Figure 1-2(c)). A linear polariser is placed at the end of the input fibre to ensure transverse-magnetic

(TM) polarised light excitation. A connectorless form photodiode (IF-D91) was chosen as optical detector as shown in Appendix E.

Figure 5-2(a) depicts both the time domain intensity modulation technique and the schematic of the sensor used in the current system. In this modulation technique, the RGB LED is turned on in a sequential manner and only a single colour is switched at a time. This is to ensure no mixing between different colour channels. The timing for the LED to switch on and off is synchronised with the system’s analogue-to-digital converter (ADC). The light intensity data is labelled with a different header for each colour during data acquisition. Each colour channel emitted by the RGB LED covers a specific wavelength band as illustrated using the example in Figure 5-3(a). The recorded value by the detector for each channel is proportional to the integration of the SPR spectrum over the colour band.

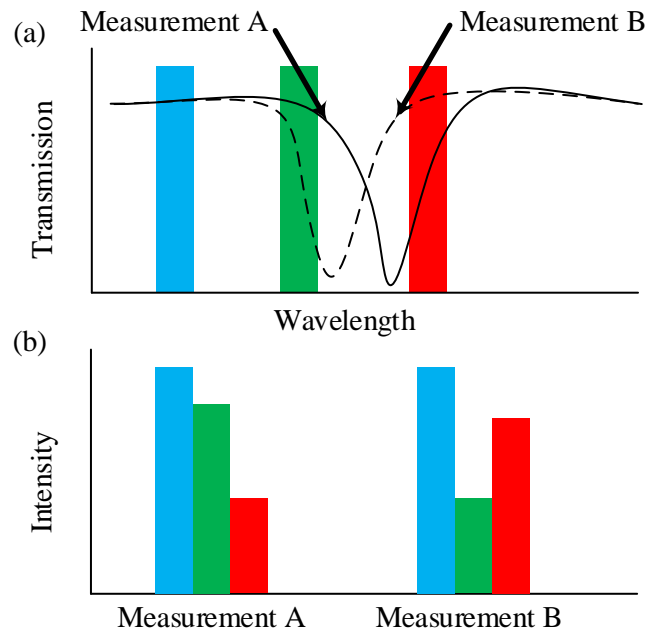


Figure 5-3 Sensor response with different measurand (Refractive Index).

An example of the SPR sensor’s response is illustrated in Figure 5-3(a). The figure shows a drawing of the two possible SPR responses at different conditions, labelled measurements A and B. The characteristic dip shifts under different conditions. As shown in the illustration in Figure 5-3(a), the dip in measurement A is red shifted compared to B. The change of the spectrum response yields different intensity values for the three colour channels. This can result in a detector read out as illustrated in Figure 5-3(b). Such scheme also can accommodate different sensor as well. The light source and receiver may need to be changed to suits the sensor response.

5.3 Sensor system electronic design

For the presented example in section 5.2, the accuracy of the readings depends on the stability of the detected signal for each colour channel. Several factors can affect the signal readout stability such as interference from the ambient light, mechanical vibration, and stability of both the electronic output and light output. To minimise the ambient light effect, a black cover and time modulation scheme were applied to eliminate low frequency ambient light effects. Mechanical stability was ensured by using a 3D printed structure. Both the stability of the electronic circuit output and light output depend on the electronic circuit design. The electronic circuit design includes power supply, light source circuit and photodetector circuit are detailed in this section. This section also discusses a low power wireless communication module, XBee, used in the reported work. Figure 5-4 depicts the block diagram of the required electronic subsystem and connection. The arrow represents the signal direction while the communication with computer is available in both wired and wireless.

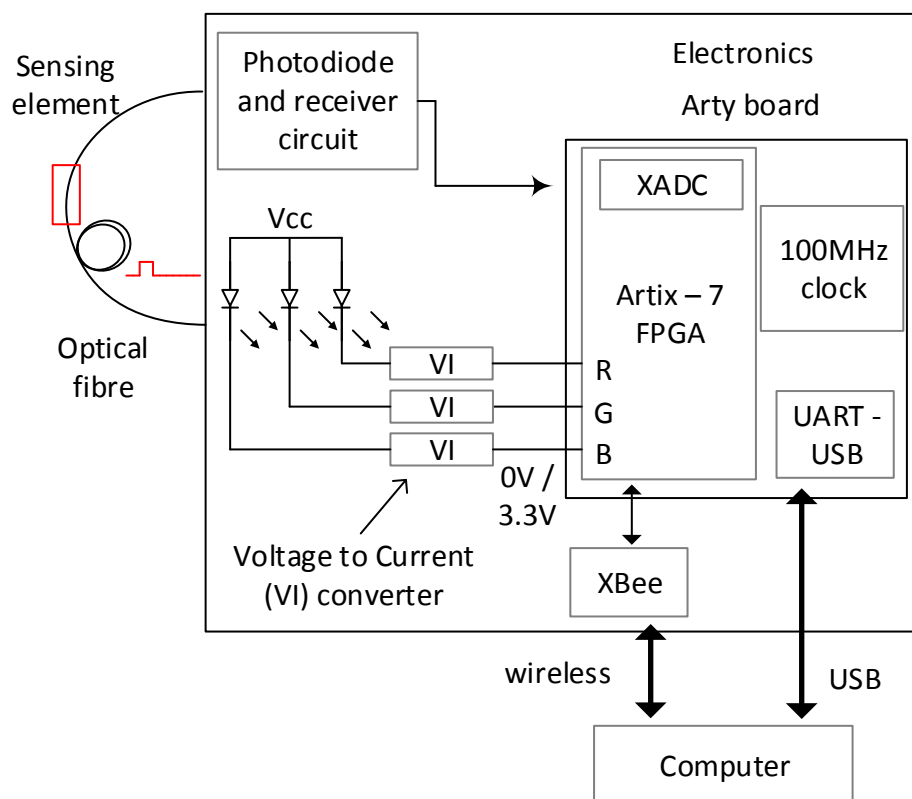


Figure 5-4 Sensor system and connection.

5.3.1 Power supply and regulator

To ensure that the electronic components operated as required, the power supply was required to provide a stable DC voltage level. This was to minimise noise being introduced onto the recorded signal. The power supply consisted of battery packs and voltage regulators for both positive and

negative voltages. The voltage regulator was designed to make sure that the applied potential remained constant at different battery states of either charging or being under load. Here, a 7805 linear regulator was used for the +5 V supply and another 7905 linear regulator was used for the -5 V supply. A linear regulator was chosen, as it produces no switching noise. A reverse bias protection was included as well in the design using 1N4001 rectifier diodes. It was connected in reverse biased fashion across the battery input pin. This allowed current to pass through it instead of the rest of the circuit in the event of a reverse biased connection.

5.3.2 Light source circuit

The RGB LED was used in this work as the light source for the optical sensor. The LED was packaged with a hemispherical lens that minimised the divergence angle. Hence, it could directly couple to a bare optical fibre. That reduced the design complexity and number of components. Each colour channel produced by the LED source had a relatively narrow spectral range (20 nm). The detected signal is an integration of the transmitted light intensity over the channel spectral range. In this work, the SPR phenomenon was used for sensing using BU-SPR configuration [85]. The original design was modified such that the white LED was replaced by an RGB LED and the output signal intensity is recorded instead of the spectrum.

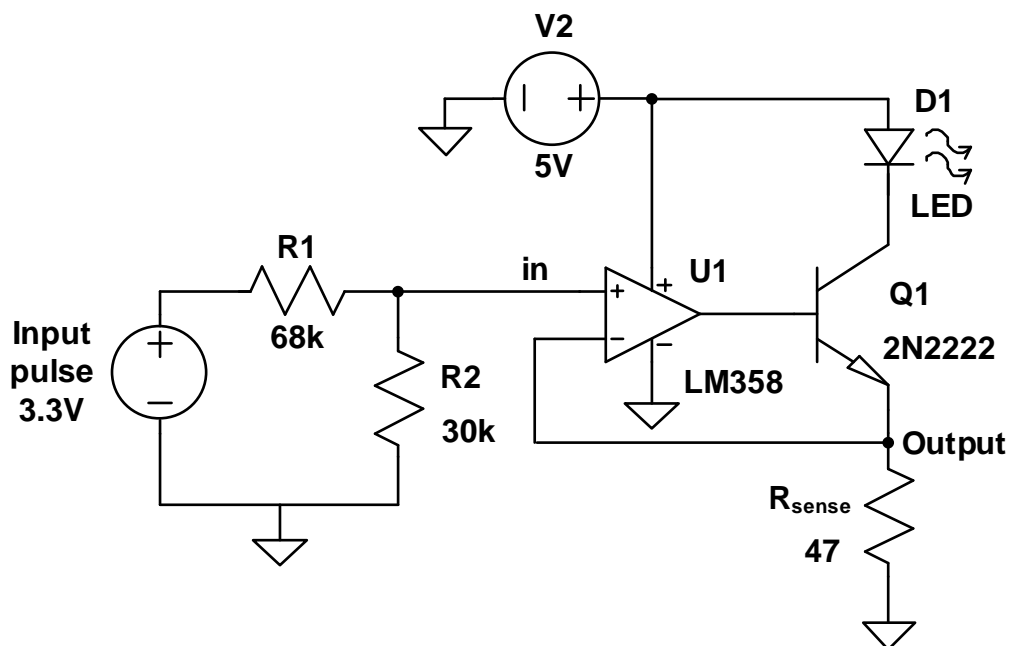


Figure 5-5 LED driver – current source and potential divider. Input pulse represent FPGA output.

Similar to every diode, the LED current flow increases exponentially with forward voltage after the knee voltage. Its forward voltage also varies with temperature or with the current passing through it. Therefore, the LED is suitable to be regulated using a current source. In this work, the current source was designed to receive voltage input for the LED's brightness control as shown

in Figure 5-5. Three separate identical current sources are used to drive the RGB LED. The current source consists of an op-amp LM358, a transistor 2N2222, and current sensing resistor, R_{sense} (47Ω), that sets the LED maximum allowed current to avoid damage to the LED. Lower cost components instead of faster components were chosen in this system to have lower total cost. The voltage-to-current (VI) conversion was achieved using an op-amp as a unity gain amplifier with a transistor in the feedback loop and a current sensing resistor at the output. A transistor was included to provide current flow control, whilst the op-amp ensured the voltage output was stable. The potential difference across the current sensing resistor was used as feedback to the op-amp to maintain the current flow through it. Before the current source, a potential divider was used to step down the FPGA's digital output from 3.3 V to 1 V. This resulted in lower voltage across the current sensing resistor

Time domain intensity modulation was used in this work and only a single colour LED was switched on at a time. For each LED colour, the transmitted light was sampled and recorded by the receiver. Figure 5-6 depicts the output of the FPGA and the response of the LED driver circuit. The LED current is measured by the voltage across the current sensing resistor (R3). Due to the op-amp's slew rate, the current flow could not react immediately with the FPGA's raising edge and required some time to reach its plateau. However, this should not affect the performance of the system as the rise time is taken into account during ADC sampling.

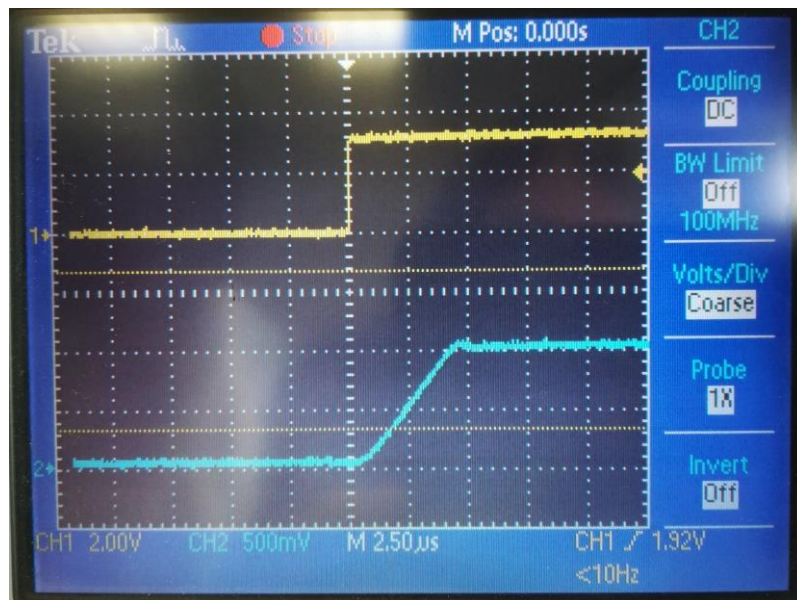


Figure 5-6 LED driver output - FPGA output in yellow line (first channel) and voltage across current sensing resistor (second channel).

5.3.3 Photodetector circuit

Typically, a photodiode provides linear current response [86] and therefore a transimpedance amplifier is used to act as a current-to-voltage (IV) converter to produce a voltage signal. The

transimpedance amplifier also reduces signal swing across the photodiode capacitance in order to increase the operational bandwidth [86]. In the simplest form, the transimpedance amplifier can be implemented using only an op-amp and a large value feedback resistor. Though simple, several criteria need to be optimised such as amplifier gain and phase margin. During design, the amplifier output voltage swing needs to be initially determined in order to match the ADC's input range. The amplifier output voltage is determined by the amplifier gain that is proportional to the feedback resistor's value. Phase margin is an important criterion that determines the circuit stability. Component choice is also critical in the designing stage when it involves a small current signal or large gain. [87] In this work, the current output from the photodiode is in the range of tens of nA's. This is comparable with typical op-amp input bias current. A precision picoamp input op-amp, LT1112, was used in this work with input bias current only at maximum of 250 pA. To achieve a better matching with the ADC input voltage span, a noninverting amplifier was also used to increase the voltage swing.

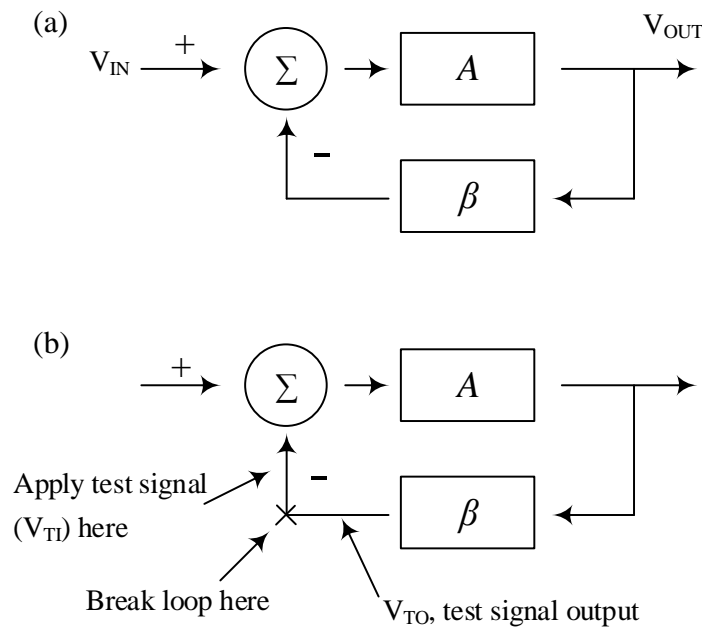


Figure 5-7(a) Feedback circuit block diagram. (b) Feedback loop is broken to find out the loop gain.

The circuit stability was analysed using a feedback loop as shown in Figure 5-7(a). Figure 5-7(a) depicts a block diagram of the op-amp's inverting feedback loop in order to simplify the analysis. The output voltage is

$$V_{out} = A(V_{in} - \beta V_{out}) \quad (5.1)$$

In equation (5.1), A is the amplifier gain and β is the feedback ratio. The amplifier gain can be expressed as:

$$\frac{V_{out}}{V_{in}} = \frac{A}{1 + A\beta} \quad (5.2)$$

A low value of phase margin for the op-amp circuit produces undesired oscillations that degrades the signal integrity. The circuit becomes unstable when the amplifier gain reaches infinity. Instability happens when the value of the loop gain, $A\beta$ reaches -1 or $|1| \angle -180$. To measure the loop gain, the feedback loop is broken as shown in Figure 5-7(b). An AC test signal with an amplitude of $V_{TI} = 1$ V is injected at the broken loop end. The output of the feedback loop, V_{TO} , is measured at the other end, $V_{TO} = A\beta$ [88].

In the analysis of the circuit in Figure 5-8, the photodiode is replaced by its equivalent circuit shown in a red dotted box. The photodiode equivalent circuit consists of a current source, a capacitor and a resistor that is placed in parallel and its values set accordingly to the characteristics of the Industrial FiberOptics IF-D91 photodiode used. In this simulation, the current source was removed to probe the feedback behaviour. A test signal is injected into the inverting input and the output is measured.

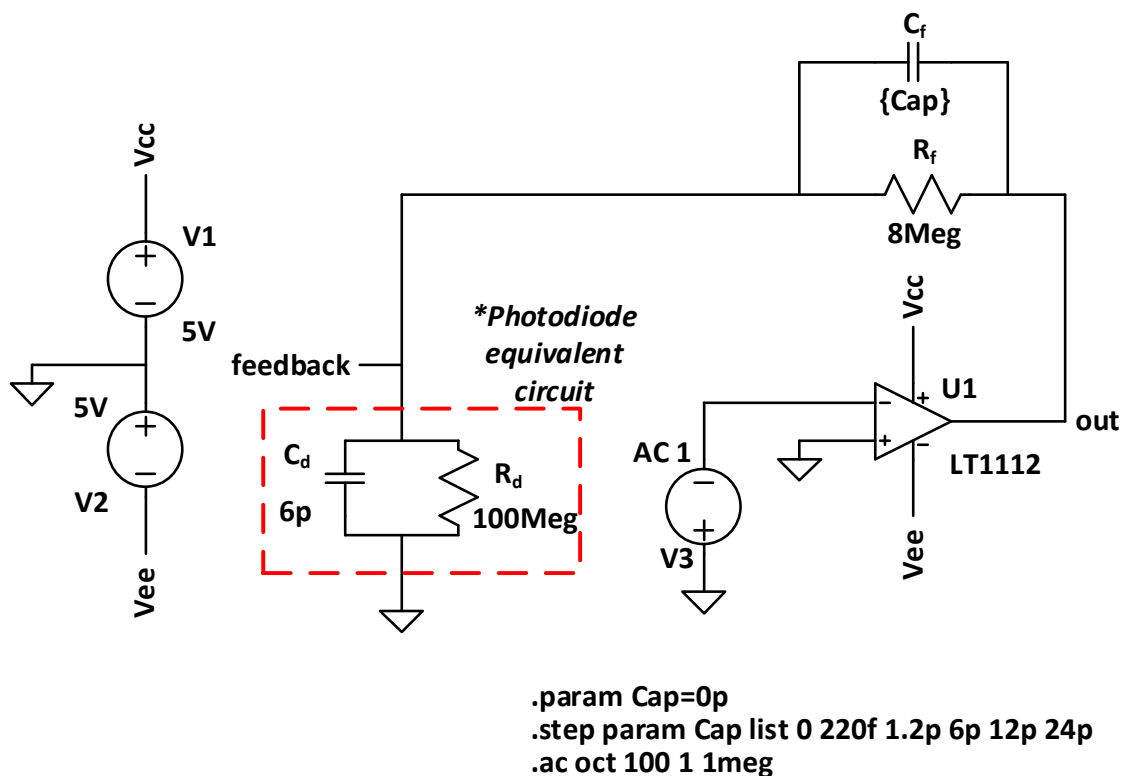


Figure 5-8 Feedback analysis circuit (loop is broken to inject test signal).

The stability of the circuit in Figure 5-8 was estimated using transient analysis and phase margin. The common practice to stabilise the transimpedance circuit is reported in [89] using a feedback capacitor:

$$C_f = \frac{1}{4\pi R_f f_{GBWP}} \left(1 + \sqrt{1 + 8\pi R_f C_d f_{GBWP}} \right) \quad (5.3)$$

The gain bandwidth product (GBP) for the LT1112 is $f_{GBWP} = 750 \text{ kHz}$ and the feedback resistor's value is $R_f = 8 \text{ M}\Omega$. The diode capacitance for IF-D91 photodiode is $C_d = 6 \text{ pF}$. Based on these values, the calculated C_f is 12 pF . The photodetector circuit was simulated using the Analog Devices LTspice IV software (Appendix B) for a range of capacitor values from 0 to 24 pF . The simulations included the estimated capacitance between the op-amp output and the inverting input pin (220 fF) [90]. This capacitance was estimated using a toolbox from Emissoftware [91] with the assumption that the op-amp package was a DIP (dual in-line package) and the DIP socket was combined as a coplanar flat conductor. The flat conductor width was set as 1.15 mm , the length as 7 mm and separated by 0.695 mm . The material of the DIP socket was an unknown plastic type. In the calculations, its relative permittivity was set to 2 (the lower end value for common plastic). The simulation results for both the gain and the phase are shown at Figure 5-9.

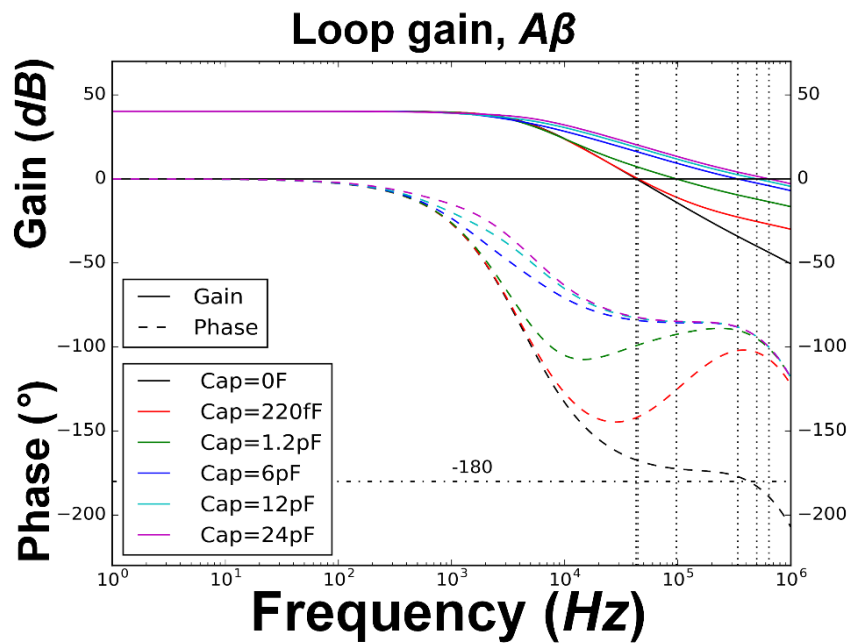


Figure 5-9 Bode Plot of loop gain signal.

Table 5-1 Phase margin for sweep simulation.

Feedback Capacitor(pF)	0	0.22	1.2	6	12	24
Phase Margin(°)	12.70	38.37	87.48	91.76	86.13	79.73

Table 5-1 shows the calculated phase margins from the simulation sweep. The phase margin is estimated as how far a circuit is to instability (-180° at 0 dB intercept). If the loop gain is lower than 0 dB , the feedback signal decays slowly. To ensure a stable circuit, the required loop gain's phase margin has to be at least 45° , otherwise oscillating transient behaviour prevails. Problems

that include overshoot and ringing are manifested before the loop gain reaches 180° phase shift [88]. The phase margin for 0 capacitance in the feedback loop is 12.73° and the oscillating behaviour is noticeable at the transient analysis in Figure 5-10. For $C_f = 220$ fF, the phase margin is 38.37° , which is close to stability. The stability improves with further increase of the feedback capacitance value until it reaches 6 pF. The circuit is stable with phase margin at 86.13° using the previously suggested value of $C_f = 12$ pF. Further increment of the capacitance provides no significant improvement for stability.

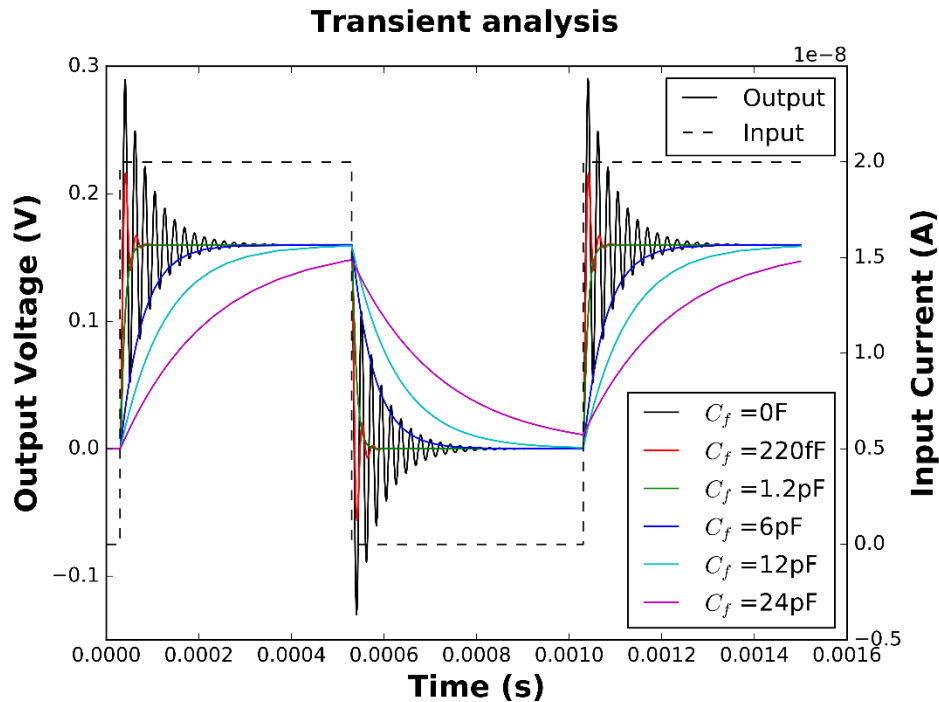


Figure 5-10 Transient analysis of optical receiver circuit.

The transient analysis is shown in Figure 5-10. The output becomes more stable as the feedback capacitance increases but the response becomes slower at the same time. Slow response of circuit slows down the overall system throughput. At a 0 capacitance value, the oscillation's swing is large and it is sustained for a long period. The oscillation behaviour slowly fades out with increasing the capacitance. It ceases to exist at 1.2 pF. For the 220 fF case, it shows acceptable oscillating behaviour and it reaches plateau response in a short time (~ 500 μ s). In the current system, the simplest form of the transimpedance circuit configuration is used. That includes the existing capacitance (220fF) between op-amp output and the inverting input pin. A screen shot of the circuit's transient behaviour (with 220fF effective capacitance) when measuring the SPR sensor's output (with air as measurand) is shown in Figure 5-11. The output response shows acceptable overshoot behaviour ($\sim 18\%$ of amplitude) during rising edge and it resembles the simulated waveform.

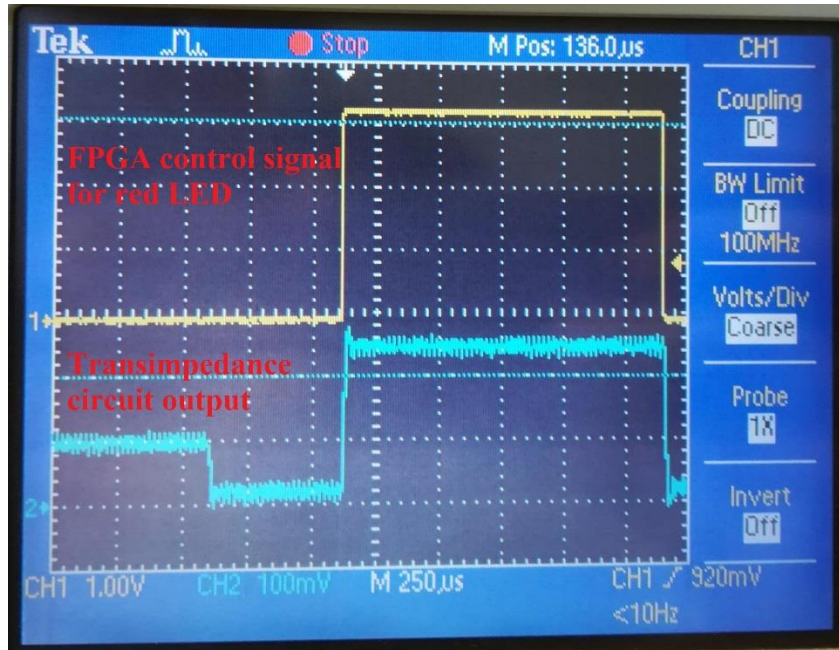


Figure 5-11 Transient behaviour (with 220 fF effective feedback capacitance) of transimpedance circuit in current system.

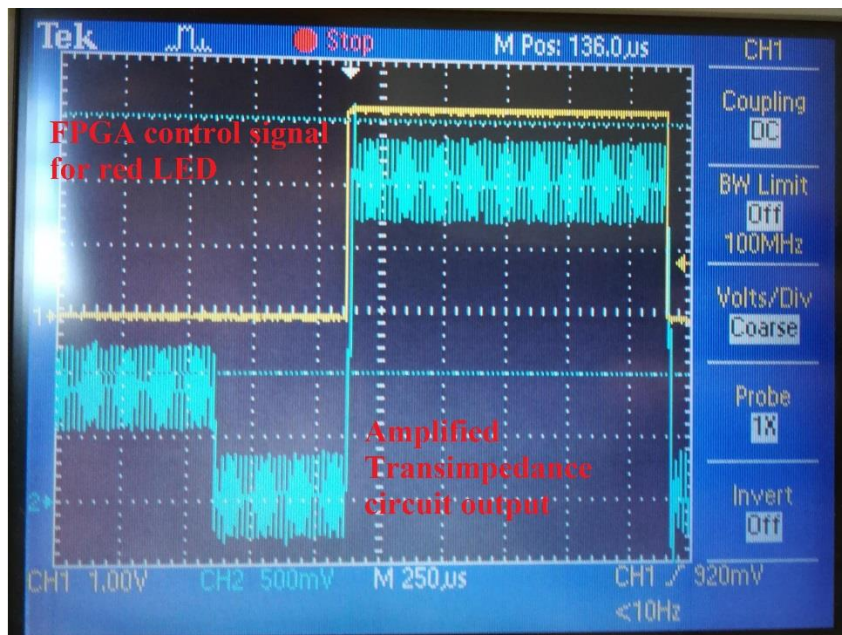


Figure 5-12 Transient behaviour (with 220 fF effective feedback capacitance) of amplified transimpedance circuit in current system.

In the current system, the transimpedance output is sampled using a built-in ADC (XADC) in Xilinx® Artix 7 FPGA. The output voltage measurement at the detector side is shown in Figure 5-11 for the red channel. It shows a dynamic range of 250 mV. This is not optimised to utilise the available XADC's range. The output is therefore amplified eleven times using non-inverting amplifier setup to have better match to the single ended XADC's 3.3 V range.

Figure 5-12 depicts the amplified transimpedance circuit output. The observed amplified output noise is roughly 500 mV peak-to-peak which is too high to be sampled. The noise is due to electromagnetic interference that is radiated from nearby digital circuit. The digital noise is suspected to be the FPGA clock signal as the noise is sustained even with the LED switched off. Therefore, an aluminium sheet is folded into a box shape and is grounded to shield the analog circuit. Figure 5-13 illustrates the amplified output of the shielded transimpedance circuit. The noise level is reduced to less than 100 mV.

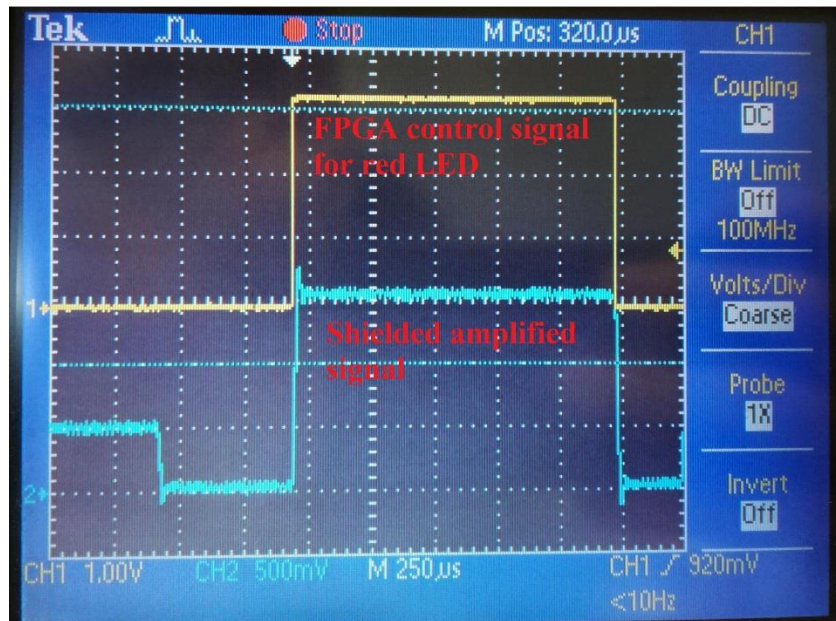


Figure 5-13 Transient behaviour of shielded amplified transimpedance circuit in current system. The resolution of the sensor system depends on both the gain and the ADC resolution. An 8.2 M Ω resistor was used for the IV conversion. This was combined with the ten times amplification and 12-bit ADC with its voltage span of 3.3 V giving a 9.83 pA/bit resolution.

5.3.4 XBee wireless communication

Wireless communication is an essential component for sensors network systems where a huge number of nodes needs to be connected to a server. Operating a remote node with wireless connection demands a constant power supply. Therefore, a low power and long-range wireless method is desired for this type of application. In some designs, data rate is compromised to achieve low power consumption such as Bluetooth Low Energy (BLE), LoRa and XBee [92]. To achieve long-range communication, LoRa uses frequency modulation method whilst both BLE and XBee capable to use mesh of nodes to pass information from one end to another. For line of sight coverage, LoRa can cover more than 1500 m (till 4000 m) and outperform other transmission technologies; XBee can cover around 1000 m; BLE can only covers less than 80 m but consume the least power [92]. All technologies are available in small radio module form that

can be included in a small PCB to interface with a microcontroller. These radio modules can be put into sleep mode for power conservation and only wake up for a short period for data transmission.

XBee is a flexible wireless communication module that is capable of operating in two different modes: mesh network, and serial communication. For a mesh network, the module can be programmed in different roles: a coordinator, a router, or an end device. In the coordinator mode, the XBee is responsible to form the network at the first place. In the router mode, it is responsible to keep or pass information from the child node. Both the coordinator and the router are able to accept new child nodes. In the end device, it is however a member in the network that is only responsible for its own incoming or outgoing data. There are different ways to address the receiver in the mesh, which can be in either point-to-point (unicast) or broadcast. The simplest function a XBee can provide is to act as a serial link between two XBee devices under transparent mode. It provides simple connection between the two nodes. In this mode, one XBee device acts as a coordinator and the other as an end device. The coordinator's destination address needs to be set as the end device serial number and the same applied to the end device. The serial number is a unique permanent 64-bit address that is assigned during manufacturing.

In this work, XBee S1 has been used. It operates with a 115200 Baud rate in transparent mode which communicates with the FPGA using the UART protocol. It is powered by 3.3 V power supply from the Arty board and draws ~50 mA during the operation. The radio module is mounted on the designed PCB and its two data IO – TX and RX are connected to the FPGA. The radio module is for serial communication between the proposed system and personal computer.

5.3.5 Printed circuit board development

The interface board for connecting the FPGA to the OFS was designed to be modular and with a small footprint (120 mm x 80 mm) as shown in Figure 5-14. Modularity allowed isolation of functionality and to make the functionality easier to change. The developed electronic circuit components were placed onto a printed circuit board (PCB) which was designed using the Target 3001 software. The PCB was designed to fit the existing Arduino shield connector on the FPGA development board. This design allowed the user to plug in the interface board from top. To ensure the high gain photodiode circuitry signal integrity, the circuit was partitioned from noisy digital circuitry and ground planes used. To minimise electromagnetic interference, the signal is mostly placed on the top surface and shielded using a grounded aluminium enclosure (bottom left of the figure) and bottom ground plane.

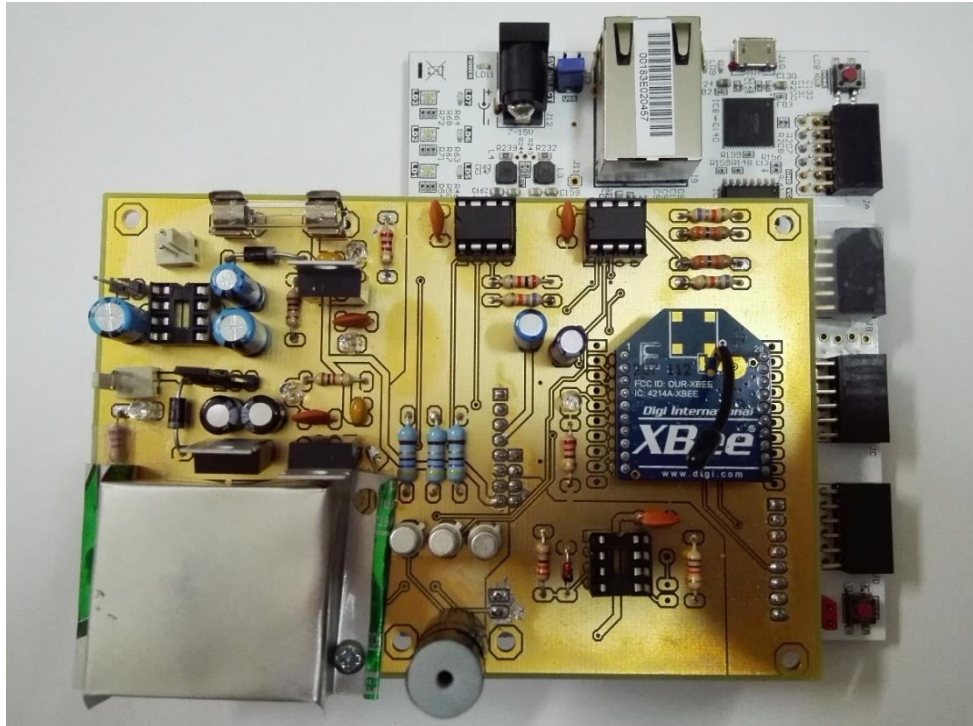


Figure 5-14 Developed PCB with the FPGA development board (underneath).

5.4 FPGA hardware component design

In the previous section, the development of the electronic circuit was discussed. The designed circuit was formed using a PCB that interfaced with an FPGA (using a 100MHz clock). The FPGA come with a development board that was used as a digital core to coordinate the system operation. The FPGA was responsible for different functions that includes communication, XADC timing control, light pulse generation, and classification. The proposed FPGA architecture is shown in Figure 5-15. The system was designed to implement time domain intensity modulation by synchronising the ADC acquisition timing and LED colour switching timing. This was achieved by synchronising the timing of the XADC controller and the light pulse generator. The synchronisation is realised by pairing the both modules using signals instead of hard coding timing. The XADC output was then processed for display and transmission. These functions were designed as modules and each module was assigned an address. For module setting, the user required to specify the module setting along with the module address. The setting was then passed to the target module by the main controller. The main controller is simply working as a multiplexer that matches the module address with the look up table and forwards the setting to the module. These functions were designed in the form of finite state machines using VHDL code before being synthesised into the targeted FPGA. The modules were also individually simulated before being simulated as a whole.

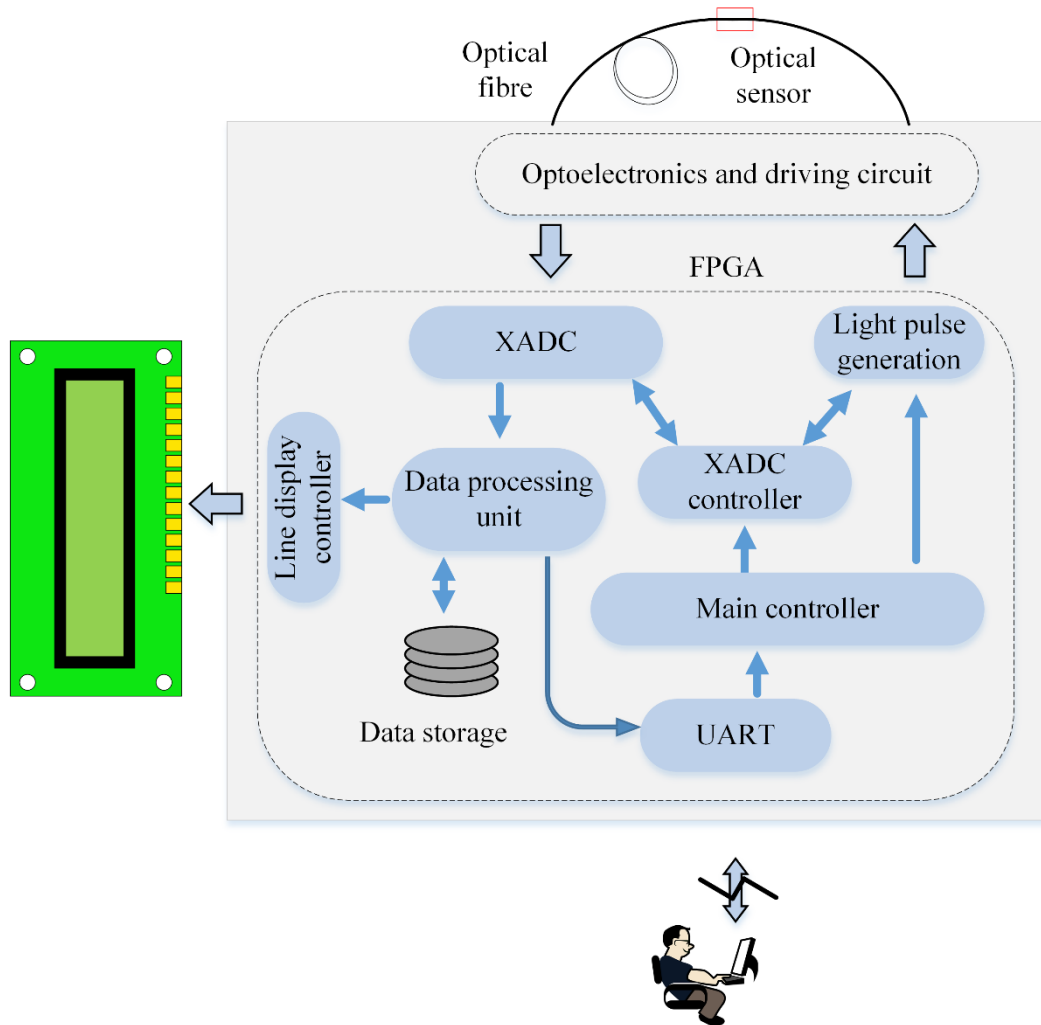


Figure 5-15 Block diagram for the FPGA.

5.4.1 Universal Asynchronous Receiver/Transmitter (UART)

The UART module was used for serial data communication. The UART's transmitter converts the incoming parallel data into serial data while the receiver converts the serial data back into parallel. The conversion is performed using a shift register. Figure 5-16 depicts the data packet for each transaction that only sends one byte (8 bit). It uses one 'low' bit as a lead bit to indicate an incoming data. Subsequently, the data transmission starts with the least significant bit to the most significant bit and it is followed by a 'high' bit as a stop bit. The data bit transfer rate is described using the set Baud rate. The module operated with a 115200 Baud rate and was used for communicating with the Arty board USB controller or the XBee radio module.

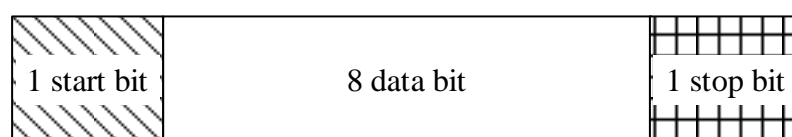


Figure 5-16 UART data packet.

In this work, the UART module was comprised of three subcomponents: (i) clock, (ii) transmitter (TX), and (iii) receiver (RX) as depicted in Figure 5-17. The UART clock was required to be set to 16 times faster than the required Baud rate. The clock signal was generated by applying a frequency divider to the FPGA’s 100 MHz input clock signal. Each serial data bit duration was 16 times of the UART clock cycle and this setting allows minute mismatch of clock cycle.

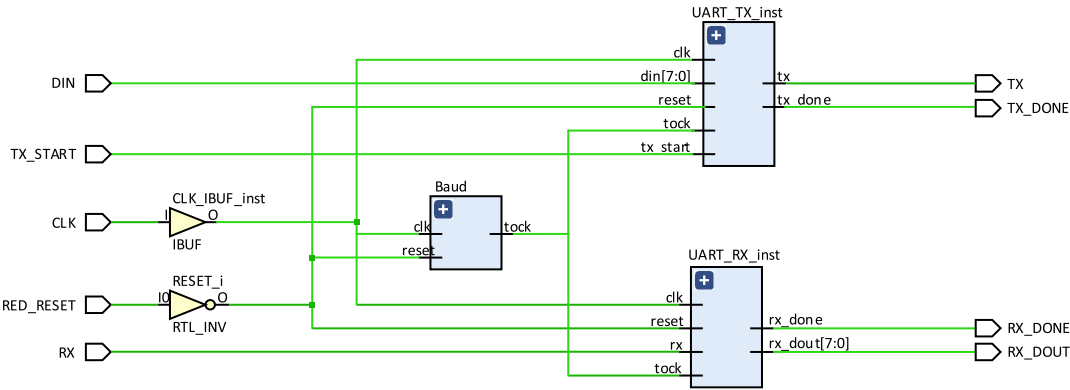


Figure 5-17 RTL code schematic for the UART module.

The simulation waveform diagram for both TX and RX is shown in Figure 5-18. The RX was set to always monitor on the incoming serial data for start bit. The receiver samples the incoming serial data at the middle point (tock_count =9) of each serial data bit. Once 8 bits is collected the RX assert ‘high’ on ‘rx_done’ signal to inform the main controller in this system. The transmitter, TX, transmit serial data once being initiated using ‘tx_start’ signal by assert a ‘low’ on the ‘tx’ as a start bit. Subsequently, each data bit was asserted for 16 UART clock cycles.

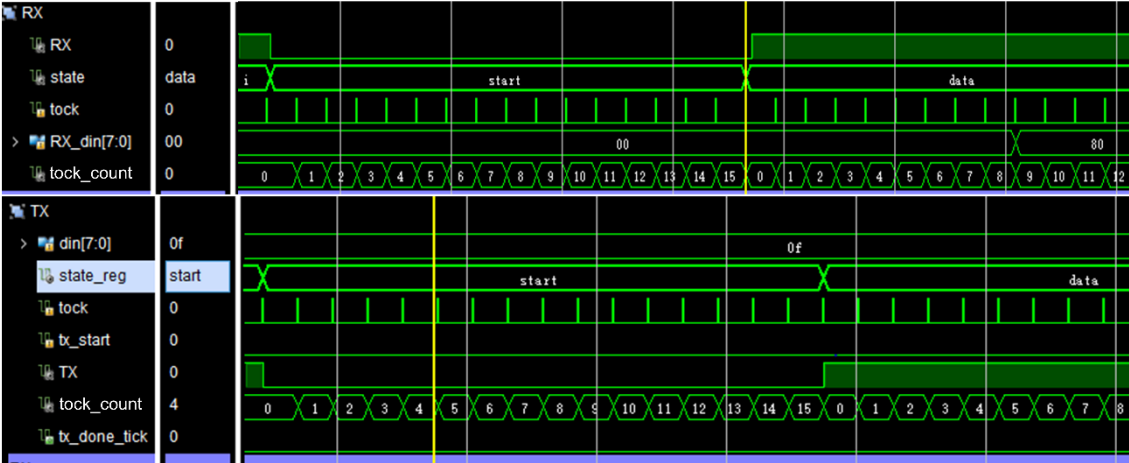


Figure 5-18 Simulation waveform of the UART module (RX and TX).

5.4.2 XADC controller design

In this work, the built-in Xilinx® ADC (XADC) was used to sample an electrical analogue signal. Within the Arty board, every single ended analog pin has a small sampling capacitor (1 nF) for

signal sampling. Similar to other ADCs, the accuracy of XADC relies on the accuracy of the voltage across the capacitor. Therefore, the behaviour of the receiver circuit needs to be understood. The minimum required time for the capacitor to reach the stable level is determined using the circuit simulation by using LTspice IV (Appendix B) and shown in Figure 5-19. A switch controlled by pulsed voltage source is included to examine the time that the capacitor requires to reach the needed resolution. The switching is set to be instantaneous for the simplicity of the analysis. The rising edge of the input signal is avoided by turning on the switch after the signal stabilised.

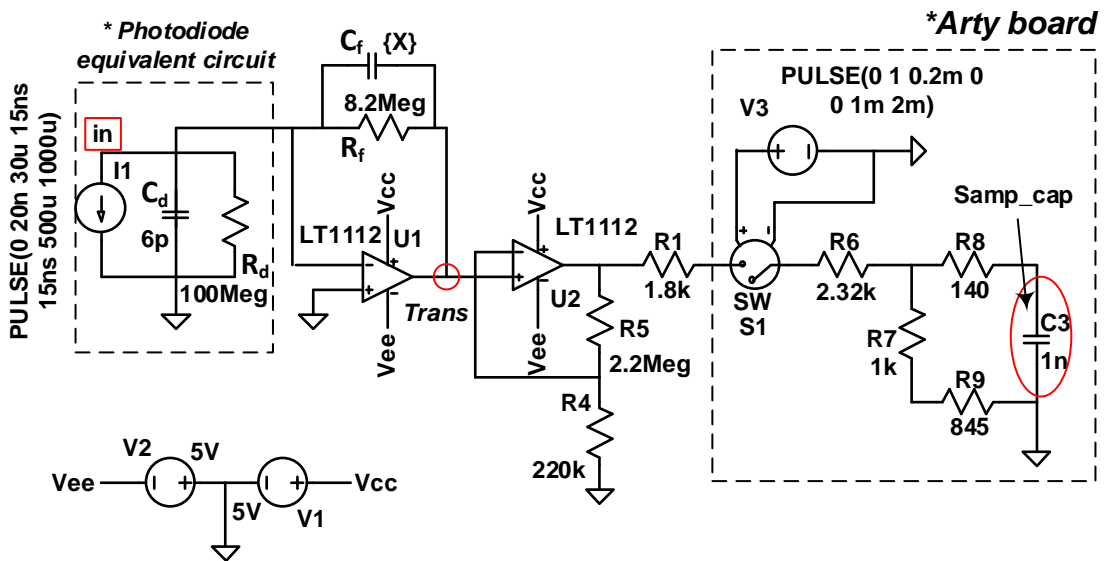


Figure 5-19 XADC sampling capacitor charging simulation circuit.

The input current pulse, transimpedance output and sampling capacitor response are shown in Figure 5-20. The switch is turned on at 200 μs after the transimpedance output stabilised. The sampling capacitor voltage takes 22.11 μs to reach the desired level (the settling error is smaller than one half of the least significant bit (LSB) of a 12-bit value). In other word, it takes 22.11 μs for the sampling capacitor to reach the correct voltage level. This delay is then taken into account during both XADC and its driver design.

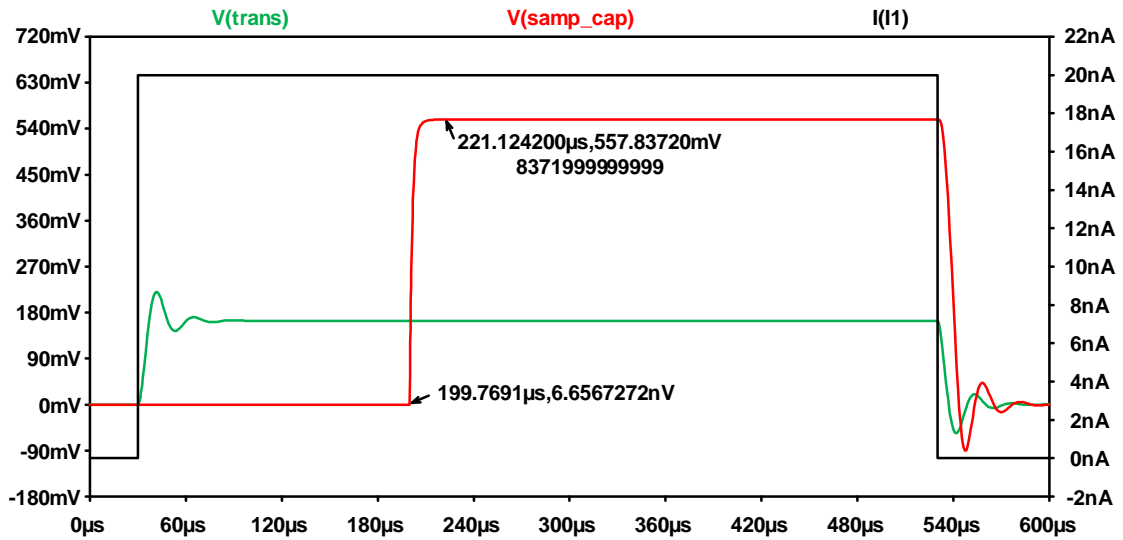


Figure 5-20 XADC sampling circuit simulation result.

In the reported work, the pulsed light source was paired with the XADC to implement the time domain intensity modulation scheme. The XADC was configured as event driven sampling mode and a controller (XADC controller module) was designed to regulate the XADC sampling time using trigger signal, *CONVST*. This allowed the user to initiate and stop the system operation. The module was designed to be aware of the LED switching by monitoring the light pulse generator's ON signal as shown in Figure 5-21. This was to avoid sampling during the signal transient. The module accepted the user input to register the total number of data to be acquired. XADC operation was also monitored for the sampling counter and conversion trigger. The controller also governed the data extraction for other modules. Data extraction on selected channel was performed by this module by monitoring channel address that was being sampled by XADC. The output data from the XADC is labelled using a 4-bit header to identify the colour.

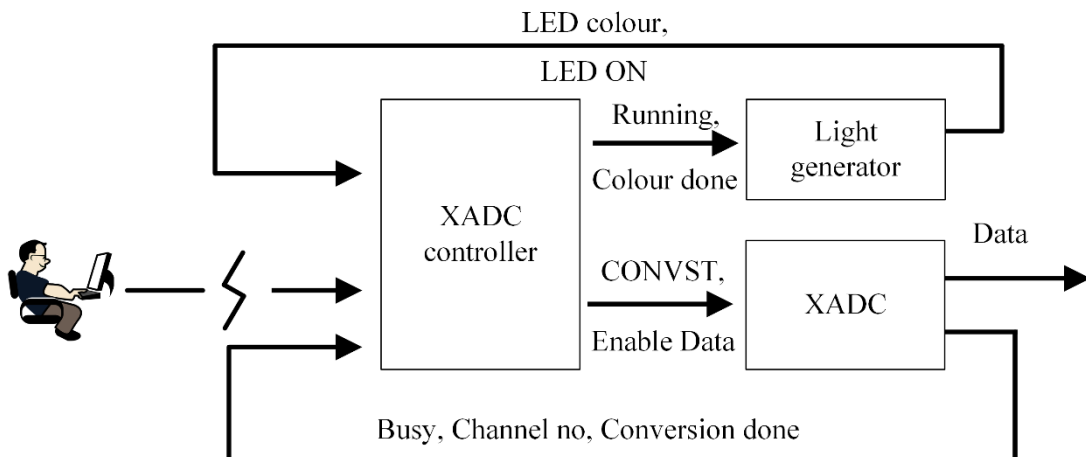


Figure 5-21 Simplified XADC controller module.

The XADC controller received the user parameter for the number of data points to be sampled and passed this value to the main controller. There are two counters for this controller, one for

the total number of sampled data points and a second for the number of cycles for the current colour channel. The controller asserts a 'running' signal 'high' if the number of data acquisitions was less than the user setting. Figure 5-22 depicts state timing and state diagram for the controller.

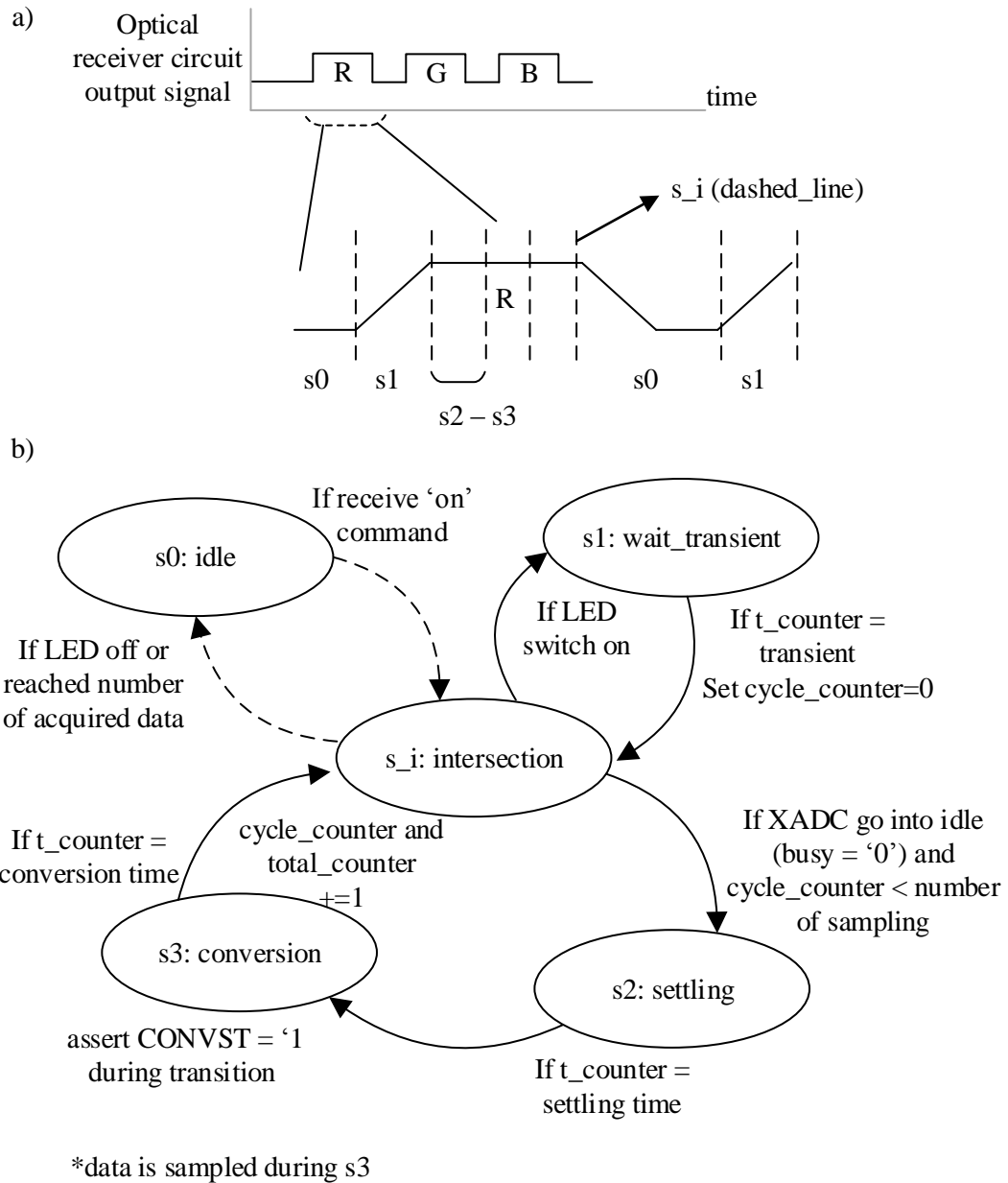


Figure 5-22 XADC controller a) state timing and b) state diagram.

The XADC controller has five states to regulate the XADC to monitor the photodetector circuit output. The state transition timings were dictated by both the XADC timings (settling and conversion timing) and the signal transient timing. An intersection state (s_i) was used as a decision point for the state machine to decide the next state (idle (s_0), transient waiting (s_1) or sampling cycle (s_2 to s_3)). Initially, the module remained idle (s_0) until the LED was switched on and then the state machine transitioned into a waiting state (s_1) to wait for the signal to stabilise. After the transient period, the controller entered the a sampling cycle (s_2 to s_3) to sample the

selected single colour channel. The transient timing are dictated by the both XADC timing (settling and conversion timing) and the transient timing. For a single sampling cycle, the controller waited for the sampling capacitor voltage to settle and then triggers the XADC conversion at the end of the conversion counter. The voltage reading was then sampled at the end of the conversion state and passed to data processing module as shown in Figure 5-31. This cycle restarted three times before sending signal to the light pulse generator module to inform the current colour acquisition cycle is complete. Multiple sampling per colour channel was used to reduce the transient waiting. The cycles restarted once the LED was switched on again.

5.4.3 Light pulse generation

Time domain intensity modulation scheme was utilised here to measure three wavelength regions of interest using a single photodiode. This scheme was strongly dependent on the synchronisation of both the generated light pulse and the photodiode sampling time. It was designed to synchronise and regulate the LED outputs. The hardware module is shown in Figure 5-23. The colour channel switching was performed using a combination of the counter and a demultiplexer. The output signal then forwarded to the selected colour channel. The output signal was adjustable through pulse width modulation (PWM) to control the LED brightness. The PWM duty cycle could be changed using the PC's software interface during the operation. The module also provided the LED's colour information for data labelling.

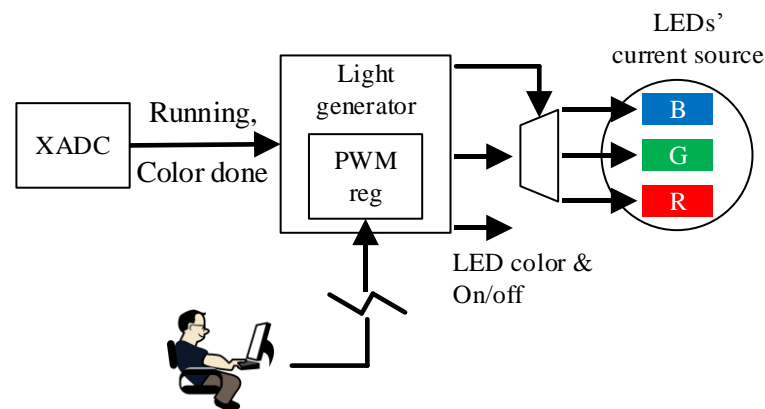


Figure 5-23 Simplified light pulse generator module.

The “color done” signal generated by the XADC was used by the light generator module in order to keep track of the sampling timing. Sampling signal tracking instead of hard-coded counter can reduce resource usage and minimise update during parameter change. To regulate the LED light intensity control signal, the receiver circuit transients, including raising edge and oscillation, needed to be taken into account. To reduce the total idle time for the rising and falling edges, each colour LED was switched on for a period of three XADC sampling periods instead of a single sampling period only. An indicator signal was used to inform the XADC driver that the

LED was turned on and subsequently delays the XADC sampling by the required time. That was to avoid the rising edge transients.

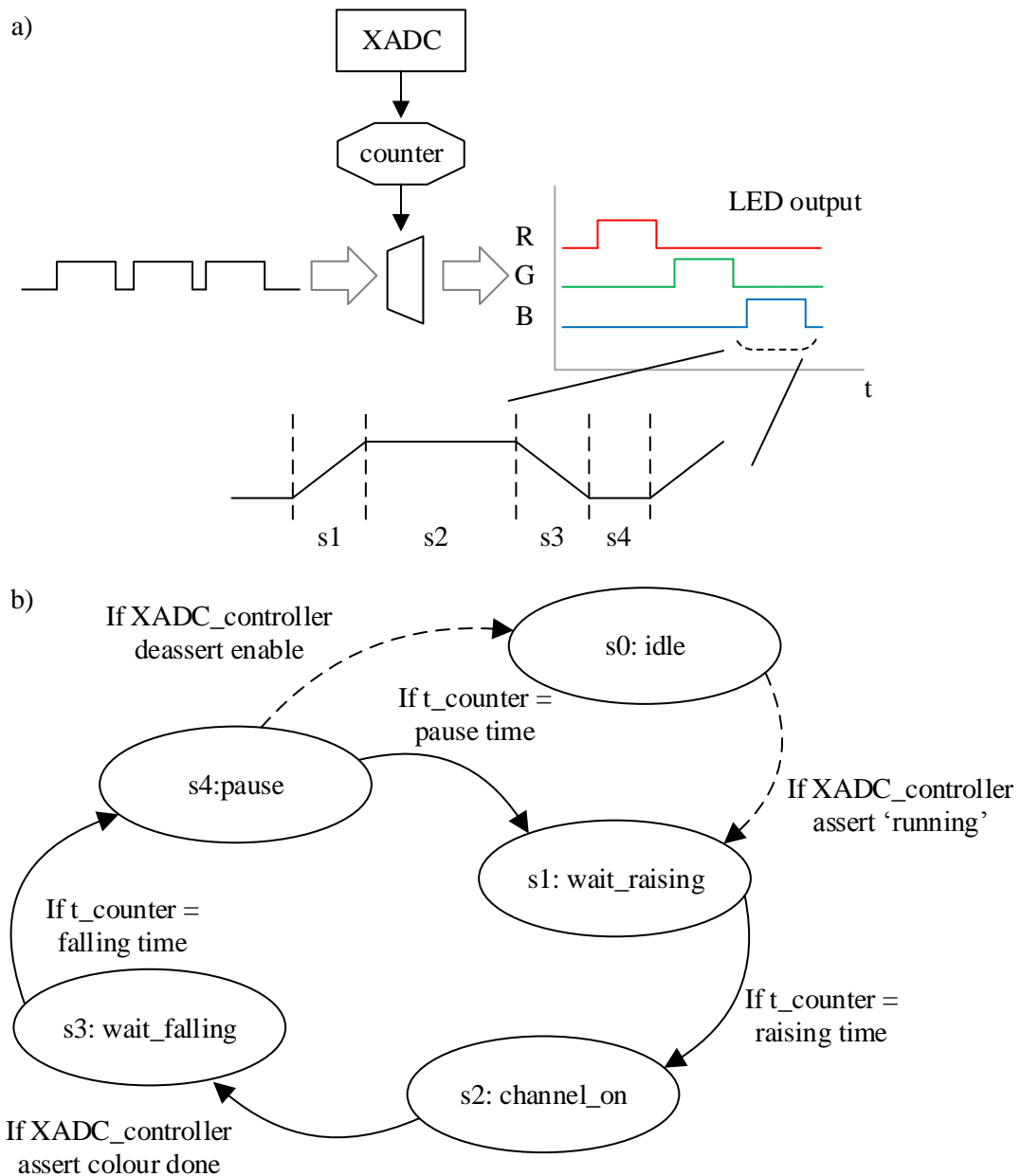


Figure 5-24 Light pulse generator a) light channel selection mechanism and state timing. b) state diagram.

The light pulse generation module used five states to provide the required light and synchronise with the XADC controller. The module goes into a waiting state after the XADC controller asserts a 'running' signal. After the light output has stabilised, the module signals the XADC controller to indicate the light output (colour channel) is switched on. Once the XADC controller collected the required number of samples, the light pulse generator switched off the output and waited for the signal falling transient. The cycle is restarted with another colour channel if the XADC controller's enable is still asserted.

5.4.4 LCD display controller

An LCD display controller module that comprised of two components was designed to provide the readings to the user as shown in Figure 5-25.

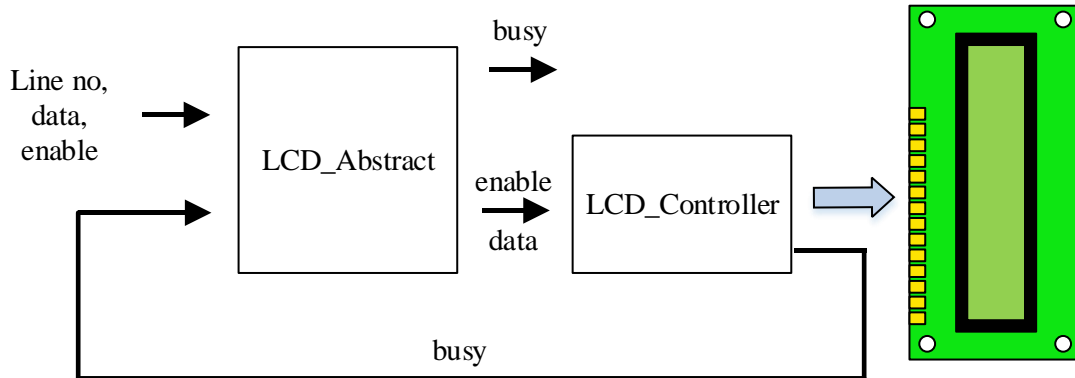


Figure 5-25 Simplified LCD controller module.

The LCD was initialised to have four lines display where each line can contains 16 characters. It was designed to display the XADC output up to a maximum value of 4095. The line can be selected using the display's place cursor position instruction as shown in the user manual (a link to the user manual given in Appendix E). To simplify the data entry process for single line, a layer of abstraction was created on top of the controller to display data in hexadecimal number format. This create a more readable code style for designer and the state transition diagram for this design is shown in Figure 5-26. The component, LCD_abstract, set the cursor at designated position at first. Then, the component looped through characters from the incoming line and instructed the LCD_controller to display the data. The cursor then incrementally shifted to the next position once the LCD instructed to display one character. Therefore, the cursor position setting was only performed once.

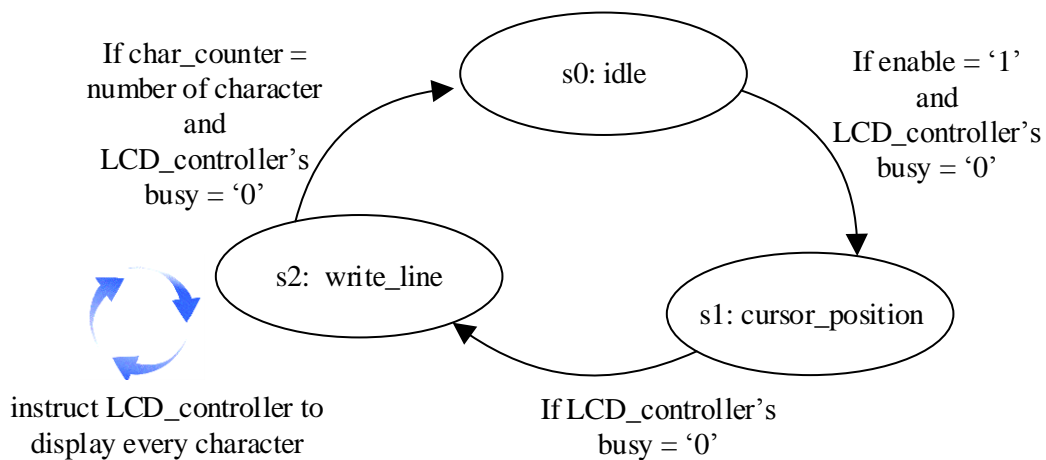


Figure 5-26 LCD abstract state diagram.

The FPGA was connected to the LCD using 4 data cables and required to break the eight bits input data into two nibbles (four bits). The nibble instruction routine as shown in Figure 5-27 was performed according to the user manual.

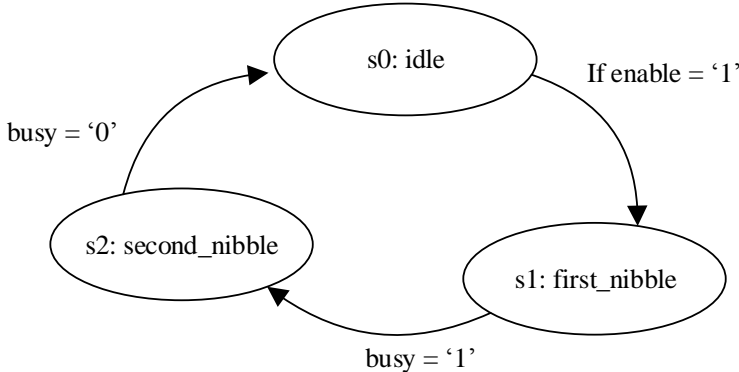


Figure 5-27 LCD controller state diagram.

5.4.5 Data processing unit

The data processing unit was designed to process incoming data from the XADC module. The module then instructed the LCD display controller module to display the data and the TX unit in the UART module to transmit the output data as shown in Figure 5-28. The module also incorporated the later included classification module.

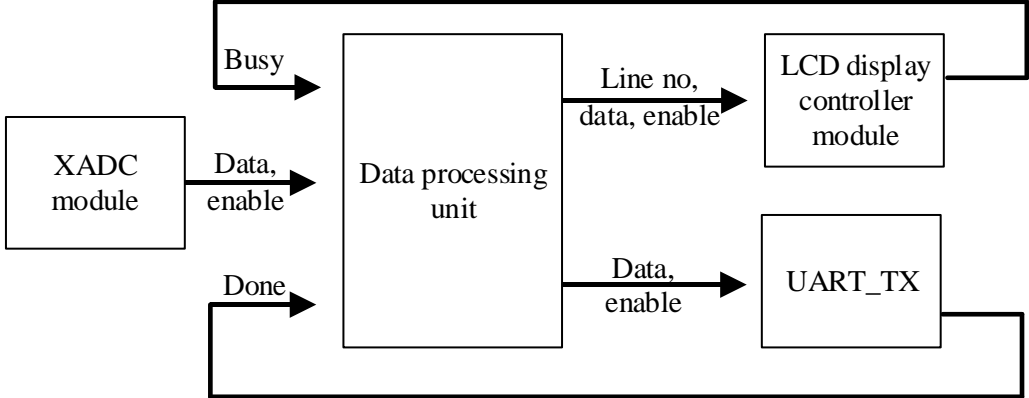


Figure 5-28 Simplified data processing unit module.

There are two main processes within the module, the process for UART transmission and the process for the LCD display as shown in Figure 5-29. For UART transmission, the module breaks the incoming data into two bytes and sends the bytes consecutively to UART_TX unit. The module waits for the UART_TX unit to assert a 'done' signal to high before forwarding the data and enabling the UART_TX. The module waits for the LCD controller module to initialise before shifting into the 'idle' state. The module takes in the XADC module's output data that includes header data. From the header, this module selects the predetermined LCD display line and forwards the data for display. The process repeats if there is data and the LCD is not busy.

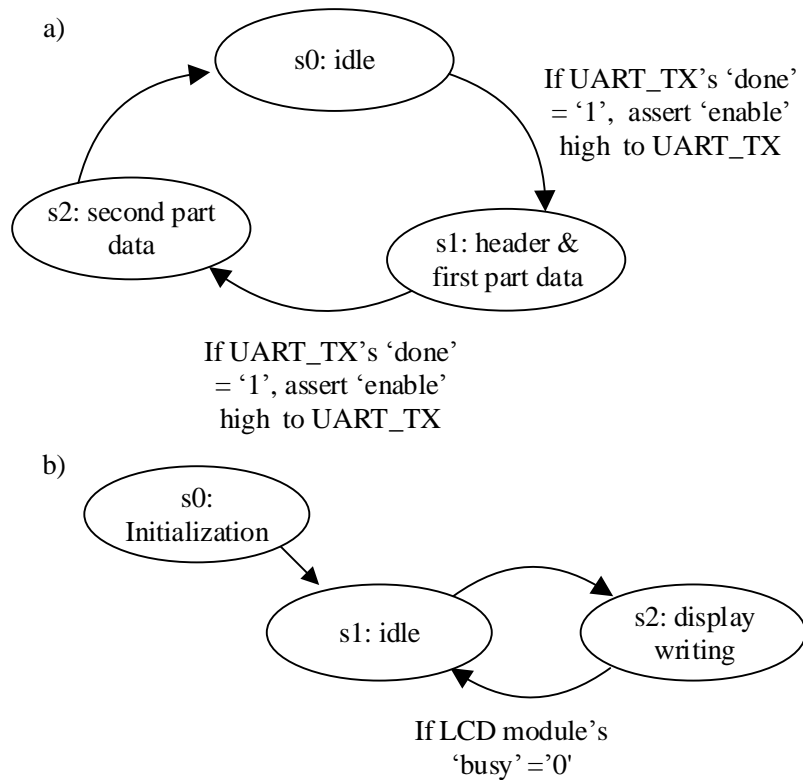


Figure 5-29 LCD controller state diagram for a) UART transmission and b) LCD display.

5.4.6 System summary

Previous sections discussed the designs of different individual core modules for the sensor system. These design descriptions in VHDL were synthesised into circuitry for the targeted FPGA (Artix-7) using Xilinx® Vivado and the RTL description schematics for the system components including the top level are shown in Appendix G. The resource utilisation of the system implementation comprised of all modules including the kNN modules in latter section is presented in Figure 5-30.

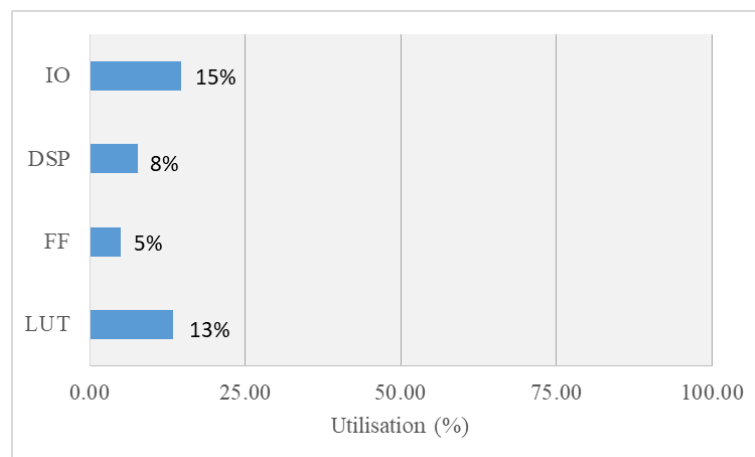


Figure 5-30 Resources usage for the sensor system.

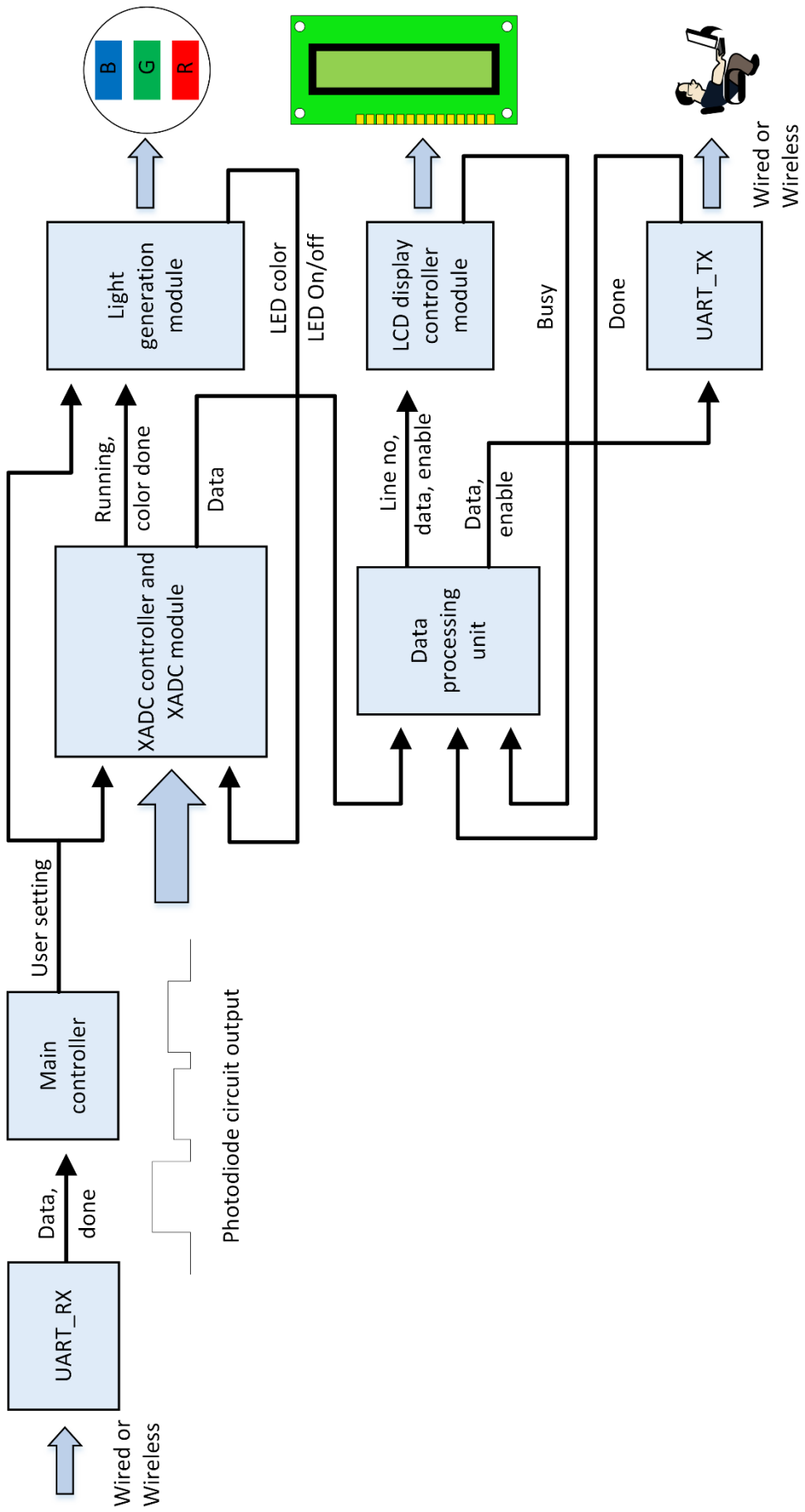


Figure 5-31 Simplified digital design system module connection.

The Artix-7 FPGA used has 20800 slice LUTs, 210 I/O's and 90 DSP slices available to use. Each slice LUT is a 6-input look up tables (LUT) for logic use. It is coupled to a pair of flip-flops that functions as registers. The total resource usage for the whole system developed is lower than 20% in every resource type. This allows the system accommodate more sensor types and/or sensors. The sensor system design consists of 5 main modules: (i) UART module, (ii) XADC and XADC controller module, (iii) light generation module, (iv) data processing module, and (v) LCD display module as shown in Figure 5-31. The dependencies between modules that are shown in the figure were discussed in previous sections. The figure illustrates the top level of the design that completed with signal flow between modules and external signals include photodiode circuit output and communication signal. Communication between user and system can be performed using XBee (wireless) or USB (wired) connection through UART module.

5.5 Interfacing with a PC

The sensor system was controlled by a PC user using a software script written using Python open source scripting language. The developed software depended on several Python modules including PySerial, NumPy, Pandas, Tkinter and Matplotlib. A graphical user interface (GUI) was developed for a simple interface for system control such as data acquisition initiation and system parameters setting. The GUI was designed using object orientated programming method. From the parameters setting, the user could manipulate the LED light intensity and the label for data collection. The number of data acquisitions per session was defined within the software script. Figure 5-32 depicts the GUI process that is identified within the dashed line box. The region also includes two other sub-processes: (i) acquisition initiation, and (ii) acquisition termination.

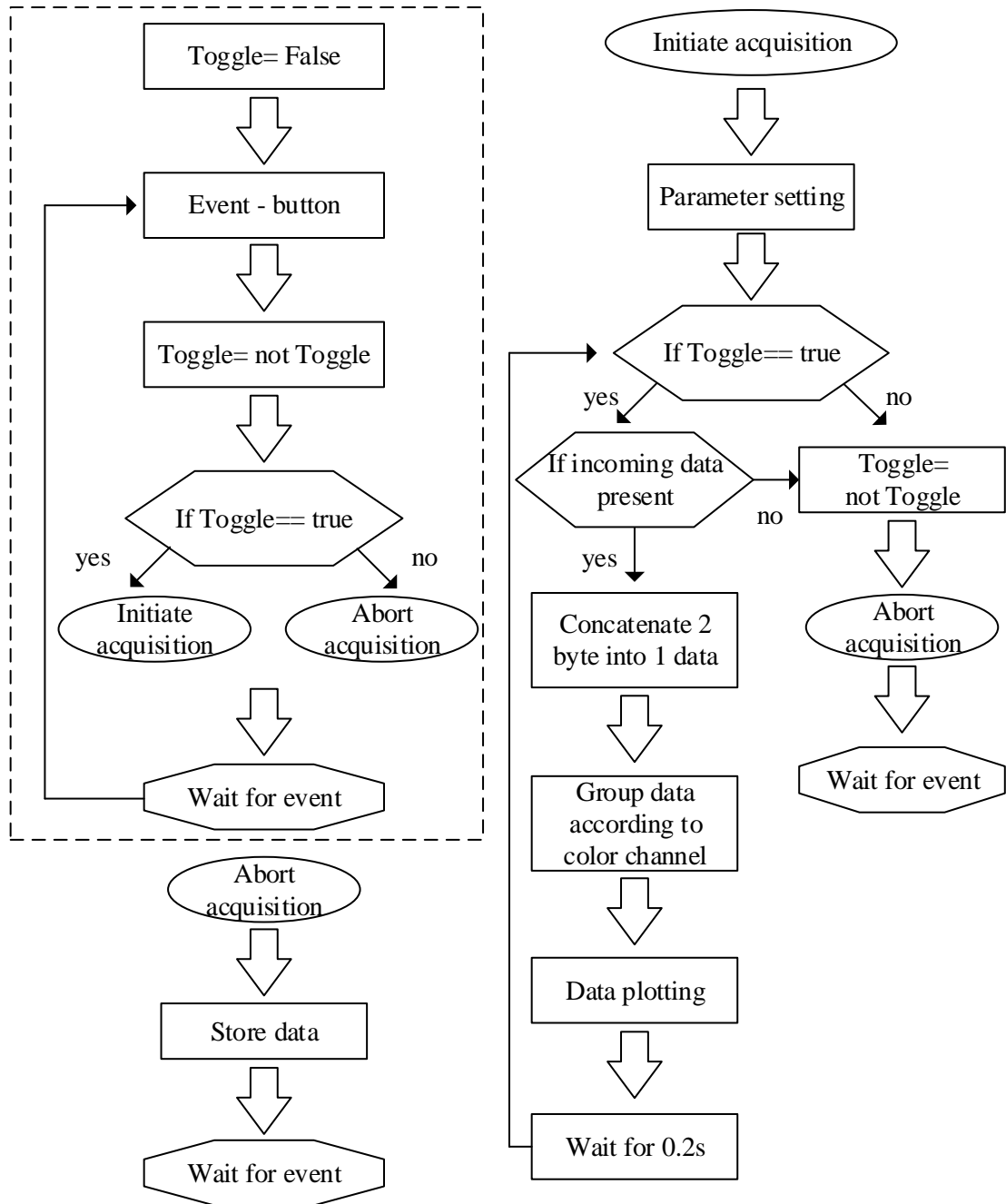


Figure 5-32 Data acquisition and termination process.

The GUI operation starts with a toggle variable set to a false value that indicates idle system status, and awaits a GUI event. The GUI captures events such as button being pressed. The data acquisition process is initiated by setting the toggle variable to true using the ‘start’ button. It can be terminated using the ‘stop’ button. The acquisition process starts by sending the defined system parameters to the FPGA board from personal computer. The incoming data from the FPGA board to the PC, such as ADC data, was 12-bit data and was divided into two bytes to fit the UART data transmission requirements as shown in Figure 5-33.

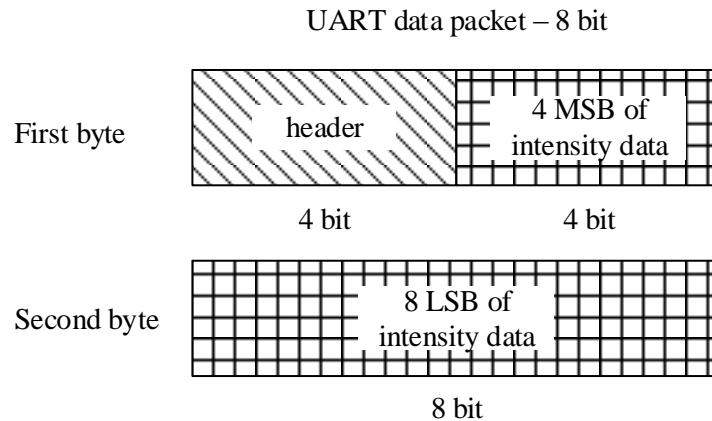


Figure 5-33 Incoming data packets and its content.

For intensity data, the first byte contained the metadata as the header (four most significant bits) and part of the intensity value (four most significant bits of 12-bit intensity data). The second byte contained the other part of the intensity value (eight least significant bits). The metadata indicated which module the data was produced from and its label (e.g., LED colour). Other data, such as classification result, was also transmitted in similar fashion but may only use less than 12 bit data and other bits filled with zero. The two intensity byte data were identified through the header and combined as a whole. The complete intensity data were then plotted in a rolling display. The data transmission was performed in batch as all the incoming data could not fit into the buffer. The data was collected by the software continually by invoking the collection function after each collection (using the *after* function from Tkinter module). The data collection ceased when there was no incoming data present as shown in snippet below so the timing parameter could be omitted. This reduced script editing whilst maintaining system flexibility to accommodate different XADC timing requirement. The acquisition process was terminated when no more incoming data were present. The acquired data were then stored in the system database.

```

Class process():
Def __init__(self):
    self.toggle=false

Def cycle(self):
    If self.toggle:
        If Data_present:
            self.incoming_data_processing()
            self.root.after(0.2s ,self.cycle)
        elif not Data_present:
            self.toggle=False
            self.abort()

```

```

## for single color example
Def incoming_data_processing(self):

    ## locating data with header and without header
    data_w_header= self.incoming_data[::2]
    data_wo_header= self.incoming_data [1::2]

    ## locating data for specific color LED
    red_index=np.where((data_w_header >=16) &
    (data_wo_header <2*16))[0]

    ## splicing of two data
    data= (data_w_header %16 *256)+ data_wo_header
    red_data= data[red_index]

    ## plotting
    self.red_plot.scatter(time, red_data, color='red')

## abort the data acquisition and store collected data
Def abort(self):
    self.send_abort()
    self.export_data()

```

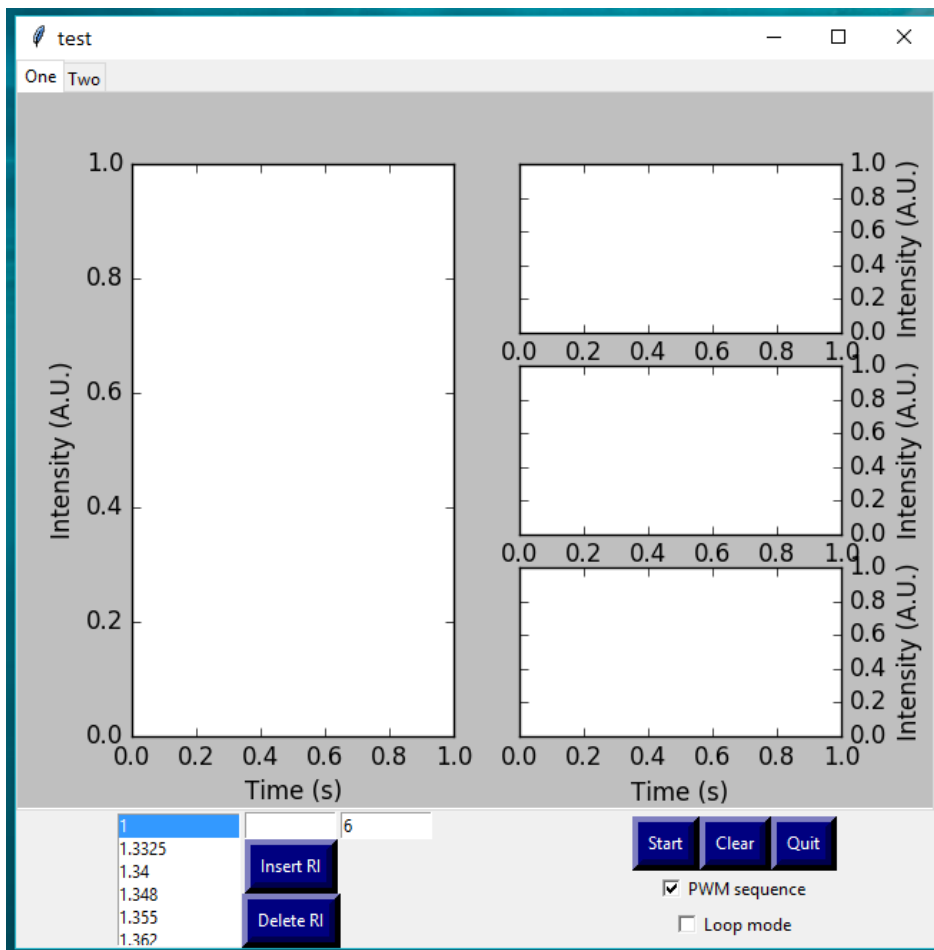


Figure 5-34 GUI for system control and display.

Figure 5-34 depicts the snapshot of the GUI for system control and data display. It consists of one main plotting area at the left where all the data is displayed. Three smaller plotting areas are included to the right where individual colour channel intensities are plotted. The lower left list is to label the data being acquired. This list can be updated by inserting or deleting the new values. The button 'start' allows the user to initiate the sensor system using the predefined parameters within the script. The 'start' button turns into 'stop' after initiation for user to abort the acquisition. User can clear the display data using 'clear' button. The 'Quit' button is to save the acquired data and quit the application.

5.6 System performance

The previous sections illustrated the development of each individual subsystem as well as the circuitry within FPGA. The circuitry is designed to interface with different subsystems. This section discusses the integration of these subsystems and evaluate the integrated system performance. This evaluation was conducted in the absence of the optical sensor influence (the optical fibre bypassed the sensor by connecting the LED output directly to the photodiode input). Two main test points - output of light source and the ADC output - were used to evaluate the system performance. The output of the light source was measured using a spectrometer (the Ocean Optic QE65000). The spectrometer's integration time was set to be 18000 μ s and the data averaged three times.

5.6.1 Optical source output

The light source used was designed to provide three different colours in sequence with adjustable intensities. The intensity of each colour was adjusted using PWM by manipulating the duty cycle of the FPGA's digital output. There are 16 levels for the PWM strength (PWM level from 0 to 15). The PWM period was 150 ns and each level of the PWM code gave an increment of 10 ns to the pulse width high time. The LED output behaviour under different conditions was examined using a spectrometer. The three colour channel's spectra are integrated over the each band to emulate the intensity collected by the photodiode. During the dimming test, the light output intensity was manipulated using PWM as shown in Figure 5-35. The scatter plots for the three colours exhibited a good linear behaviour as the linear fitting correlation coefficient is close to unity.

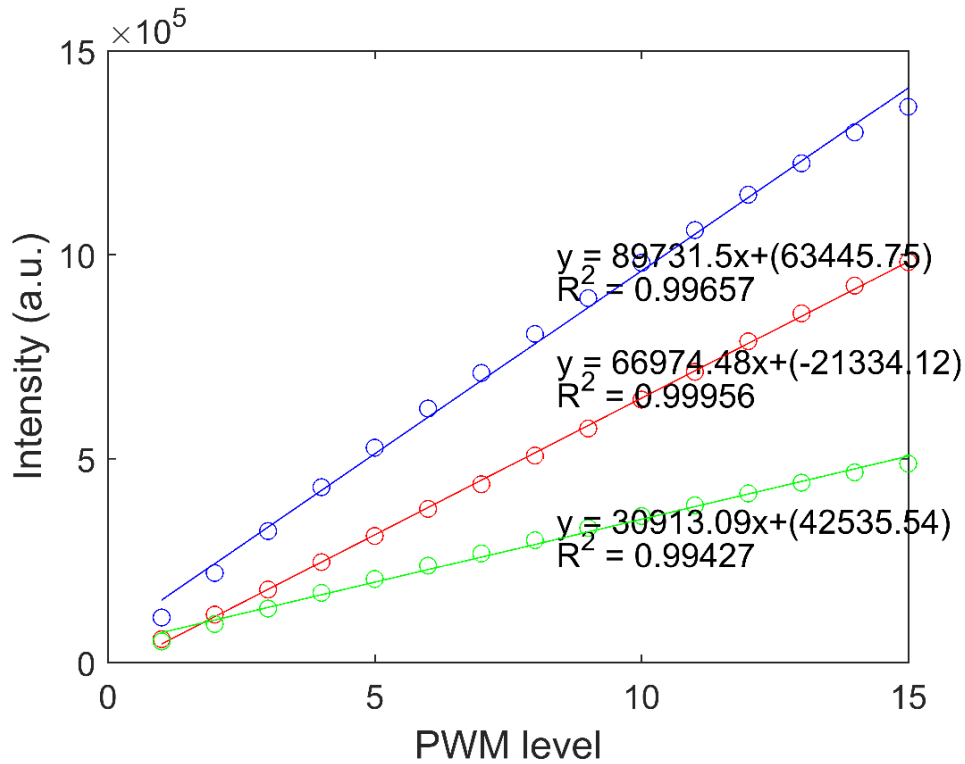


Figure 5-35 Output intensity (a.u., arbitrary unit) as function of different pulse width modulation level (min – max = 0 – 15) for red, green, blue LED indicated by their line colour.

The peak wavelength was used to evaluate the spectrum stability for different PWM levels as shown in Figure 5-36. The peak wavelengths were extracted from the spectra. The red LED's peak wavelength exhibited a red shift whilst the green and blue LEDs exhibited blue shift. This behaviour has been reported by studies on dimming using continuous current reduction method which was claimed to be due to material properties [93]. The continuous current reduction method manipulates the current DC level during the dimming instead of the LED duty cycle. In this scheme, PWM was applied however its output resembles a fixed current rather than pulse waveform. This is because the current source response rate is restricted by the op-amp's slew rate. The slow response rate is explained in Section 5.3.2. To support the FPGA's PWM signal (with the FPGA operating at 100 MHz), this required careful signal path planning and a high-speed op-amp.

The colour shift effect is not significant for the current application as the sensor response is a broaden dip but it may affect applications that require better spectrum stability. The red shift exhibited by the red LED could be due to the reduction of the bandgap energy when junction temperature increases (the heat generated by the LED itself) [94]. An additional dominating mechanism acts on both green and blue LED which causes blue shift. Different shifting direction of the peak wavelength could be due to the material difference, red LED is fabricated using AlGaInP material whilst blue and green LED are InGaN-based [93]. The continuous current reduction method has also been reported in the same work that provides better intensity linearity

than duty cycle manipulation. This behaviour needs to be understood for applications which are sensitive to spectrum change.

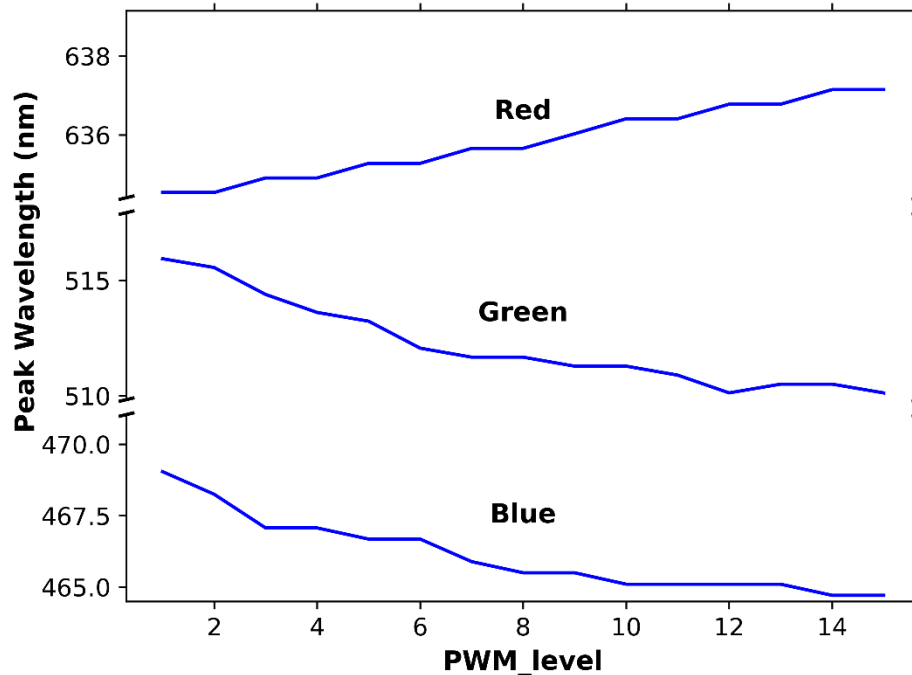


Figure 5-36 Peak wavelength of red, green, and blue LED for different PWM levels.

5.6.2 System response assessment

The system output was assessed without using an optical sensor (bypass) to evaluate the system performance in isolation. The LED's output was coupled to a photodiode using a single plastic optical fibre. The sampled photodiode output by XADC is shown in Figure 5-37. The XADC was able to sample voltage levels from 0 to 3.3 V and its output ranges from 0 to 4095. The LED brightness was fixed at the highest level (15) and it was monitored for 17 s for each colour. The intensity in the y-axis represents the XADC's output code value (12 bit). A sheet of paper was placed between the LED and the optical fibre to act as a light attenuator to prevent saturating the XADC. The output shows noise and transient behaviour for red LED. The noise is filtered out by averaging over 24 data points as shown in the figure. This results in a smooth signal and gives a consistent value over time. The decreasing trend over time for the red LED intensity that persists after averaging could be due to thermal heating. [95] However, it is not significant as the overall drop is less than 0.5% of the average. The drop is found using the difference between the mean of first 10 averaged data and mean of the data from 7 seconds to the end. This dropping trend can be omitted by considering data between 7 and 10 seconds only. For the classification operation in next chapter, a small amount of data (24 data points) were required to classify the measurand. The classification was performed remotely in the system and the PC only received the classification output. This can minimise power consumption when using wireless data transfer

and system operation. During the prototyping phase, all the raw data are sent to the PC for the purpose of developing better understanding of the system.

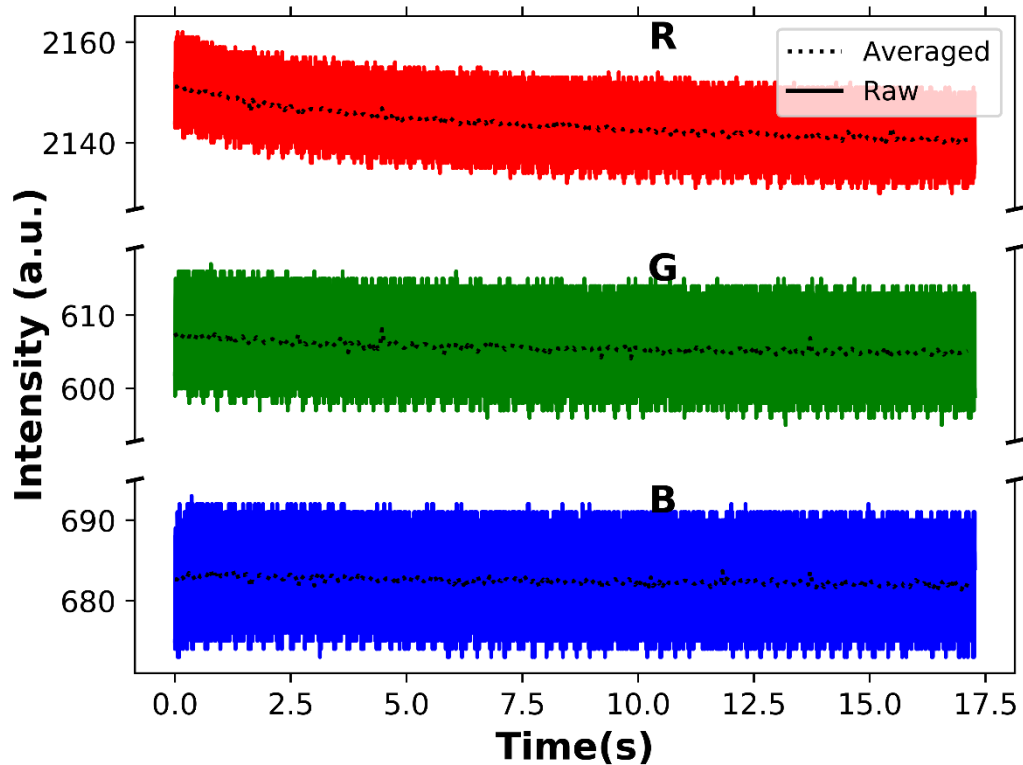


Figure 5-37 Time series of LED output data (a.u. = arbitrary unit) measured using the system (dash line and solid line represent averaged data and raw data, respectively).

To understand the noise present at the LED output (Figure 5-37), the red output from 7 to 8 second was enlarged and is shown in Figure 5-38. The noise is appeared to be a repeating pattern with frequency around 50Hz that was coincident with the power supply line frequency (46 Hz in actual count). To aid the peak counting visually, two shade of colours were added alternatively where each shade covers 10 peaks. The deviation in frequency could be due to software acquisition that was not optimised for real time keeping. The noise also appeared to have jagged curve (not repeatable) that could be due to random noise. The dominating noise have peak to peak noise around 20 (max 4095) that equivalent to 16 mV out of 3.3 V range. This noise is subsequently minimised further using accumulation method as discussed before.

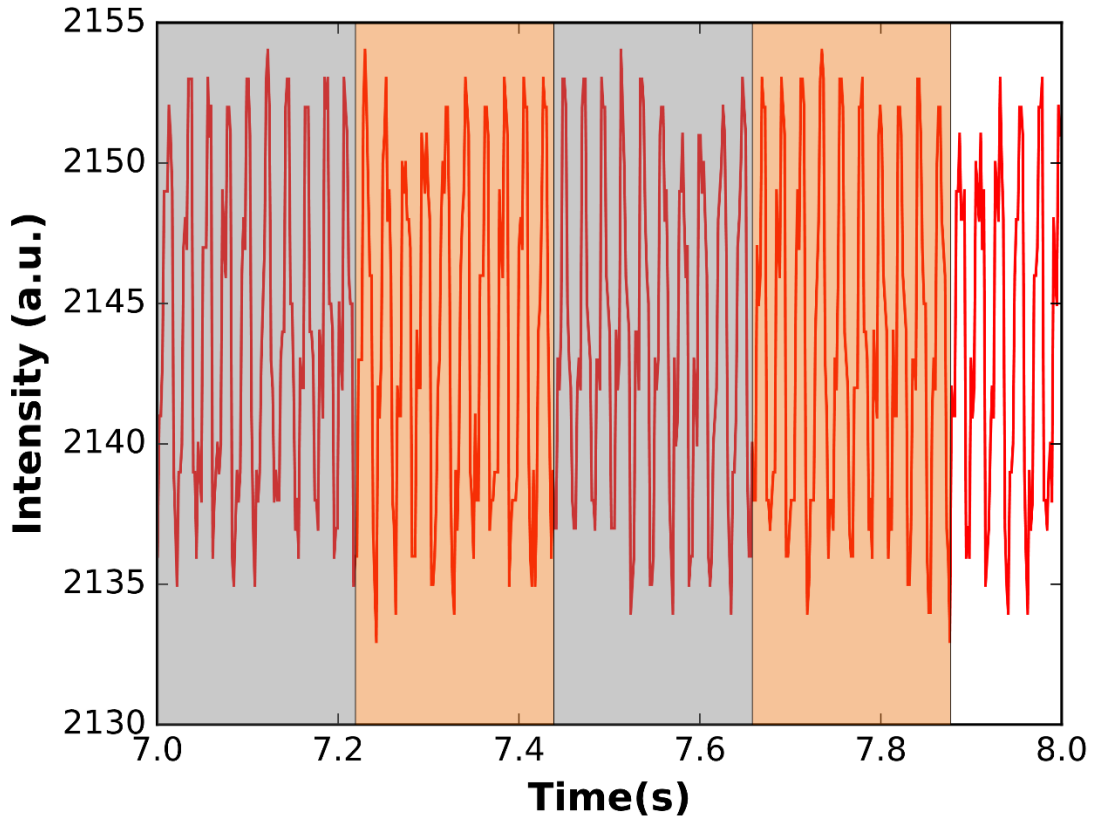


Figure 5-38 Zoomed-in plot for time series of the red LED output.

The system output repeatability was evaluated to assure the repeatability of the classification process later. The repeatability test was proceeded by having data acquired for 3 times with each session having 10 minutes in between as shown in Figure 5-39. The decreasing trend in red colour was persistent for 3 different acquisition sessions and the drop was around 9 with a standard deviation of 0.071 for this 3 sessions, which is negligible. The standard deviation between sessions was found by working out the average of the highlighted 3 second period in the figure for each session and then determining its standard deviation for the colour averages. The highlighted period is between 7 and 10 s. The standard deviation for R, G, and B colour's average in this highlighted period are 1.560, 0.1488, and 0.1446 respectively. This output repeatability is acceptable for both intensity interrogation and classification given the standard deviation less than 1% of the output.

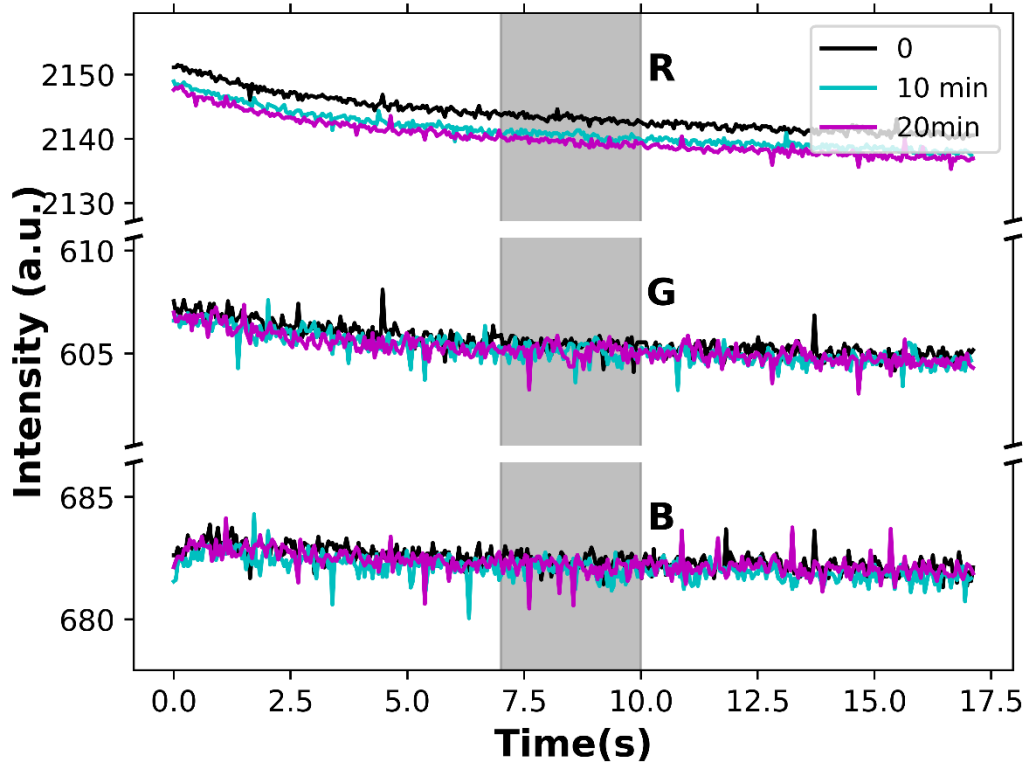


Figure 5-39 Output repeatability assessment. (a.u. = arbitrary unit)

5.7 Summary

In this chapter, the SPR sensor behaviour was discussed along with the time domain intensity modulation scheme. The modulation scheme was chosen to utilise the components (RGB LED and photodiode) which are widely available in the market. A low cost 3D printed structure was used for optical alignment and structural integrity. To implement the modulation scheme, a number of subsystems were designed and mounted onto a PCB. These subsystems are designed in isolation and tested through both experimental and simulation. The corresponding modules within FPGA were designed to coordinate the subsystem according to system behaviour. The sensor system could be controlled by the user using the developed software. The software was developed using Python language and equipped with GUI for ease of use. Finally, the system was tested as a whole without optical sensor to evaluate its performance.

Chapter 6 Results and discussion

6.1 Introduction

The early stage of this work focused on building a reliable readout system for the surface Plasmon resonance (SPR) sensor. Following the system performance evaluation, as discussed in the previous chapter, the SPR optical fibre sensor (OFS) was integrated to make a complete working system.

In this chapter, a classification approach (k-nearest neighbour (kNN) classification) is discussed and was used to correlate experimental measurements to actual refractive index changes. To apply kNN classification practically in the embedded system, two approaches were adopted: (i) pre-processing, and (ii) hardware acceleration. These were selected in order to reduce both memory requirements and time complexity. The classification algorithm was initially tested and validated using a script written in the open source Python language (on PC) before the corresponding hardware block is developed in VHDL (VHSIC hardware description language (HDL)). To accelerate the kNN algorithm execution, a scalable pipelined hardware design for the kNN algorithm was developed in VHDL. The FPGA hardware design was created using the Xilinx® Vivado integrated development environment (IDE) and verified using simulation before synthesis. The overall embedded data processing architecture depicted in Figure 6-1.

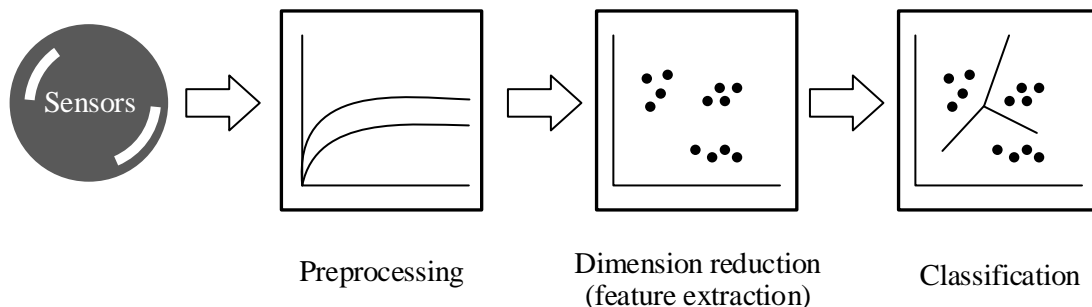


Figure 6-1 Embedded data processing architecture.

In the following sections, the system design operation is assessed and compared with the test results from the spectrometer. To prepare the sensor system for an embedded system application,

the combination of the kNN classification and clustering using representative (CURE) was applied. The CURE method is adapted as a positioning adjustment variant of prototype generation method (Section 4.3.4) which comprises prototype selection and position modification. This combination (CURE and kNN) was tested to determine the influence of the number of representatives on the kNN's 'correctness' (or generalisation error). After the suitable number of representatives was determined, the kNN algorithm was described in VHDL to form a component. Part of the component was constructed as a scalable pipelined building block. The classification component is placed within the data processing unit module (Figure 5-31). The system was then tested in real time with its classification capability.

This chapter is structured as follows. Section 6.2 discusses the sensor response recorded by the sensor system. Section 6.3 discusses the classification validation on personal computer. Section 6.4 discusses the classification implementation on FPGA. Section 6.5 summaries this chapter.

6.2 Sensor response

In the previous chapter, the system's operation (without the sensor) was proven to be reliable. To understand the sensor system behaviour, two sensors with different thicknesses (Sensor A: 50 nm and Sensor B: 70 nm) were tested with different weight ratio of glucose in water (different refractive index). The solution is drop on the sensor surface using plastic pipette and the solution coverage on the sensor is fixed using a tape with hole in the middle (Figure 1-2 (c)). The system testing was performed at every stage to examine the influence of each component. The solution were prepared by mixing glucose with deionized (DI) water at different weight ratio and sonicated for 3 minutes to ensure the solutions are well mixed. Deionized water is used to standardise the water content (without other ion species). The refractive index of the solutions were measured using refractometer and tabulated in following table (Table 6-1):

Table 6-1 Solution's refractive index.

Solution	Weight ratio (glucose to water)	Refractive index
Water	0:100	1.3312
G1	5:100	1.3378
G2	10:100	1.3436
G3	15:100	1.3514
G4	20:100	1.3578
G5	25:100	1.3633

The sensor output was recorded using both spectrometer (Ocean Optic QE65000) and the proposed system (tricolour LED, photodiode controlled by FPGA system) for comparison. Figure

6-2 depicts both system's ADC's output and spectrometer output for Sensor A (50 nm gold thickness). The LED intensity level was set at 15 (highest level).

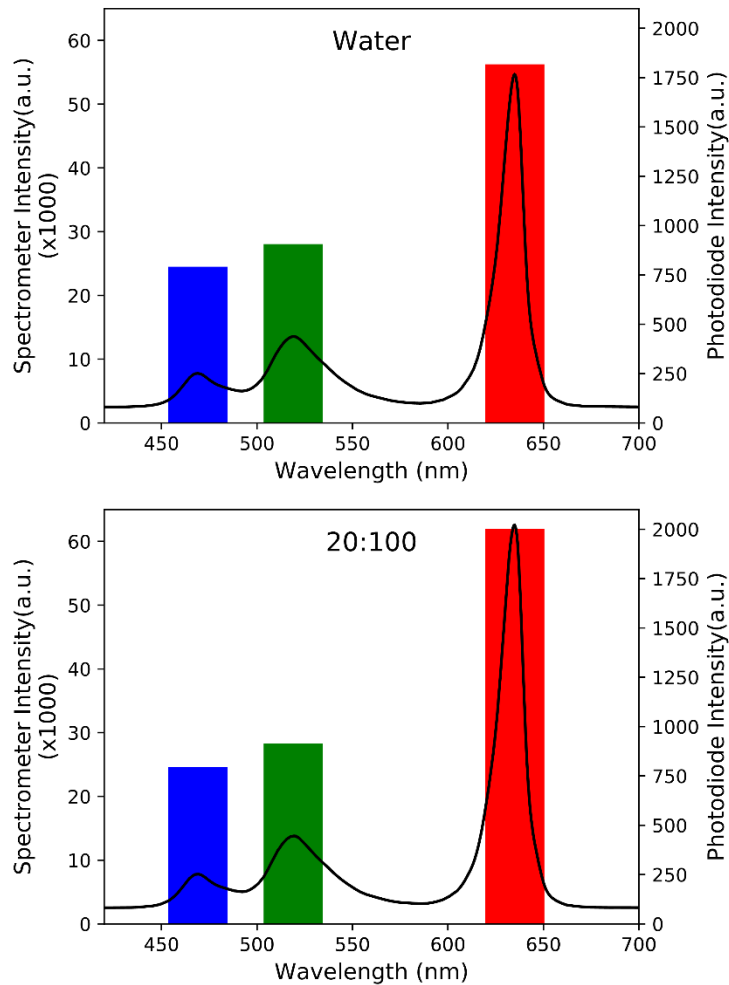


Figure 6-2 Sensor (50 nm gold thickness) response(arbitrary unit, a.u.).

The top and bottom plots show the sensor's output for measurements of water and solution G4. The solution was cleared from the sensor after every measurement and cleaned with DI water before the next measurement. The x-axis, wavelength, is for the spectrometer reading while the photodiode reading is shown in the overlapped bar plot for comparison. The measurements from both spectrometer and photodiode showed a good agreement on the change of the signal for different solutions. This trend agreement, however, did not produce the same magnitude of change (intensity change) that could be due to different spectral responsivity of the detector. The blue colour for both photodiode and spectrometer outputs exhibited a weak response to solution change and it was therefore suitable to use as reference signal. A simple pre-processing operation that utilised subtraction between red and blue (R-B) and subtraction between green and blue (G-B) was suitable to make use of the blue channel as reference signal. This simple subtraction also provided a self-referencing capability to minimise influence of external perturbation such as

persistent ambient light that commonly exists in three colours. This method also reduced the data from three dimensions to two dimensions. This reduced both computation complexity and memory requirement for classification use.

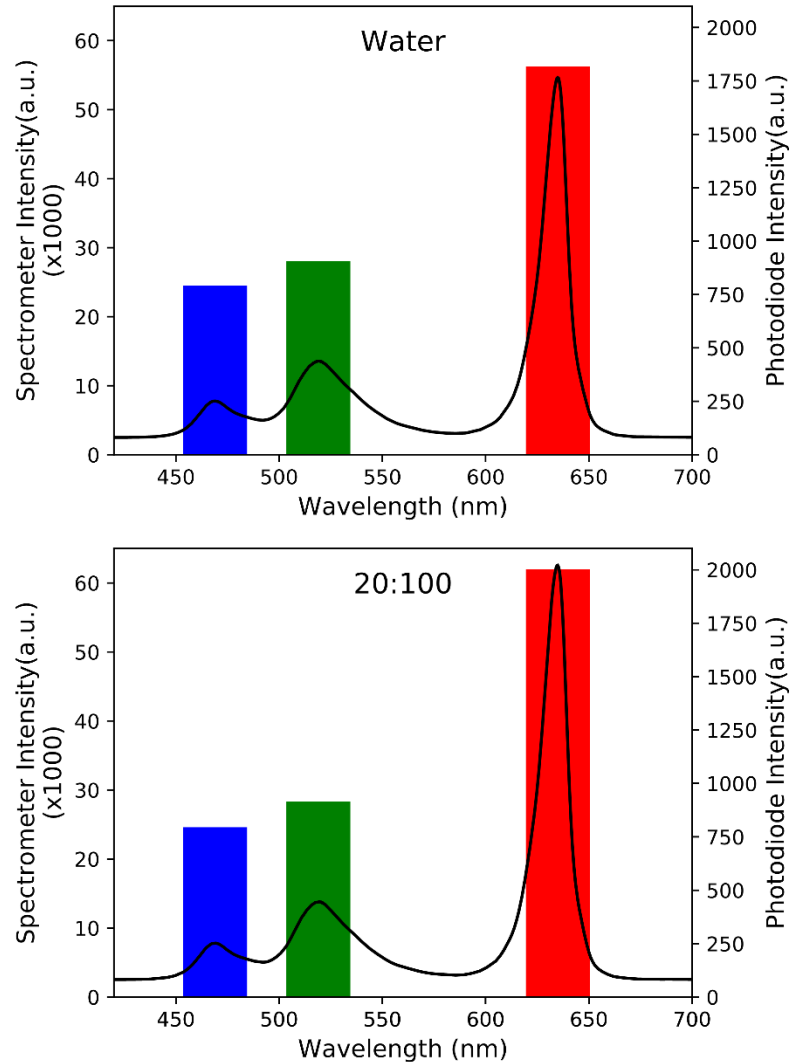


Figure 6-3 Sensor (70 nm gold thickness) response (arbitrary unit, a.u.).

Figure 6-3 depicts Sensor B (70 nm gold thickness) response and the LED intensity level was set at 6. The intensity setting was different for both sensor as the reading for Sensor B saturated at highest intensity level. The intensity difference could be due to difference in coupling condition from optical fibre to the glass slab. The sensor glass slab was produced using diamond cutter (handmade) and therefore small difference in dimension exists. The plot setting is similar as in Figure 6-2 plot and the sensor was tested with the same solutions. The measurements from both the spectrometer and the photodiode also show good agreement in the trend of the signal when the solution changes. However, two sensors produce different responses. Sensor A gives strong response in both green and red regions while Sensor B mostly responds to red region. The

difference in responses was due to different thickness of gold film but both the sensors can use the same dimension reduction scheme to reduce computation complexity and memory requirement. This allows most of the FPGA's configuration to be reused.

6.3 Classification validation using personal computer

In the reported work, a combination of the clustering using representative (CURE) approach [96] and the kNN approach was used to implement the data classification. CURE, a clustering method, was applied as an instance reduction pre-process method to reduce the training set size of the kNN algorithm needed whilst trying not to degrade its generalisation accuracy. The CURE approach was only required to be run once to generate the representatives set before being stored into the FPGA's memory. This reduction results in a shorter total execution time for the kNN classification. In this section, the colour channels output were first reduced into two dimensions. The reduced data were then fed into the classification input. The classification firstly took place in the PC for validation before its hardware counterpart was designed. The CURE approach is originally an unsupervised clustering approach that finds patterns in an unlabelled data set and then generates a smaller set size of representatives to replace the existing data. In this section, the clustering approach has been adapted as a supervised approach to exploit its capability to reduce both the memory and execution time requirements for the kNN. This however does not change the complexity of the kNN. The algorithm consists of two major parts: farthest neighbour and shrinking for every cluster.

CURE algorithm

Input: training set, number of required representative points (J), shrinking factor (α)

Output: Representative set

Algorithm pseudocode:

```
First part - farthest neighbour
BEGIN
Temp_l= [];
For each cluster:
##Data with same label fall into same cluster
    farthest_l= [];
    First point is randomly selected into farthest_l;
# For the rest of the J number of farthest neighbours.
    For i:= 2 to J:
        neighbour_l= [];
        neighbour_dist= [];
        Euclidean distance between farthest_l and training set is
        measured;
        Rank the distance;
```

```

# Find neighbours (nearest point from selected point).
    Find a neighbour (from farthest_l) for every member of
    the training set (remaining) and insert its coordinate
    into neighbour_l. (insert its distance into
    neighbour_dist);
    Rank the neighbour_l according to the distance;
# Find farthest neighbour.
    Find the largest distance between the neighbour pair and
    insert the coordinate into farthest_l;
    insert farthest_l into Temp_l;

Second part - shrinking
i:=0;
For each cluster:
#Clusters are separated into lists in Temp_l
    farthest_l =Temp_l[i];
    representatives= [];
    Find centroid for the cluster;
    Move every member in farthest_l toward the centroid by a
    fraction of  $\alpha$ ;
    Return representatives;
END

```

This adapted method generates a J-number (J is used here instead k as symbol k is used by the kNN classification) of well scattered points for each cluster using farthest neighbour. These points are then shrunk moving them towards the centroid by a fraction of α . These well scattered points are used to preserve the geometry of the cluster. The second step is to shrink the scattered points towards the centroid to mitigate the influence of the outlier data points on the kNN classification results. The outlier data points are typically placed far away from the cluster and thus the shrinking by fraction moves the outlier towards the centroid in larger magnitude while the remaining normal cluster members experience much smaller shift. The difference in the shift distance heavily discounts the influence of the outliers in the classification outcome. The shrinking also moves the representatives away from class boundary which could otherwise lead to misclassification.

This CURE clustering combined with kNN is a scheme between two extremes, centroid approach and plain kNN approach. The centroid based approach favours a data set with a spherical shape and utilises only single point as a representative for each cluster. The kNN without modification provides high classification accuracy by using large training sets. Data structures have been applied for other studies to reduce total kNN execution time as it provides a more efficient distance comparison structure. However, it sometimes leads to an increase of the memory requirements. This combination sustains the kNN efficiency whilst reducing the computational load during the classification process. The time complexity for the plain kNN is $O(n^2)$ (n is the size of the memorised training set). Therefore the execution time heavily depends on the training set size. The reduction of training set size allows kNN implementation within the embedded systems without suffering significant performance degradation.

The time complexity of this adapted CURE approach is $O(nJ - (J^2 + J)/2)$ (n_i is the original training set size, where $n_i > J$) and is dominated by the farthest neighbour algorithm. The time complexity for a small J is dominated by $O(n_i J)$ and the influence of the J number towards the execution time is linear. However, the relationship behaves almost as an inversed exponential pattern when the J number is increased substantially. The adapted clustering approach intends to capture the geometry of clusters using a small amount of representative points, J , and therefore the time complexity should be $O(n_i J)$. For huge raw data sets, the adapted CURE's execution time can be further reduced by random selection before applying the clustering approach. The random selection output size should be sufficient to maintain the clusters geometry without being compromised. This method, however, is not suitable for clusters that overlap.

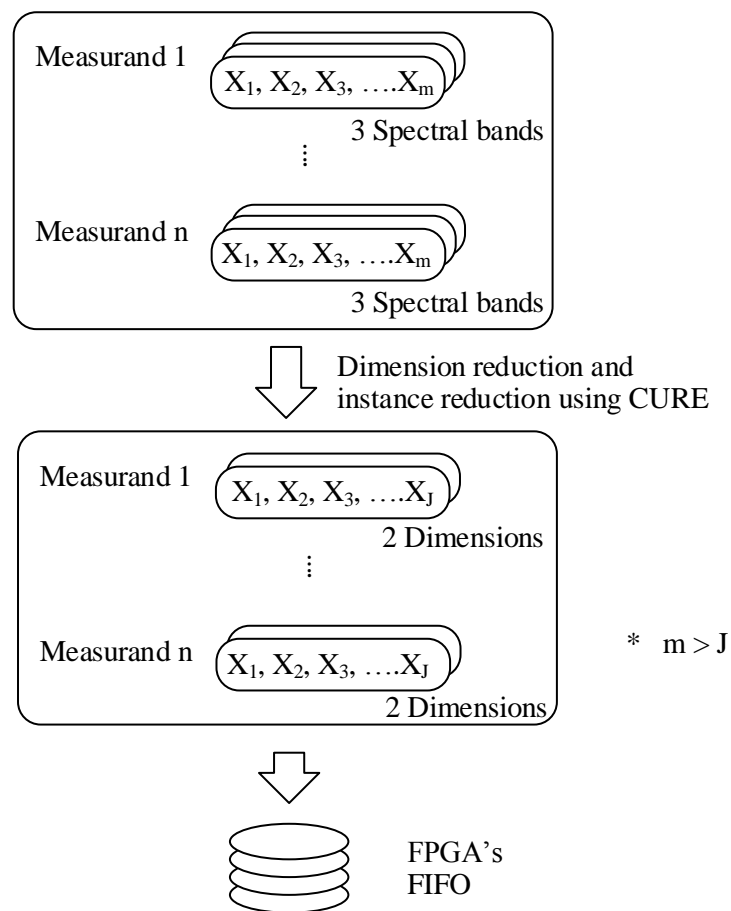


Figure 6-4 Representatives generation process schematic diagram.

For the next two subsections, the classification combination was tested with two sensors to demonstrate its feasibility to compress the training set into representatives. The found representatives were to be stored in the FPGA's FIFO (first in, first out data storage) as shown in Figure 6-4. kNN was applied using odd values of k as the acquired data distribution later (refer to Figure 6-5 and Figure 6-6) shows that space only exists between not more than two clusters. Space exists between clusters only bordered with two clusters. The k parameter was set as one for number of representative points (J) less than three and three otherwise.

6.3.1 Classification validation on Sensor A

The combination of the CURE and the kNN approach was evaluated using both sensors before implementation in the embedded system. The evaluation was performed using the holdout validation method [97]. The collected data were split into three sets: (i) training, (ii) validation, and (iii) test at ratios of 50: 25: 25 respectively. The splitting process is done randomly using *sample* function from Python's Panda module. Optimum representative points were founded using training set and validated using validation set. The test set was used for assessment of the generalization error. The amount of the collected data using Sensor A (50 nm) is tabulated in Table 6-2 and is split according to the defined ratios for the holdout validation.

Table 6-2 Number of collected samples for Sensor A.

Refractive index	Number of collected samples
1.3312	188
1.3378	276
1.3436	270
1.3514	243
1.3578	295
1.3633	260

Figure 6-5 depicts the representatives generated from the training sets for $J=2, 4, 6$ and centroid. The commonly used centroid approach is included in this section for comparison. The figure is a 2D plot which makes use of the blue LED channel as a reference by subtracting it from red and green as discussed in the previous section. Different sensing mechanisms might need to adopt different reduction schemes such as principal component analysis [98].

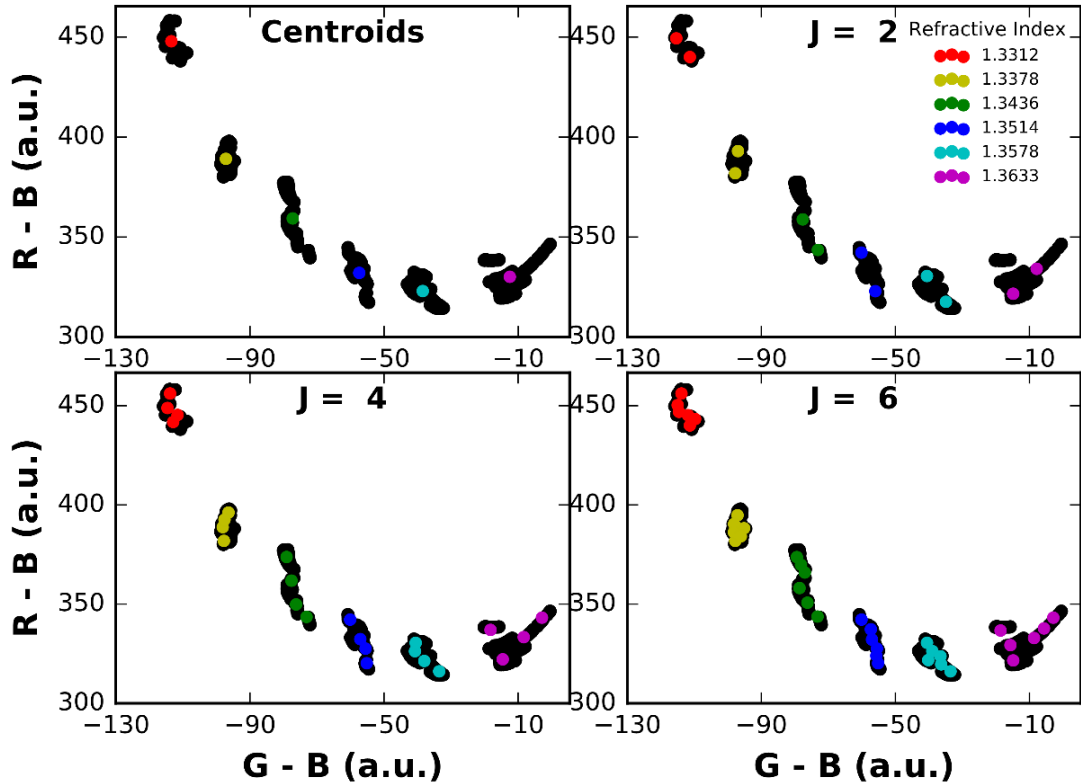


Figure 6-5 Representatives generation from Sensor A training set for scenarios included centroid, and J=2, 4, 6 from CURE approach. Training set is shown as black dot and representatives are shown as enlarged colour dots.

The centroid approach finds the mean for each cluster as a representative, while the CURE approach finds multiple representatives to fit the distribution geometry of the cluster. The centroid approach treats the cluster shapes indifferently. It is not suitable for clusters with arbitrary shape such as the elongated clusters produced by this sensor in the desired refractive index (RI) range. These representatives are used to classify the validation set. The optimum number of representatives is found with the prediction's success rate.

Table 6-3 Validation success rate for different representatives points (Sensor A, 50nm gold thickness).

CURE J-point	2	3	4	5	6	7	8	Centroids
RI	Validation success rate (%)							
1.3312	100	100	100	100	100	100	100	100
1.3378	100	100	100	100	100	100	100	100
1.3436	98.53	98.53	100	100	100	100	100	89.71
1.3514	100	100	100	100	100	100	100	100
1.3578	100	100	100	100	100	100	100	100
1.3633	100	100	100	100	100	100	100	100
Overall success rate (%)	99.74	99.74	100	100	100	100	100	98.17

The validation for different representative points as well as the centroids are tabulated in Table 6-3. The table shows the success rate for different clusters' label along with the overall success rate. The centroids and low number of representatives' performance are not ideal compared to higher number of representatives. The degradation of the performance is largely due to the RI=1.3436 cluster that has elongated shape. Such shape is difficult to be represented by a single point, centroid, or a small number of points that are not well scattered. The prediction's success rate is improved with more representative points. It reaches 100% overall success rate at J=4 scenario. Further increase in the number of representatives does not provides better improvement. Therefore, J=4 is chosen as the optimum number of representative points.

Table 6-4 Test success rate for sensor A at optimum case, J=4.

RI	Test success rate(%)
1.3312	100
1.3378	100
1.3436	100
1.3514	100
1.3578	100
1.3633	100

The optimum representative points are then used for the test set to assess the error generalisation. The test success rate is tabulated in Table 6-4 and shows that four representative points are sufficient for accurate prediction.

6.3.2 Classification validation on Sensor B

This classification combination is also examined using the second sensor to test its ability on different data distribution. The acquired data are also split into three sets for the holdout validation. The splitting process and ratios are similar to Sensor A.

Table 6-5 Number of collected samples for sensor B.

Refractive index	Number of collected samples
1.3325	597
1.34	493
1.348	592
1.355	592
1.362	595
1.3675	594

Figure 6-6 depicts the representatives generated for Sensor B. The dimension reduction in the previous section is also applied here. The data distribution is observed to have larger deviation in the y-axis compared to the x-axis and the data points are closer compared to the previous sensor.

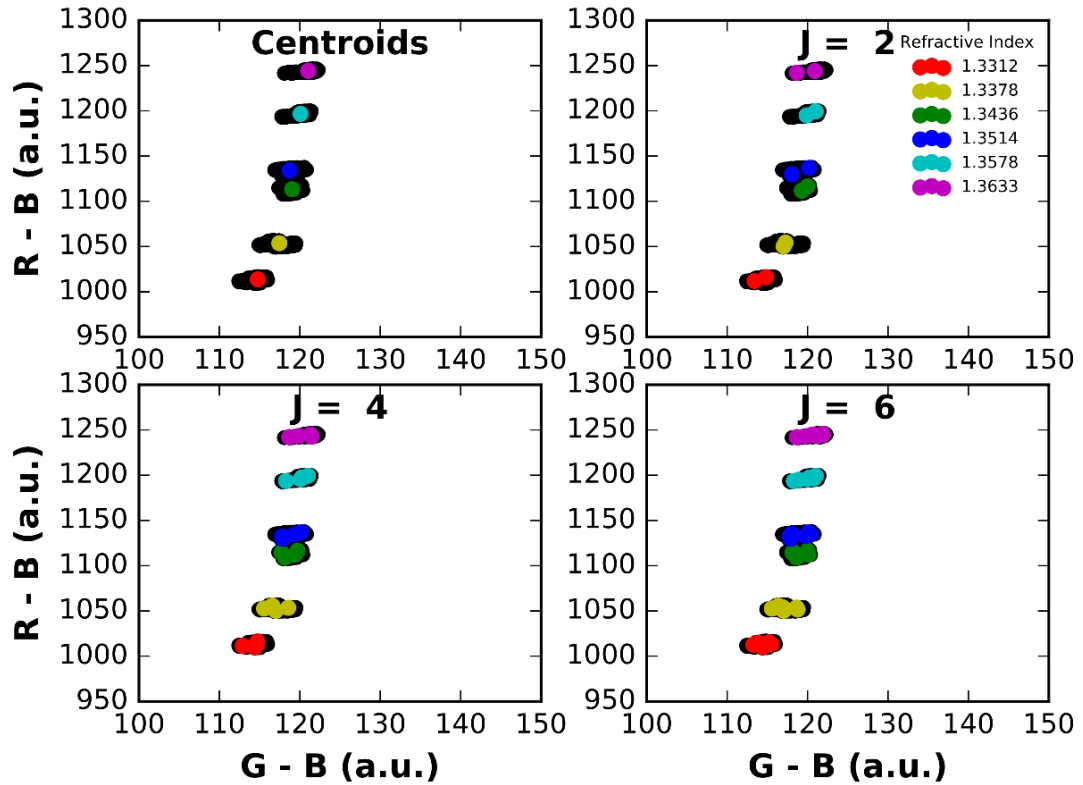


Figure 6-6 Representatives generation from sensor B training set for scenarios included centroid, and J=2, 4, 6 from CURE approach. Training set is shown as black dot and representatives are shown as enlarged colour dots.

Table 6-6 shows the validation success rate for different representative points and centroids. The success rate for all scenarios are 100% as the distribution of data is well defined and the cluster geometry is close to spherical shape. The good success rate for CURE and kNN combination shows that the classification combination works well for different distribution patterns.

Table 6-6 Validation success rate for different representatives points (Sensor B, 70nm gold thickness).

CURE J-point	2	3	4	5	6	7	8	Centroids
RI	Validation success rate(%)							
1.3312	100	100	100	100	100	100	100	100
1.3378	100	100	100	100	100	100	100	100
1.3436	100	100	100	100	100	100	100	100
1.3514	100	100	100	100	100	100	100	100
1.3578	100	100	100	100	100	100	100	100
1.3633	100	100	100	100	100	100	100	100
Overall success rate (%)	100	100	100	100	100	100	100	100

The same number of representative points as previous section is used for the test set as this number of points give both distribution 100% success rate during the validation process. Table 6-7 shows the test success rate for sensor B. The number of representative points provides good generalisation for both sensor and is used for later hardware module designs.

Table 6-7 Test success rate for sensor B at optimum case, J=4.

RI	Test success rate(%)
1.3312	100
1.3378	100
1.3436	100
1.3514	100
1.3578	100
1.3633	100

6.4 FPGA-implemented classification component design and performance

The main limitation of the application of kNN within an embedded system is mainly due to the high computational load from both the algorithm complexity and the large training set size [80, 99]. The combination of the dimensional reduction and the classification is used to reduce the complexity and compress the training set. In this work, a suitable portable platform, the FPGA, was selected to perform the kNN algorithm. The architecture of the classification process is shown in Figure 6-7. The three analog inputs are accumulated for 24 data points (each is 12 bit data) and are then reduced to two dimension data. The accumulation step is used only to avoid the resource-expensive division operator that is required in the averaging process. Distances

between the stored representatives and the acquired data are found and then ranked. The majority voting component is then triggered afterwards and produces the classification's prediction. The dimension reduction, distance calculation and ranking components for the kNN algorithm are designed to be operated in parallel and the operations are pipelined. The representatives generated from the previous section are multiplied by 24 and stored into the FIFO in a binary number form. The multiplication is to convert the average value into accumulation value over 24 data points.

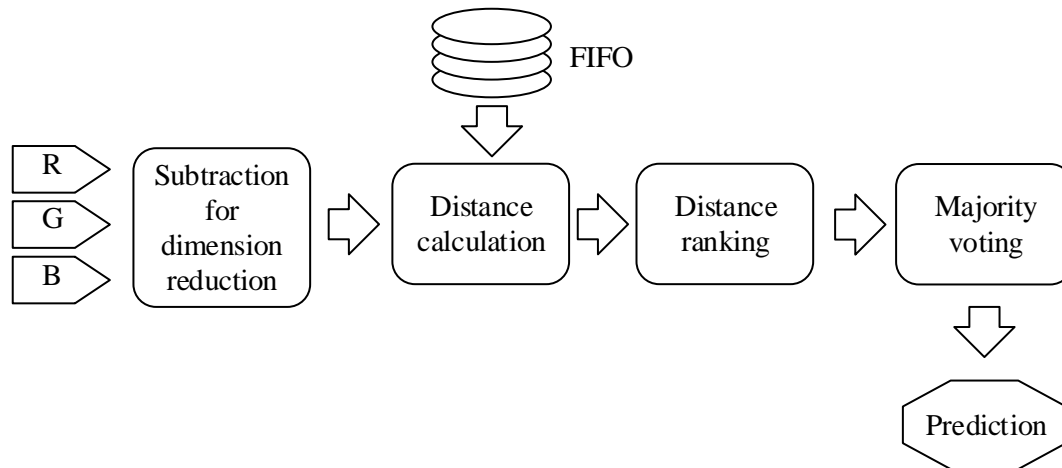


Figure 6-7 Architecture of the kNN classification in FPGA.

The next subsection focuses on the design, resource usage and performance of the kNN's individual components as well as the integration of the system as a whole. Those designed parts are then used to conduct real-time classification as detailed in the next subsection.

6.4.1 Development of the kNN algorithm components

This section presents the kNN module's components and their performance evaluation. Here, the classification relies on a hardware implementation that exploits the FPGA abilities and the custom blocks included in the DSP slices, DSP48E1, within the Artix-7 FPGA. The required operations are designed as components in VHDL and they are operated in a concurrent manner. These operations include distance calculation, distance sorting, and majority voting. The components as well as processing elements (PE) within it are synchronised using the rising edge of the clock to achieve a pipelined design.

Mixed architectures including the systolic array [80] were used in this work to realise the classification module design. Distance calculation and distance sorting components were constructed by a few different types of PE as shown in Figure 6-8. The majority voting component is constructed in a sequential operation. The PEs are described in a behaviour form and they depend on the software provided by the vendor for optimisation. The PEs are integrated

into one component through the combination of both dataflow and behavioural designs. The constructed components are then assembled as one kNN module using structural hierarchy.

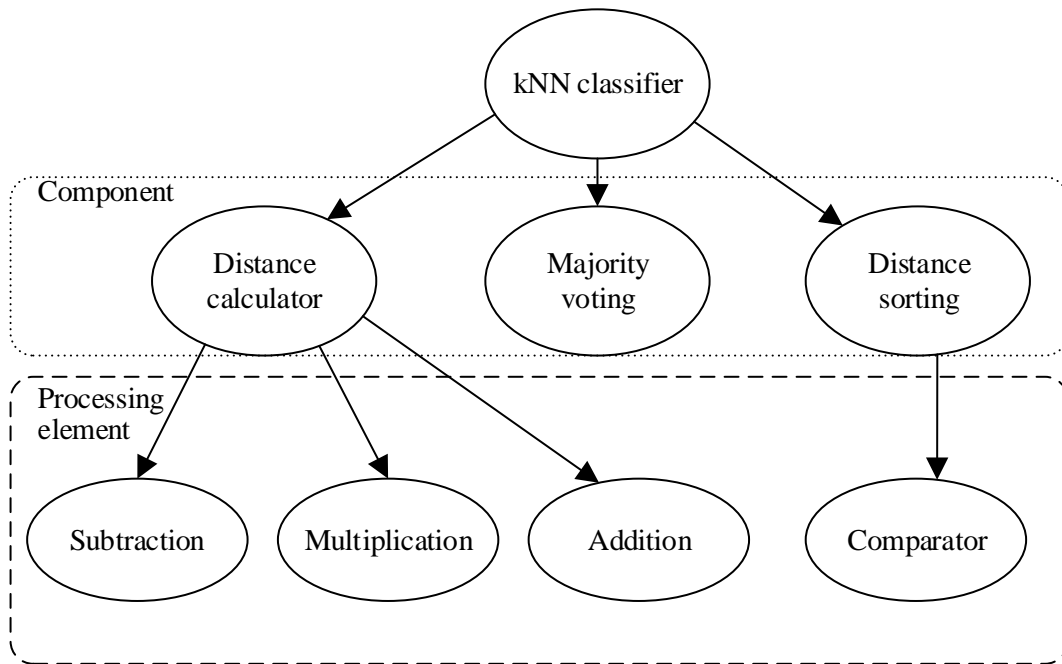


Figure 6-8 kNN module's hierarchy.

a) Distance calculator

Figure 6-9 (a) depicts the systolic array architecture of the distance calculator. The distance calculator finds similarities between the input data and the training set. In the current system, the distances are measured in Euclidean metric (equation(4.1)) by accumulating the square of the difference from every dimension. This component produced squared distance values (it is not distance value) for the next stage, distance sorting. The square root operation is omitted, as the squared distance values produced a correct ranking as the distance values. The RGB LEDs' intensities are reduced into two dimensions and therefore only two stages of accumulation are required as shown in Figure 6-9 (a). The PEs that are used to find the square of the difference for each dimension are connected together to form a subcomponent (shown in blue region). The formation of the subcomponents is for better modularity and reusability. This design can be extended by simply stacking more subcomponents to include more features or dimensions. To complete the subcomponent stacking, a constant 0 (hexagon shape in the figure) is placed at the top of the PEs array to complete the accumulation process. At the other end, at the bottom of the PEs array, the calculated distance between the training vector and the input is labelled using a header. The label is for the majority voting use. Figure 6-9 (b) shows the enable controller that manages the PEs' activation timing. The enable signal from the controller is passed to the PEs through a shift register.

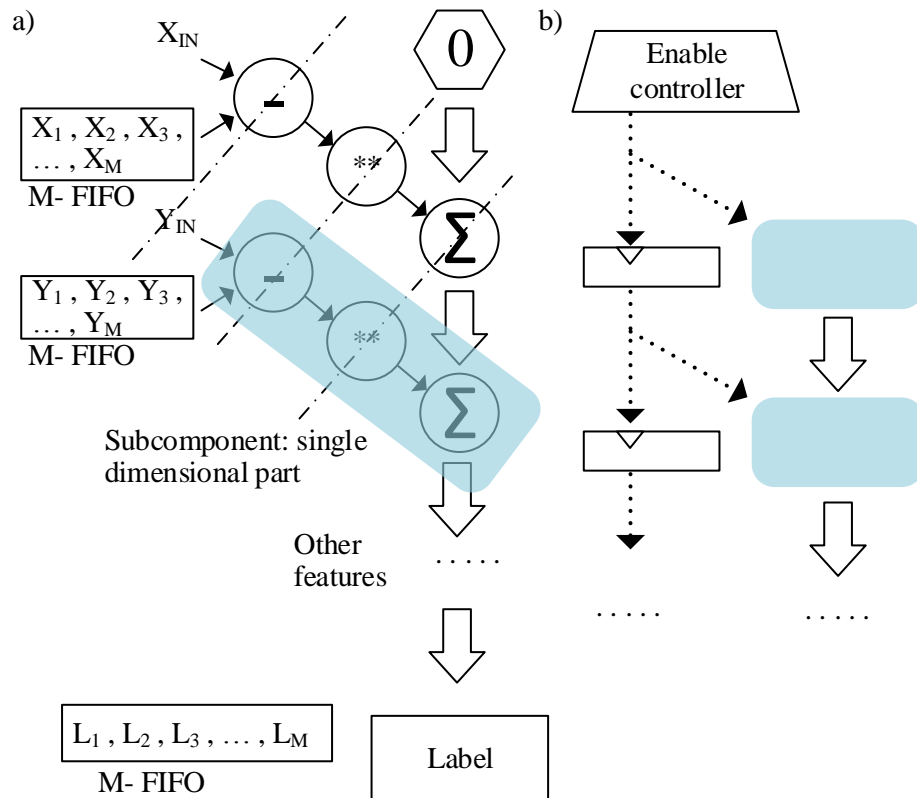


Figure 6-9 a) Systolic array architecture of distance calculator. b) Enable signal flow.

Figure 6-10 depicts the data flow process within the distance calculator. Stored data from the two FIFOs flow into the PEs that are activated according to timing or the enable signal. The operations are executed in a pipeline fashion and the dashed line is an analogue of wavefront for activation of the PEs. The small arrows and large arrows between the PEs represent the data transaction's direction within a subcomponent and between the subcomponents respectively. The dark grey nodes are inactivated PEs while activated PEs are shown in node with patterned background. The PEs in the same front have the same patterned background. The fronts propagate in diagonal manner from top left toward bottom right and never intersect to avoid conflicts [100]. This design maps a repetitive operation into a pipeline operation with the aid of registers between the PEs. The Euclidean distances between the representative points and the test vector can be calculated in $T = M + 1 + D$ clock cycles, where D is number of dimension. This results into a time complexity that equals to $O(M)$ as D is usually smaller than M . The component consumption in this design is linearly proportional to the data set dimension. This results into a space complexity that equals to $O(D)$.

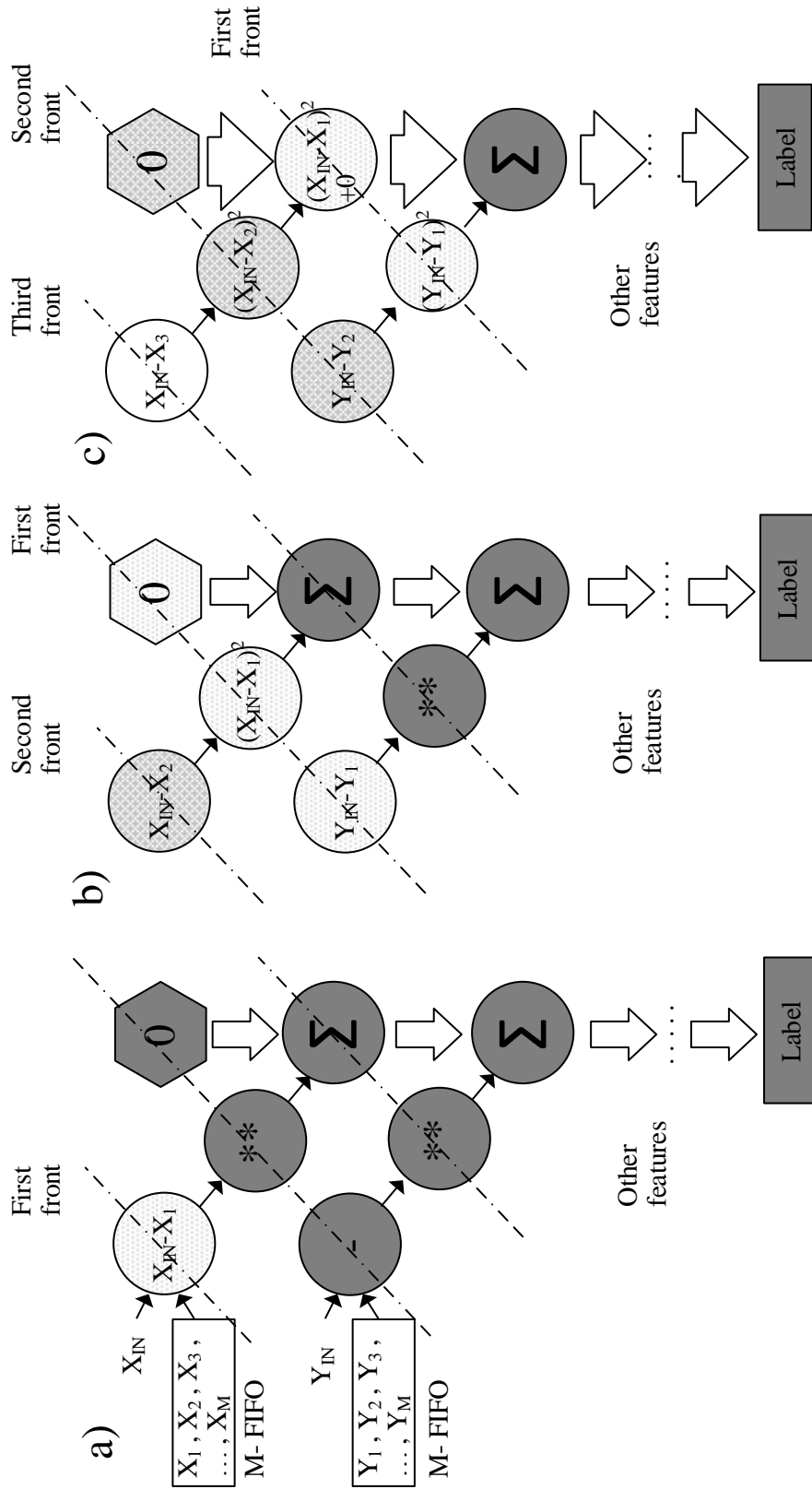


Figure 6-10 Data flow process in distance calculator.

b) Distance ranking

The k most similar (nearest in distance) representative points were found using “*winner take all*” (WTA) design as shown in Figure 6-11 [54]. This component is designed to keep k nearest neighbour in the registers at the end of the incoming data (all the stored data in the FIFO). The register ranking corresponds to the distance ranking (1st for the nearest and k^{th} for the k^{th} nearest) and initialised to the largest value the register can hold. This design utilises all the PEs work in parallel to sort the incoming data at every clock cycle. These PEs are activated together by the enable signal from the enable controller. The comparator compares the register’s data with new arriving data from the distance calculator. If a larger data was stored in the register, the data would be pushed downward to a lower rank register. Other than the first comparator, other comparator also required to compare with the data pushed from the upper rank register. The distance that is larger than all k register’s data is discarded. This component is able to rank the incoming data in 1 clock cycle and pipelined with the distance calculator. The space complexity is $O(k)$.

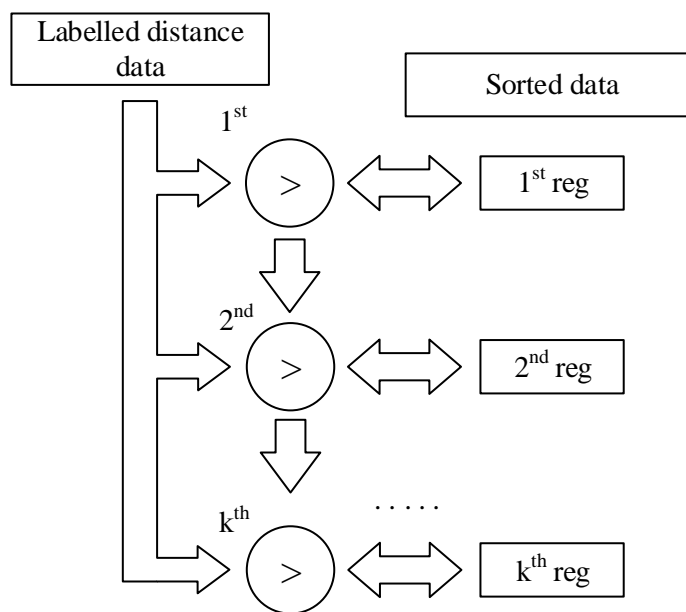


Figure 6-11 Architecture of distance sorting.

c) Majority finding

To classify an input test vector, the labels of the majority of the k NN are used to make the prediction. The majority finding components is triggered at the end of the distance ranking process and received the k labels from the distance ranking component. This component is implemented as a sequential operation and is broken into two parts as shown in Figure 6-12. The first part, score keeping, is to go through all the ranked data in loop to find what class it belongs to. The number of member for each class is recorded using a list, *score* list. The loop can be unrolled into a pipeline design if there is a large number of classes needs to go through. The

pipeline design can be implemented using a chain of combinations of 'equal to' Boolean operators and counter which is similar to the distance calculator. The second part finds the majority vote by looking for the largest value in the *score* list and its label (class) would be the classification prediction. The largest value finding can be parallelised using Bitonic sorting method [101]. This sequential process needs $T = k + L$ clock cycles, where L is the number of the labels. This component is implemented in sequential design as it can operate in parallel with the previous other two components that require longer time (total clock cycle = $M + 1 + D + 1$).

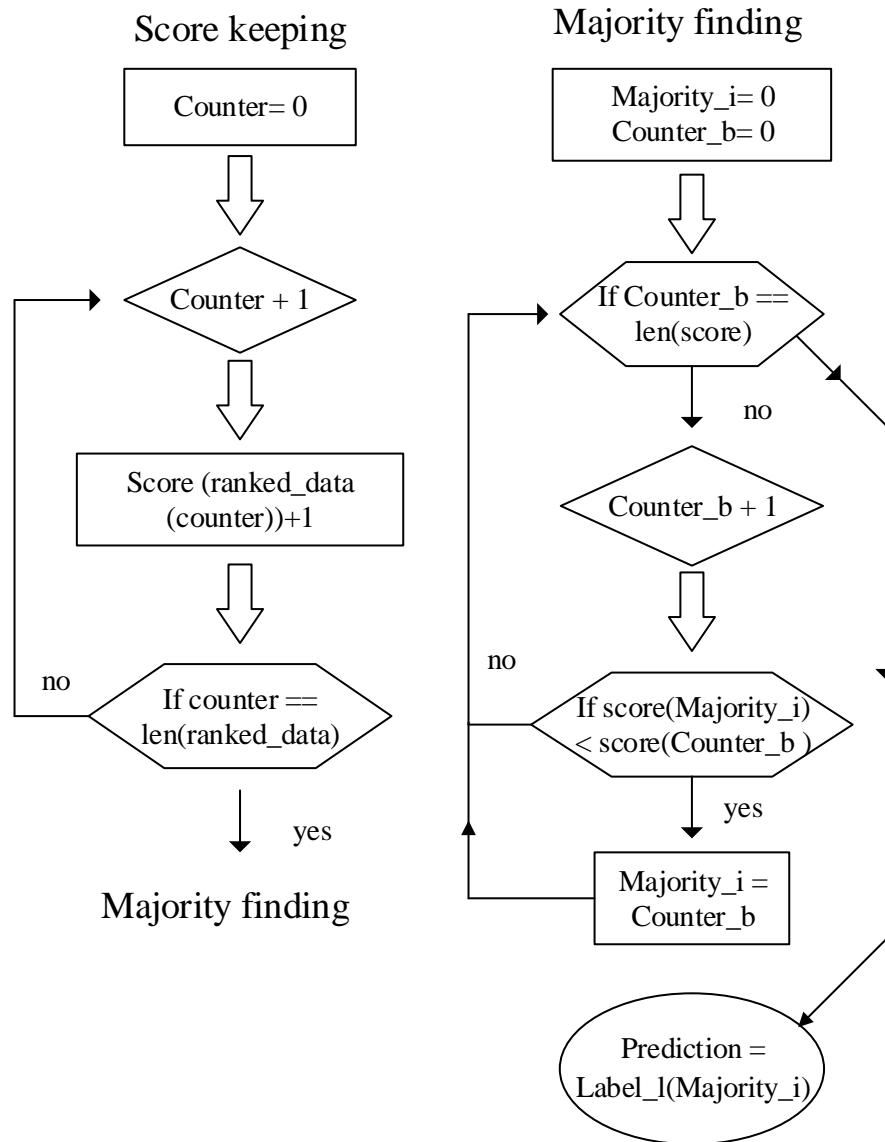


Figure 6-12 Majority voting process.

6.4.2 Synthesis results

The different components presented in the previous sections were designed to be tuneable parts/components and synthesised into circuitry for the targeted FPGA (Artix-7 35T) using Vivado. The resource utilisation for the synthesised part is presented in Table 6-8. The FPGA

have 20800 slice LUTs and 90 DSP slices. Each slice LUT is a 6-input look up tables (LUT) for logic use. It is coupled to a pair of flip-flops that functions as registers. The distance calculator and distance ranking are designed to be scalable according to user need. Therefore, the subcomponents of both are also reported as single entities.

Table 6-8 Implementation results for part and component.

Component	Required Resources		
	LUT as logic	LUT as Flip Flop	DSP slices
a) <i>Distance calculator (subcomponent: single dimensional part)</i>	120	33	3
b) <i>Distance calculator (whole)</i>	208	50	6
c) <i>Distance ranking (subcomponent)</i>	88	46	0
d) <i>Distance ranking (whole)</i>	221	95	0
e) <i>Voter</i>	114	55	0
f) <i>Label</i>	9	8	0

The distance calculator is comprised of two subcomponents as shown in Figure 6-9. From Table 6-8, the LUT utilisation of the distance calculator component does not scales exactly as the number of the subcomponents. The small discrepancy is due to the first stage have only a constant 0 before it and effectively only need to make a single connection to the component that follows it. This results in fewer connections, less logic circuitry and fewer registers required. This scaling mismatch also happens to distance ranking component as the last processing element discards the unwanted data instead passes to next processing element. Therefore less connection is required by the last processing element. On the other hand, the DSP utilisation scales well with the number of subcomponents as the number of the arithmetic operations remains the same. The available DSP slice, DSP48E1, in the current FPGA does not support square operation immediately after addition/subtraction operation. Therefore, multiple slices are required to achieve the series of operations. This, however, can be implemented using single DSP slice such as DSP48E2 [102], which is available in UltraScale FPGA family. Overall, the kNN module consumes 552 LUTs as logics, 208 LUTs as flip flops and 6 DSPs. These are 3.6% of the total available LUTs and 6.7% of the total available DSPs. The remaining FPGA's resources are still able to accommodate more functions.

6.4.3 Real-time classification

The system's real time performance was evaluated three times over three consecutive days. The measurements were conducted using sensor B and the test subjects were classified using

representatives set stored in the FIFOs. The results in terms of the system's classification success rate (over three times) are tabulated in Table 6-9.

Table 6-9 Confusion matrix for the real-time embedded kNN based refractive index classifier. Predicted classes are on the columns and actual classes are on the rows.

Solution	RI	Water	G1	G2	G3	G4	G5
Water	1.3325	100	0	0	0	0	0
G1	1.34	0	100	0	0	0	0
G2	1.348	0	0	100	0	0	0
G3	1.355	0	0	0	100	0	0
G4	1.362	0	0	0	0	100	0
G5	1.3675	0	0	0	0	0	100

The reliability of the sensor system is proven to be 100% success rate for all measurand. This may be due to the distribution of data - distance between clusters are large compared to the cluster shape itself. The large distance is due to the large refractive index's difference between the solutions, in the order of 10^{-3} . To put this sensor system into real application, the system can be modified into binary classification application such as identifying the presence of metal ion in water. The presence of metal ion (Mercury, Lead, and other) in the order of $100\mu\text{M}$ would produce refractive index changes in the order of 10^{-3} [103]. For applications that require finer difference between solutions, kNN can be applied as a regression method or include more types of solution within the training set. The kNN can be applied as regression method by using a division operator to average the k nearest neighbours' label instead of voting.

6.5 Summary

In this chapter, two different sensors' thicknesses were used to develop generalised understanding toward the SPR sensor system responses. The system responses were evaluated using both a spectrometer and the FPGA based sensor output. Through the knowledge about the system response, a suitable and general dimension reduction method was used to reduce the kNN computational complexity. To develop an embedded classifier module, the CURE (representative generation method) was applied to reduce the computational load. The classification combination was then evaluated to generate the optimised number of representatives for good generalised classification. The generated representatives were then stored in the classifier module for real time classification evaluation.

The kNN classifier was developed as a hierarchical design with three different components: (i) distance calculator, (ii) distance sorting, and (iii) majority voting. These components are

constructed as scalable designs. The component designs were discussed and evaluated in term of space and time complexity. The majority voting component was constructed in sequential manner as it could be conducted in parallel with other two components. However, for large number of class or k , it was readily realised as a pipelined and parallel operation. The developed components are then synthesised using Vivado and evaluated in term of resource utilisation. The performance of real-time classification were discussed at the end of the chapter.

Chapter 7 Conclusions

7.1 Thesis summary

This thesis has considered the development of a modular FPGA based OFS system. The use of the FPGA in an optical fibre sensor (OFS) system design was explored as the enabling technology. This led to the creation and evaluation of a portable sensor system design that could perform data processing functions in the field without the need for a personal computer (PC). For sensor data classification, the k-nearest neighbour (kNN) machine learning algorithm was adopted for use in this system to achieve embedded and real-time system operation. This work used a SPR sensor to demonstrate the feasibility of the system. The sensor was connected to a light source and photodetector using a plastic optical fibre (POF), allowing the detection of refractive index that has a resonance wavelength that falls within the visible wavelength range.

The main focus of this work was to design a low cost, portable and flexible OFS system using the FPGA. The system aimed to perform data processing with the FPGA including intensive computation tasks instead of requiring the use of an external computing unit. The OFS and the supporting elements of the system, the processing unit for data processing, and their implementation were discussed in detail. Here, a surface Plasmon resonance (the BU-SPR) optical sensor was used with time domain intensity modulation for three colour channels: red, green and blue. POF was used to send an optical signal to the sensor and collect the sensors' output light. The scheme implementation took advantage of the available off-shelf optoelectronic components to realise a low cost system. The electronic and optoelectronic components were grouped into individual systems based on their functionality. The subsystems that acted as the interface between OFS and the FPGA were simulated and characterised independently. The characteristics of the subsystems were taken into consideration during the design of the digital system to be implemented with the FPGA. The modules within the FPGA that were designed to support the system operation have been discussed. A GUI was designed for system control, data acquisition, data labelling and live data plotting on a PC. The system performance was investigated to ensure system repeatability which is crucial for later classification work. It was

discovered during the study that random noise existed in the signal. Hence, signal averaging was used to provide a more stable signal value.

Operation of the OFS system was conducted in a laboratory environment to classify solutions based on refractive index. The classification of six different refractive indices were performed using differential signals in order to reduce data dimensions. This also helped cancel noise that existed on both signals. kNN was chosen as the classification method. To implement the kNN algorithm in the embedded system with efficiency, the size of the data set was reduced and most of the classifications were conducted in pipelined fashion. CURE, a clustering method, was adopted as the instance reduction method and produced a data set that was an order of magnitude smaller in size than the original data set. The representatives were generated from the real measurement data and the optimum number of representatives are evaluated through the validation. The CURE algorithm is $O(n_t J)$ where n_t is the training set size and J is the representative set size. The embedded kNN module was designed using three components: (i) distance calculator, (ii) distance sorting, and (iii) majority voting. The time complexity for the distance calculator is $O(M)$, where M is the size of the stored training set, as the design is pipelined and the space complexity is $O(D)$. For distance sorting, the space complexity is $O(k)$, where k is the target number of nearest neighbours. The component was connected to last component in the pipeline and therefore the overall time complexity for the module did not change. The majority voting component was implemented in sequential fashion and conducted in parallel with other two components and therefore did not affect the module operation time. The kNN classification module was integrated with the sensor system to conduct a real-time prediction and the result could be used to qualitatively describe the system performance.

7.2 Contribution of the work

The major contributions of the work to OFS system are as follows:

1. Utilisation of time domain intensity modulation for three colour channels: red, green and blue, to interrogate the SPR sensor.
2. Utilisation of kNN in an OFS sensing application. This classification approach provides intuitive graphical results and therefore minimum training is needed for the personnel on field.
3. Adoption of CURE as a supervised method for instance reduction.
4. The sensor system design can be applied to most of the resonance based optical sensors that have broad response.
5. The system was been constructed from commercially available components, and utilises a simple design. This have a potential for wide adoption of the sensor system.

7.3 Future work

Potential research directions could be derived from this thesis and include:

1. To perform the instance reduction algorithm using FPGA by reuse of part of the classification module.

This allows the sensor system to be an independent system without relying on an external computing unit to generate the representatives. The CURE algorithm was required to perform a large number of distance calculations and hardware implementation by reusing the distance calculator component can greatly benefit the algorithm operation. The effect would be noticeable for data with large number of dimensions. To validate the representatives (holdout validation or other validation method), the user might need to conduct three different data collections or perform the dataset separation using a random number generator module as the dataset splitting have to be randomised.

2. To develop a quasi-distributed sensor with multiple OFS (as shown in Figure 7-1) by duplicate modules within the FPGA using the remaining available resources.

The plastic optical fibre can easily transmit optical signals over few tens of metres and this allows one system to monitor multiple points or multiple parameters at the same time. This implementation could reuse components within the FPGA and the light source. A small change is only required within the FPGA module design to sample and analyse the extra inputs. The optical fibres could be bundled together and coupled to the same LED using a new 3D printed coupler. However, this implementation would be required to include more photodetectors for signal sampling. The analogue output could be sampled by the same FPGA as it has multiple analogue inputs. This implementation may spawn one more future research directions as the aggregated data from multiple sensors might provide new information to the monitored environment. To provide such information, a different classification method might be required.

3. To incorporate power saving features such as clock gating for FPGA modules or to implement the system using flash based FPGA to lower the power consumption.

Power saving features such as clock gating could be implemented by adding logic to reduce dynamic power consumption by suppressing and minimising nonessential activity within the design, especially in sequential processes. The designer could implement clock gating using the solutions provided by Xilinx® or manually develop the clock gating. In addition, the FPGA's clock frequency

could be manipulated using clock management within the FPGA to slow down activity and reduce dynamic power consumption. These methods can extend the system operation time under battery power. However, the design requires extra considerations if there are different clock regions to avoid timing conflicts. Flash based FPGAs are well known to have low static power that could rival the microcontroller.

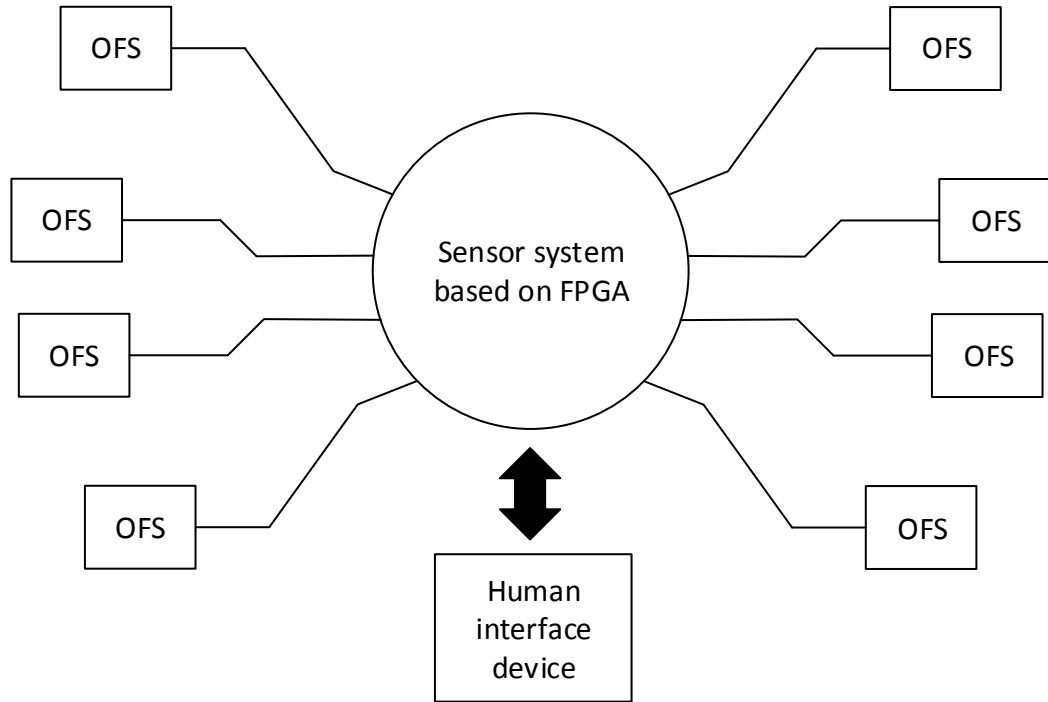


Figure 7-1 Proposed quasi-distributed sensor system based on FPGA.

References

1. Capitan-Vallvey, L.F. and A.J. Palma, *Recent developments in handheld and portable optosensing—A review*. *Analytica chimica acta*, 2011. **696**(1-2): p. 27-46.
2. O’Toole, M. and D. Diamond, *Absorbance based light emitting diode optical sensors and sensing devices*. *Sensors*, 2008. **8**(4): p. 2453-2479.
3. Wang, Y., et al., *Hardware embedded fiber sensor interrogation system using intensive digital signal processing*. *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, 2014. **13**(2): p. 139-155.
4. Fernandez-Jaramillo, A.A., et al., *FPGA-based chlorophyll fluorescence measurement system with arbitrary light stimulation waveform using direct digital synthesis*. *Measurement*, 2015. **75**: p. 12-22.
5. Jiang, W. and M. Gokhale. *Real-time classification of multimedia traffic using fpga*. in *Field Programmable Logic and Applications (FPL), 2010 International Conference on*. 2010. IEEE.
6. Cifuentes, A. and E. Marín, *Implementation of a field programmable gate array-based lock-in amplifier*. *Measurement*, 2015. **69**: p. 31-41.
7. Wang, J., H. Wang, and X. Liu, *A portable laser photoacoustic methane sensor based on FPGA*. *Sensors*, 2016. **16**(9): p. 1551.
8. Wöhrle, H., et al., *A Hybrid FPGA-Based System for EEG-and EMG-Based Online Movement Prediction*. *Sensors*, 2017. **17**(7): p. 1552.
9. Drimer, S. and M.G. Kuhn. *A protocol for secure remote updates of FPGA configurations*. in *International Workshop on Applied Reconfigurable Computing*. 2009. Springer.
10. Zhang, C., et al., *Optical fibre temperature and humidity sensor*. *Electronics Letters*, 2010. **46**(9): p. 643-644.
11. Saegusa, T. and T. Maruyama, *An FPGA implementation of real-time K-means clustering for color images*. *Journal of Real-Time Image Processing*, 2007. **2**(4): p. 309-318.
12. Markou, M. and S. Singh, *Novelty detection: a review—part 1: statistical approaches*. *Signal processing*, 2003. **83**(12): p. 2481-2497.
13. Alwis, L., T. Sun, and K. Grattan, *Optical fibre-based sensor technology for humidity and moisture measurement: Review of recent progress*. *Measurement*, 2013. **46**(10): p. 4052-4074.
14. Kuang, K.S.C., et al., *Plastic optical fibre sensors for structural health monitoring: A review of recent progress*. *Journal of sensors*, 2009. **2009**.
15. Roriz, P., et al., *Review of fiber-optic pressure sensors for biomedical and biomechanical applications*. *Journal of biomedical optics*, 2013. **18**(5): p. 050903.

16. Boozer, C., et al., *Looking towards label-free biomolecular interaction analysis in a high-throughput format: a review of new surface plasmon resonance technologies*. *Current opinion in biotechnology*, 2006. **17**(4): p. 400-405.
17. Wilson, D. and B. Ferguson. *Optimization of surface plasmon resonance for environmental monitoring*. in *Sensors, 2010 IEEE*. 2010. IEEE.
18. Homola, J., S.S. Yee, and G. Gauglitz, *Surface plasmon resonance sensors*. *Sensors and Actuators B: Chemical*, 1999. **54**(1-2): p. 3-15.
19. Okamoto, K., *Fundamentals of optical waveguides*. 2006: Academic press.
20. Udd, E. and W.B. Spillman Jr, *Fiber optic sensors: an introduction for engineers and scientists*. 2011: John Wiley & Sons.
21. Morey, W.W., G. Meltz, and W.H. Glenn. *Fiber optic Bragg grating sensors*. in *Fiber Optic and Laser Sensors VII*. 1990. International Society for Optics and Photonics.
22. Poeggel, S., et al., *Fiber-optic EFPI pressure sensors for in vivo urodynamic analysis*. *IEEE Sensors Journal*, 2014. **14**(7): p. 2335-2340.
23. Kasap, S.O. and R.K. Sinha, *Optoelectronics and photonics: principles and practices*. Vol. 340. 2001: Prentice Hall New Jersey.
24. Grattan, K.T. and B.T. Meggitt, *Optical fiber sensor technology*. Vol. 1. 1995: Springer.
25. Wang, W.-C., *Optical Detectors*. Seattle: Department of Mechanical Engineering-University of Washington, 2011.
26. Borecki, M., *Intelligent fiber optic sensor for estimating the concentration of a mixture-design and working principle*. *Sensors*, 2007. **7**(3): p. 384-399.
27. Spillman Jr, W.B., *Fiber optic sensors: an introduction for engineers and scientists*. 2010: John Wiley & Sons.
28. Ghetia, S., R. Gajjar, and P. Trivedi, *Classification of fiber optical sensors*. *Int. J. Electron. Commun. Comput. Technol*, 2013. **3**: p. 442-445.
29. Gupta, B.D., *Surface plasmon resonance based fiber optic sensors*, in *Reviews in Plasmonics 2010*. 2012, Springer. p. 105-137.
30. Johansen, K., et al., *Surface plasmon resonance: instrumental resolution using photo diode arrays*. *Measurement Science and Technology*, 2000. **11**(11): p. 1630.
31. Zeng, Y., et al., *Wavelength-scanning SPR imaging sensors based on an acousto-optic tunable filter and a white light laser*. *Sensors*, 2017. **17**(1): p. 90.
32. Wong, C.L. and M. Olivo, *Surface plasmon resonance imaging sensors: a review*. *Plasmonics*, 2014. **9**(4): p. 809-824.
33. Pereira, E.G., et al. *Implementation of a FPGA-based data acquisition and processing system for image sensors employed in SPR biosensing*. in *Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International*. 2014. IEEE.
34. Klantsataya, E., et al., *Plasmonic fiber optic refractometric sensors: From conventional architectures to recent design trends*. *Sensors*, 2016. **17**(1): p. 12.
35. Di, H., Y. Xin, and S. Sun, *Electric current measurement using fiber-optic curvature sensor*. *Optics and Lasers in Engineering*, 2016. **77**: p. 26-30.
36. Krehel, M., et al., *An optical fibre-based sensor for respiratory monitoring*. *Sensors*, 2014. **14**(7): p. 13088-13101.

37. Sciacca, B., et al., *Radiative-surface plasmon resonance for the detection of apolipoprotein E in medical diagnostics applications*. *Nanomedicine: Nanotechnology, Biology and Medicine*, 2013. **9**(4): p. 550-557.
38. Yanase, Y., et al., *Development of an optical fiber SPR sensor for living cell activation*. *Biosensors and Bioelectronics*, 2010. **25**(5): p. 1244-1247.
39. Laskar, S. and S. Bordoloi, *Monitoring of moisture in transformer oil using optical fiber as sensor*. *Journal of photonics*, 2013. **2013**.
40. Van Hoe, B., et al., *Ultra small integrated optical fiber sensing system*. *Sensors*, 2012. **12**(9): p. 12052-12069.
41. Liu, Y., et al., *Surface plasmon resonance biosensor based on smart phone platforms*. *Scientific reports*, 2015. **5**: p. 12864.
42. Bremer, K. and B. Roth, *Fibre optic surface plasmon resonance sensor system designed for smartphones*. *Optics express*, 2015. **23**(13): p. 17179-17184.
43. Attaran, N., et al., *Embedded Low-Power Processor for Personalized Stress Detection*. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2018.
44. Farooq, U., Z. Marrakchi, and H. Mehrez, *FPGA architectures: An overview*, in *Tree-based heterogeneous FPGA architectures*. 2012, Springer. p. 7-48.
45. Rodríguez, A., et al., *FPGA-Based High-Performance Embedded Systems for Adaptive Edge Computing in Cyber-Physical Systems: The ARTICo3 Framework*. *Sensors*, 2018. **18**(6): p. 1877.
46. Serrano, J., *Introduction to FPGA design*, in *CERN Accelerator School Course on Digital Signal Processing*. 2008, CERN
47. Xilinx, *Vivado Design Suite User Guide - Logic Simulation*. 2018. p. 252.
48. Xilinx, *HDL SYNTHESIS FOR FPGAs™ DESIGN GUIDE*. 1995, Xilinx.
49. Woods, R., et al., *FPGA-based implementation of signal processing systems*. 2008: John Wiley & Sons.
50. Woods, L., *FPGA-enhanced data processing systems*. 2014, ETH Zurich.
51. Viseh, S., M. Ghovanloo, and T. Mohsenin, *Towards an ultra low power on-board processor for tongue drive system*. *Circuits and Systems II: IEEE Transactions on*, accepted, 2015. **62**(2): p. 174-178.
52. Hervás, M., et al., *An FPGA-Based WASN for Remote Real-Time Monitoring of Endangered Species: A Case Study on the Birdsong Recognition of *Botaurus stellaris**. *Sensors*, 2017. **17**(6): p. 1331.
53. Martinez-Figueroa, G.D.J., et al., *FPGA-Based Smart Sensor for Detection and Classification of Power Quality Disturbances Using Higher Order Statistics*. *IEEE Access*, 2017. **5**: p. 14259-14274.
54. Shi, M., et al., *A committee machine gas identification system based on dynamically reconfigurable FPGA*. *IEEE Sensors Journal*, 2008. **8**(4): p. 403-414.
55. Zhang, W., et al., *A Fiber Bragg Grating Interrogation System with Self-Adaption Threshold Peak Detection Algorithm*. *Sensors*, 2018. **18**(4): p. 1140.
56. Yao, Y., et al., *Performance optimization design for a high-speed weak FBG interrogation system based on DFB laser*. *Sensors*, 2017. **17**(7): p. 1472.
57. Kaddachi, M.L., et al. *FPGA-based image compression for low-power Wireless Camera Sensor Networks*. in *Next Generation Networks and Services (NGNS), 2011 3rd International Conference on*. 2011. IEEE.

58. Murphy, C., *Driving the Xilinx Analog-to-Digital Converter*. 2012, Xilinx.
59. Mohsin, M.A., *AN FPGA-BASED HARDWARE ACCELERATOR FOR K-NEAREST NEIGHBOR CLASSIFICATION FOR MACHINE LEARNING*. 2017, University of Colorado Colorado Springs. Kraemer Family Library.
60. Duda, R.O., P.E. Hart, and D.G. Stork, *Pattern classification*. 2012: John Wiley & Sons.
61. Wilson, D.R. and T.R. Martinez, *Reduction techniques for instance-based learning algorithms*. *Machine learning*, 2000. **38**(3): p. 257-286.
62. Tang, B. and H. He, *ENN: Extended nearest neighbor method for pattern recognition [research frontier]*. *IEEE Computational intelligence magazine*, 2015. **10**(3): p. 52-60.
63. Hassan, D., U. Aickelin, and C. Wagner. *Comparison of distance metrics for hierarchical data in medical databases*. in *Neural Networks (IJCNN), 2014 International Joint Conference on*. 2014. IEEE.
64. Medjahed, S.A., T.A. Saadi, and A. Benyettou, *Breast cancer diagnosis by using k-nearest neighbor with different distances and classification rules*. *International Journal of Computer Applications*, 2013. **62**(1).
65. Dudani, S.A., *The distance-weighted k-nearest-neighbor rule*. *IEEE Transactions on Systems, Man, and Cybernetics*, 1976(4): p. 325-327.
66. Marchiori, E. *Class dependent feature weighting and k-nearest neighbor classification*. in *IAPR International Conference on Pattern Recognition in Bioinformatics*. 2013. Springer.
67. Tahir, M.A. and A. Bouridane. *An FPGA based coprocessor for cancer classification using nearest neighbour classifier*. in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. 2006. IEEE.
68. Beretta, L. and A. Santaniello, *Nearest neighbor imputation algorithms: a critical evaluation*. *BMC medical informatics and decision making*, 2016. **16**(3): p. 74.
69. Kumar, A. and G. Gill, *Computer vision based model for fruit sorting using K-nearest neighbour classifier*. *Int. J. Electr. Electron. Eng.*, 2015. **2**: p. 1694-2426.
70. Manne, S., S.K. Kotha, and S.S. Fratima, *A Query based Text Categorization using K-nearest neighbor Approach*. *International Journal of Computer Applications*, 2011. **32**(7): p. 16-21.
71. Shlens, J., *A tutorial on principal component analysis*. arXiv preprint arXiv:1404.1100, 2014.
72. Hwang, W.-J. and K.-W. Wen, *Fast kNN classification algorithm based on partial distance search*. *Electronics letters*, 1998. **34**(21): p. 2062-2063.
73. Merry, B., J.E. Gain, and P. Marais. *Accelerating kd-tree Searches for all k-nearest Neighbours*. in *Eurographics (Short Papers)*. 2013.
74. Triguero, I., et al., *A taxonomy and experimental study on prototype generation for nearest neighbor classification*. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2012. **42**(1): p. 86-100.
75. Garcia, S., et al., *Prototype selection for nearest neighbor classification: Taxonomy and empirical study*. *IEEE transactions on pattern analysis and machine intelligence*, 2012. **34**(3): p. 417-435.
76. Kakde, H.M., *Range searching using KD tree*. Florida State University, 2005.
77. Triguero, I., et al., *Prototype generation for nearest neighbor classification: Survey of methods*, in *Technical Report, Department of Computer Science and Artificial Intelligence*. 2011, University of Granada, Spain, Tech. Rep.

78. Tou, J.Y., et al. *Evaluation of Speed and accuracy for comparison of texture classification implementation on embedded platform*. Citeseer.
79. Chen, T.-C., et al. *1.4 μ W/channel 16-channel EEG/ECoG processor for smart brain sensor SoC*. in *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*. 2010. IEEE.
80. Stamoulias, I. and E.S. Manolakos, *Parallel architectures for the kNN classifier--design of soft IP cores and FPGA implementations*. ACM Transactions on Embedded Computing Systems (TECS), 2013. **13**(2): p. 22.
81. An, F., T. Koide, and H.J. Mattausch, *A K-means-based multi-prototype high-speed learning system with FPGA-implemented coprocessor for 1-NN searching*. IEICE TRANSACTIONS on Information and Systems, 2012. **95**(9): p. 2327-2338.
82. Kung, H.-T., *Why systolic architectures?* IEEE computer, 1982. **15**(1): p. 37-46.
83. Gonzalez, G.S., *Flexible systolic architecture for image processing at video rate*. 2007.
84. Vucha, M. and A. Rajawat, *Design and FPGA implementation of systolic array architecture for matrix multiplication*. International Journal of Computer Applications, 2011. **26**(3): p. 18-22.
85. Somarapalli, M., et al., *Demonstration of low-cost and compact SPR optical transducer through edge light coupling*. Micro & Nano Letters, 2017. **12**(9): p. 643-646.
86. Hobbs, P.C., *Photodiode front ends: The real story*. Optics and Photonics News, 2001. **12**(4): p. 44-47.
87. Jung, W.G., *Op Amp applications handbook*. 2005: Newnes.
88. Mancini, R., *Op amps for everyone: design reference*. 2003: Newnes.
89. Bhat, A., *Stabilize your transimpedance amplifier*, in *Application Note 5129*. 2012, Maxim Integrated.
90. Karki, J., *Effect of Parasitic Capacitance in Op Amp Circuits*, in *Application Report SLOA013A*. 2000, Texas Instrument.
91. emiAnalyst. *Coplanar Capacitance Capacitance Calculators*. [cited 2018 26/04]; Available from: <https://www.emissoftware.com/calculator/coplanar-capacitance/>.
92. Mahmoud, M.S. and A.A. Mohamad, *A study of efficient power consumption wireless communication techniques/modules for internet of things (IoT) applications*. 2016.
93. Gu, Y., et al. *Spectral and luminous efficacy change of high-power LEDs under different dimming methods*. in *Sixth International Conference on Solid State Lighting*. 2006. International Society for Optics and Photonics.
94. Manninen, P. and P. Orreveteläinen, *On spectral and thermal behaviors of AlGaInP light-emitting diodes under pulse-width modulation*. Applied Physics Letters, 2007. **91**(18): p. 181121.
95. Chhajed, S., et al. *Junction temperature in light-emitting diodes assessed by different methods*. in *Light-Emitting Diodes: Research, Manufacturing, and Applications IX*. 2005. International Society for Optics and Photonics.
96. Guha, S., R. Rastogi, and K. Shim. *CURE: an efficient clustering algorithm for large databases*. in *ACM Sigmod Record*. 1998. ACM.
97. Chan, H.-P., B. Sahiner, and L. Hadjiiski. *Sample size and validation issues on the development of CAD systems*. in *International Congress Series*. 2004. Elsevier.
98. Bedoui, S., et al. *Electronic nose system and principal component analysis technique for gases identification*. in *Systems, Signals & Devices (SSD), 2013 10th International Multi-Conference on*. 2013. IEEE.

99. Chen, S. *K-Nearest Neighbor Algorithm Optimization in Text Categorization*. in *IOP Conference Series: Earth and Environmental Science*. 2018. IOP Publishing.
100. Kung, S., *VLSI array processors*. IEEE ASSP Magazine, 1985. **2**(3): p. 4-22.
101. Chen, R., S. Siritiyal, and V. Prasanna. *Energy and memory efficient mapping of bitonic sorting on FPGA*. in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 2015. ACM.
102. Xilinx. *UG579 UltraScale Architecture DSP Slice*. 2018 June 4, 2018 [cited 2018 6 September]; UG579 (v1.7) June 4, 2018:[Available from: https://www.xilinx.com/support/documentation/user_guides/ug579-ultrascale-dsp.pdf].
103. Chah, S., J. Yi, and R.N. Zare, *Surface plasmon resonance analysis of aqueous mercuric ions*. *Sensors and Actuators B: Chemical*, 2004. **99**(2-3): p. 216-222.
104. Serrano, J., *Introduction to FPGA design*. 2008.

Appendix A Publication

Book Chapter

1. Ong, Y. S., Grout, I., Lewis, E., & Mohammed, W. Plastic fiber optic sensor system design using the field programmable gate array, "Optical Fiber", ISBN 978-953-51-5672-7, InTechOpen

Conference publications

1. Grout, I., Ong, Y. S., Lewis, E., & Mohammed, W. (2017, June). Programmable logic based current control of light emitting diodes using sigma-delta modulation. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2017 14th International Conference on* (pp. 91-94). IEEE. (published)
2. Ong, Y. S., Grout, I., Lewis, E., & Mohammed, W. (2017, June). Optical fiber sensor system design utilizing the field programmable gate array. In *Electrical Engineering/Electronics Computer, Telecommunications and Information Technology (ECTI-CON), 2017 14th International Conference on* (pp. 95-98). IEEE. (published)
3. Ong, Y. S., Grout, I., Lewis, E., & Mohammed, W. (2017, September). Surface Plasmon Resonance sensor read out system based on the Field Programmable Gate Array. In *26th International Conference on Plastic Optical Fibres* (Vol. 26, p. 43). Instituto de principleTelecomunicacoes, Aveiro.
4. Ong, Y. S., Grout, I., Lewis, E., & Mohammed, W. (2018, July). Results Classification in an RGB LED Based Optical Fiber Sensor System using Python. In *Electrical Engineering/Electronics Computer, Telecommunications and Information Technology (ECTI-CON), 2018 15th International Conference*. IEEE. (accepted)

Journal publications

1. Ong, Y. S., Grout, I., Lewis, E., & Mohammed, W. (2018). Utilization of Data Classification in the Realization of a Surface Plasmon Resonance Readout System Using an FPGA Controlled RGB LED Light Source, in *IEEE Sensors Journal*, vol. 18, no. 20, pp. 8517-8524, 15 Oct.15, 2018.

Chapter title	Plastic fiber optic sensor system design using the field programmable gate array
Publisher	InTechOpen
Open Access	www.intechopen.com
Books homepage	https://www.intechopen.com/books/selected-topics-on-optical-fiber-technologies-and-applications
Print ISBN	978-953-51-3813-6
Online ISBN	978-953-51-3814-3
On-line access	https://www.intechopen.com/books/selected-topics-on-optical-fiber-technologies-and-applications/plastic-optical-fibre-sensor-system-design-using-the-field-programmable-gate-array
DOI	10.5772/intechopen.68456
Publication Date	February 14th 2018
Authors	Yong Sheng Ong, Ian Grout, Elfed Lewis and Waleed Mohammed

Abstract

Extrinsic optical fibre sensor (OFS) systems use a fibre optic cable as the medium for signal propagation between the sensor and the sensor electronics using light rather than electrical signals. A range of different optical fibre sensors have been developed and electronic hardware system designs interfacing the sensor with external electronic systems devised. In this chapter, the use of the field programmable gate array (FPGA) is considered to implement the circuit functions that are required within a portable optical fibre sensor system that uses a light emitting diode (LED) as the light source, a photodiode as the light receiver and the FPGA to implement the system control, digital signal processing (DSP) and communications operations. The capabilities of the FPGA will be investigated and a case study sensor design introduced and elaborated. The OFS system will be based on the FPGA and will provide wireless communications to an external supervisory system. The chapter will commence with an overview of OFS systems and the typical architecture of the system. Then the FPGA will be introduced and discussed as a hardware alternative to a software programmed processor that is currently widely used. A case study will then be presented with a discussion into design considerations

Paper title	Programmable logic based current control of light emitting diodes using sigma-delta modulation
Conference	2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)
Location	Phuket, Thailand
Date	27-30 June 2017
Publisher	IEEE
DOI	https://doi.org/10.1109/ECTICon.2017.8096180
Publication Date	07 November 2017
Authors	Ian Grout, Yong Sheng Ong, Elfed Lewis and Waleed Mohammed

Abstract

In this paper, the design and operation of a sigma-delta modulation based signal generator for the creation of signal waveforms to be used in current mode control of light emitting diodes (LEDs) is presented. The LED is to be used as the light source within an optical fiber sensor (OFS) system and is to be driven with known current waveforms for use within the system. In this paper, the use of pulse density modulation (PDM) is considered and waveforms are generated using a MATLAB/Simulink model of the analog waveform to encode along with a model of the sigma-delta modulator used. The output from the MATLAB[®]/Simulink model is a sigma-delta encoded bit stream along with a VHDL design description that implements the signal in digital memory that can be synthesised into logic to target a suitable programmable logic device (PLD).

Paper title	Optical fiber sensor system design utilizing the field programmable gate array
Conference	2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)
Location	Phuket, Thailand
Date	27-30 June 2017
Publisher	IEEE
DOI	https://doi.org/10.1109/ECTICon.2017.8096181
Publication Date	07 November 2017
Authors	Yong Sheng Ong, Ian Grout, Elfed Lewis and Waleed Mohammed

Abstract

This paper presents the design of a system that is aimed to provide a flexible, portable and low cost solution for optical fiber based sensor systems. The field programmable gate array (FPGA) provides the digital logic to implement the system and ability to reconfigure the system operation. It aims to support different optical fiber sensing requirements by the ability to reconfigure the digital circuitry used. It is therefore a hardware configured alternative to a software programmed processor based approach. The work discussed in this paper focuses on the architecture of the FPGA based system with additional circuitry to implement the light source using a light emitting diode (LED), sensor signal sampling using a photodiode and the digital functions implemented using the Xilinx® Artix-7 FPGA. The system also includes serial communications to an external computer that allows the system to be used as part of a larger sensor network. In this paper, system control and sensor data visualization on a personal computer (PC) is undertaken using the Python open source programming language.

Paper title	Surface Plasmon Resonance sensor read out system based on the Field Programmable Gate Array
Conference	26th International Conference on Plastic Optical Fibres
Location	Aveiro, Portugal
Date	13- 15 September 2017
Publisher	Instituto de Telecomunicacoes, Aveiro
DOI	-
Publication Date	13 Sep 2017
Authors	Yong Sheng Ong, Ian Grout, Elfed Lewis and Waleed Mohammed

Abstract

In this paper, the design and operation of plastic optical fibre (POF) based surface plasmon resonance (SPR) sensor read out system is presented. The design is based on the use of the field programmable gate array (FPGA) which is aimed to provide a flexible, portable and low cost platform for this type of sensor. The light source is an RGB-red, green, blue-light emitting diode (LED) and the detector is a silicon photodiode for intensity interrogation. The LED is used with a single photodiode to provide a pseudo-wide spectrum and self-referencing capability. The system is interfaced to a computer running a Python open-source language script using Zigbee wireless connection. The script provides FPGA control, sensor data capture and visualisation, and database interfacing.

Paper title	Utilization of Data Classification in the Realization of a Surface Plasmon Resonance Readout System Using an FPGA Controlled RGB LED Light Source
Publisher	IEEE Sensors Journal
Print ISBN	1530-437X
Electronic ISBN	1558-1748
On-line access	https://ieeexplore.ieee.org/document/8444426
DOI	https://doi.org/10.1109/JSEN.2018.2866495
Publication Date	22 August 2018
Authors	Yong Sheng Ong, Ian Grout, Elfed Lewis and Waleed Mohammed

Abstract

This paper presents the realization of a surface plasmon resonance (SPR) sensor readout system using a tricolor red, green, and blue light emitting diode light source. Time domain intensity modulation of each color channel is applied to interrogate three bands of interest in the SPR spectrum using a single photodiode detector. A low computing resource classification approach is used through the combination of k-nearest neighbor (kNN) and adapted clustering using representative. An optimised number of representatives is chosen in the validation process to reduce the required amount of data for the kNN classification. This scheme was used to classify the concentrations of different glucose solutions. The sensor readout system hardware is based on the use of a field programmable gate array and the glucose solution classification is developed and undertaken on a personal computer using the Python open source programming language.

Appendix B Software and hardware used

Table B-1 Software.

Xilinx® Vivado	Version	2017.4
	Provider	Xilinx®
	Website	https://www.xilinx.com/products/design-tools/vivado.html
Python	Version	3.4
	Provider	The Python Software Foundation
	Website	https://www.python.org/
MySQL	Version	5.7
	Provider	Oracle Corporation
	Website	https://www.mysql.com
Target 3001 V18 pcb-pool	Version	18.9.0.51
	Provider	Ing.-Büro Friedrich, Eichenzell
	Website	https://ibfriedrich.com/en/index.html
LTspice	Version	4.23k
	Provider	Linear Technology Corporation
	Website	https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html

Table B-2 Hardware.

Artix-7 FPGA Development Board	Model	Arty board
	Website	https://www.xilinx.com/products/boards-and-kits/arty.html
	User Manual	https://reference.digilentinc.com/reference/programmable-logic/arty/start

Artix-7 FPGA	Model	XC7A35T
	Website	https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html
	User Manual	https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html#documentation
XBee 1 mW Trace Antenna	Model	Series 1 (802.15.4)
	Website	https://www.digi.com/support/productdetail?pid=3257
	User Manual	https://www.digi.com/resources/documentation/Digidocs/90000982/Default.htm

Appendix C Xilinx® Vivado

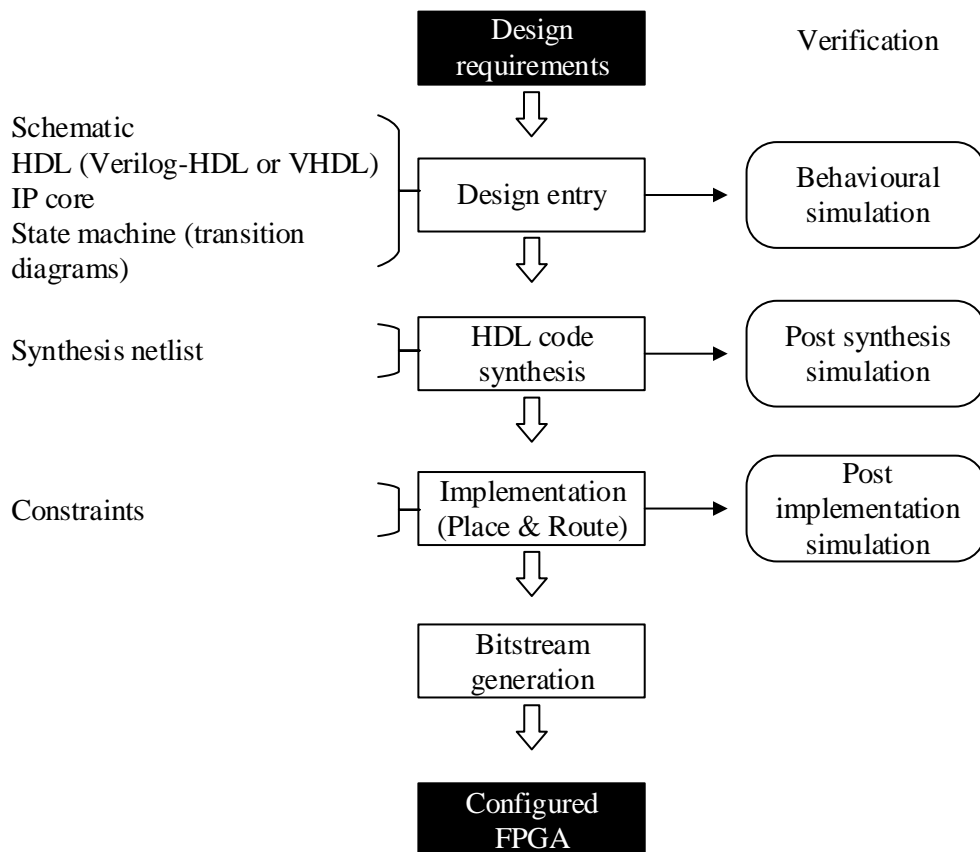


Figure C-1 Design flow [104].

Appendix D Digilent[®] Arty board

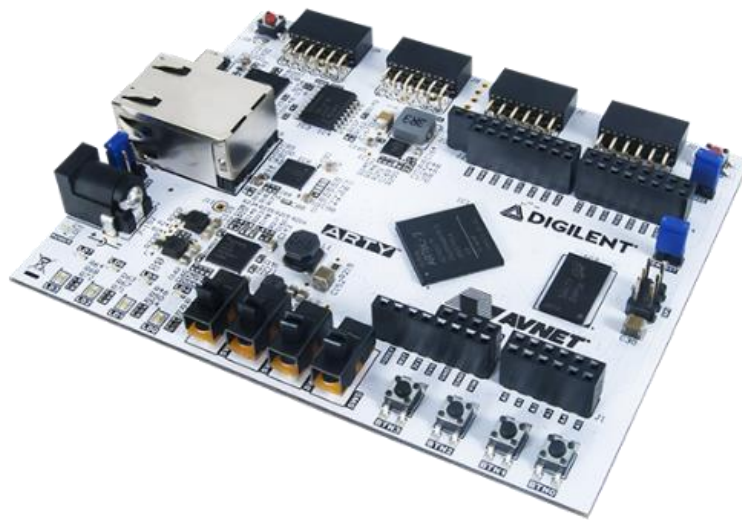


Figure D-1 Digilent[®] Arty Board.

Reference manual link: <https://reference.digilentinc.com/reference/programmable-logic/arty/reference-manual>

Xilinx[®] Artix-7 35T FPGA (part no XC7A35TICSG324-1L):

- 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops).
- 1,800 Kbits of fast block RAM.
- Five clock management tiles, each with a phase-locked loop (PLL);
- 90 DSP slices.
- Internal clock speeds exceeding 450MHz;
- On-chip analog-to-digital converter (XADC).
- On board 100 MHz clock (crystal oscillator).
- Programmable over JTAG and Quad-SPI Flash

System features:

- 256 MB DDR3L RAM @ 667 MHz.
- 16 MB Quad-SPI Flash.

- USB-JTAG Programming circuitry
- Powered from USB or any 7 V - 15 V source.

System connectivity:

- 10/100 Mbps Ethernet Interface.
- USB-UART Bridge

Interaction and Sensory Devices:

- 4 Switches
- 4 Buttons
- 1 Reset Button
- 4 LEDs
- 4 RGB LEDs

Expansion Connectors:

- 4 Pmod™ Peripheral Modules interfaces.
- Arduino/ChipKit shield connector.

Appendix E Electronic, optoelectronic and optical components



Figure E-1 Common anode tri-colour LED.

Common anode tri-colour LED (red-green-blue)

Part no: SSY-2318

Diameter: 5 mm

View Angle: approximately 25 degrees

Link to product: https://www.amazon.com/100pcs-Common-Anode-Green-SSY-2318/dp/B00M43ADEY/ref=sr_1_3?s=industrial&ie=UTF8&qid=1463500112&sr=1-3&refinements=p_4%3Abuytra

	Red	Green	Blue
Wavelength (nm)	630- 640	515- 520	465- 475
Brightness (mcd)	1000- 1200	3000- 5000	2000- 3000
Voltage (V)	1.8- 2.0	3.2- 3.4	3.2- 3.4



Figure E-2 IF-D91 photodiode.

Features:

- Connector-less style plastic fibre optic photodiode detector.
- Mates with standard 1000 μm core, 2.2 mm jacketed plastic fibre optic cable.
- Internal micro-lens for efficient optical coupling.

Table E-1 IF-D91 characteristics.

	Min.	Typ.	Max.
Wavelength for maximum Photosensitivity λ_{PEAK}	-	880 nm	-
Spectral Bandwidth ($R = 10\%$ of R_{MAX})($\Delta\lambda$)	450 nm	-	1050 nm
Rise and Fall Times (10% to 90% and 90% to 10%) ($R_L = 50 \Omega$, $V_R = 20 \text{ V}$, $\lambda = 850 \text{ nm}$)	-	15 ns	-
Total Capacitance ($\text{PIN} = 0$, $f = 1.0\text{MHz}$) (C)			
$V_R = 20 \text{ V}$	-	2 pF	-
$V_R = 0 \text{ V}$		6 pF	
Responsivity min. @ 870 nm	-	0.5 $\mu\text{A}/\mu\text{W}$	-
@ 650 nm		0.4 $\mu\text{A}/\mu\text{W}$	
Reverse Dark Current ($V_R = 30 \text{ V}$, $P_{\text{IN}} = 0$)	-	1 nA	60 nA
Reverse Breakdown Voltage	40 V	-	-



Figure E-3 GH4001 plastic optical fibre.

Plastic optical fibre

Part no: GH4001

Link to product: <http://i-fiberoptics.com/fiber-detail.php?id=47>

Table E-2 GH4001 characteristics.

		Green
Optical fibre	Core material	Poly(methyl methacrylate)
	Cladding material	Fluorinated polymer
	Core refractive index	1.49
	Refractive index profile	Step index
	Numerical Aperture	0.5
	Core diameter (typical)	980 μm
	Cladding diameter (typical)	1000 μm
Jacket	Material	Polyethylene
	Diameter (typical)	2.20 mm
Optical properties	Transmission loss (650 nm; collimated light)	170 dB/km
Mechanical characteristics	Minimum bend radius (loss \leq 0.5dB for a quarter bend)	25 mm

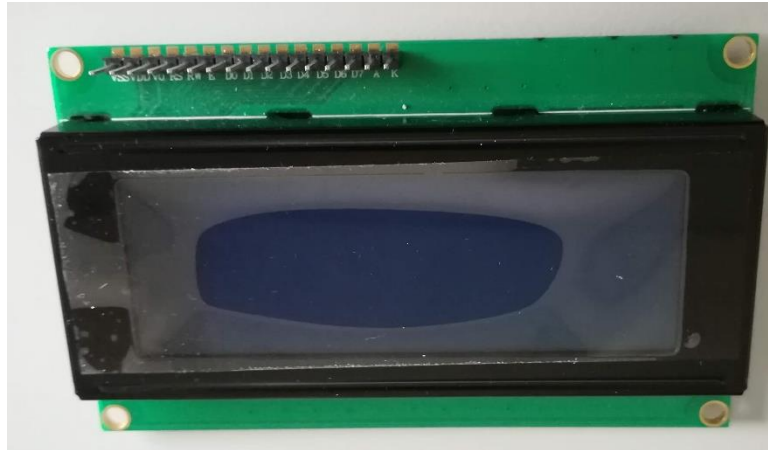


Figure E-4 LCD2004 20x4 character LCD display module.

Character LCD Display Module

Part no: LCD2004

Link to product: <http://www.lxxtech.com/lcd2004-20x4-character-lcd-display-module-5v-blue-backlight-lcd-2004a-p341.html>

Display: 20 characters × 4 lines

Character size: 29.5 mm × 47.5 mm

Operating voltage: +5 V

Operating temperature: -10 °C to +60 °C

Controller: KS0066

User manual: <https://www.lcd-module.de/eng/pdf/zubehoer/ks0066.pdf>

Appendix F Printed circuit board

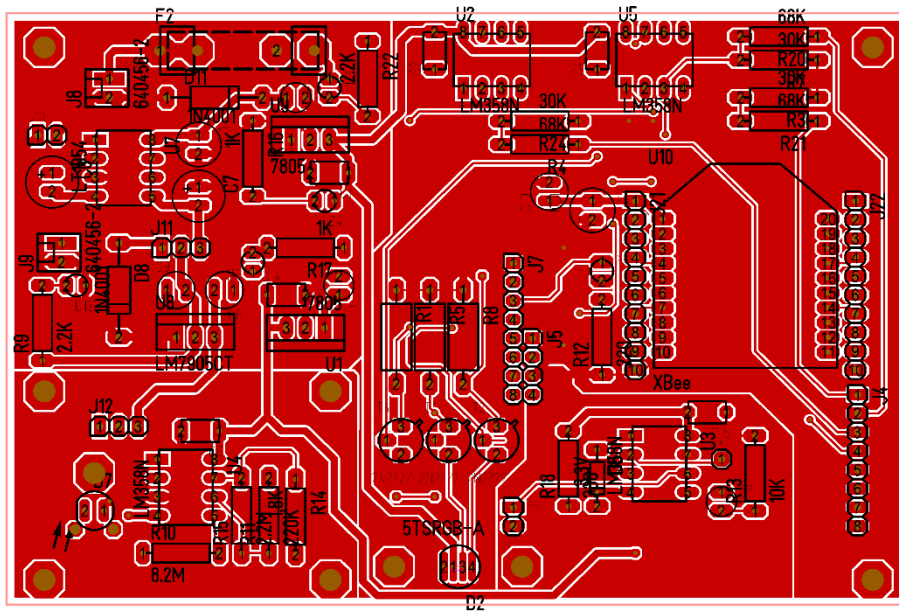


Figure F-1 PCB- top surface.

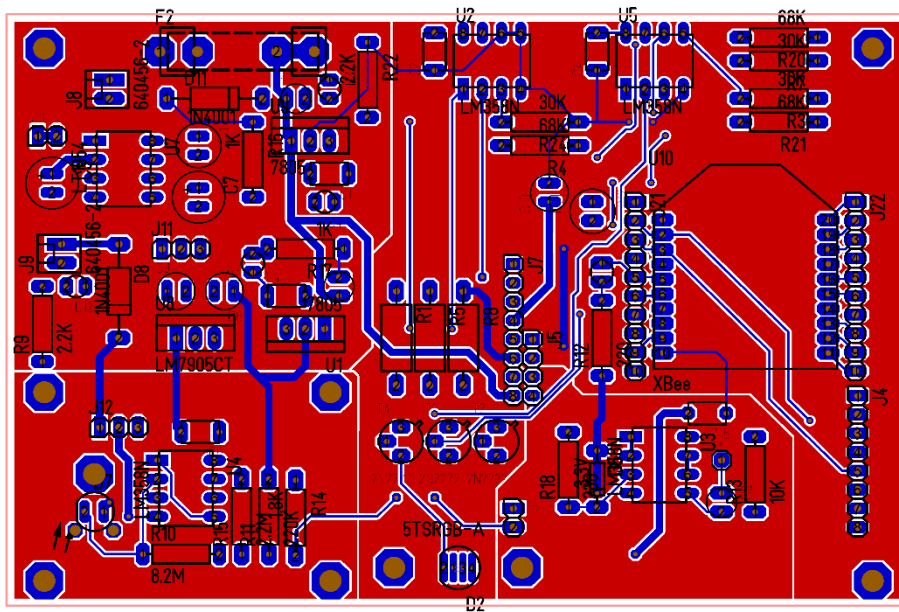


Figure F-2 PCB- bottom surface.

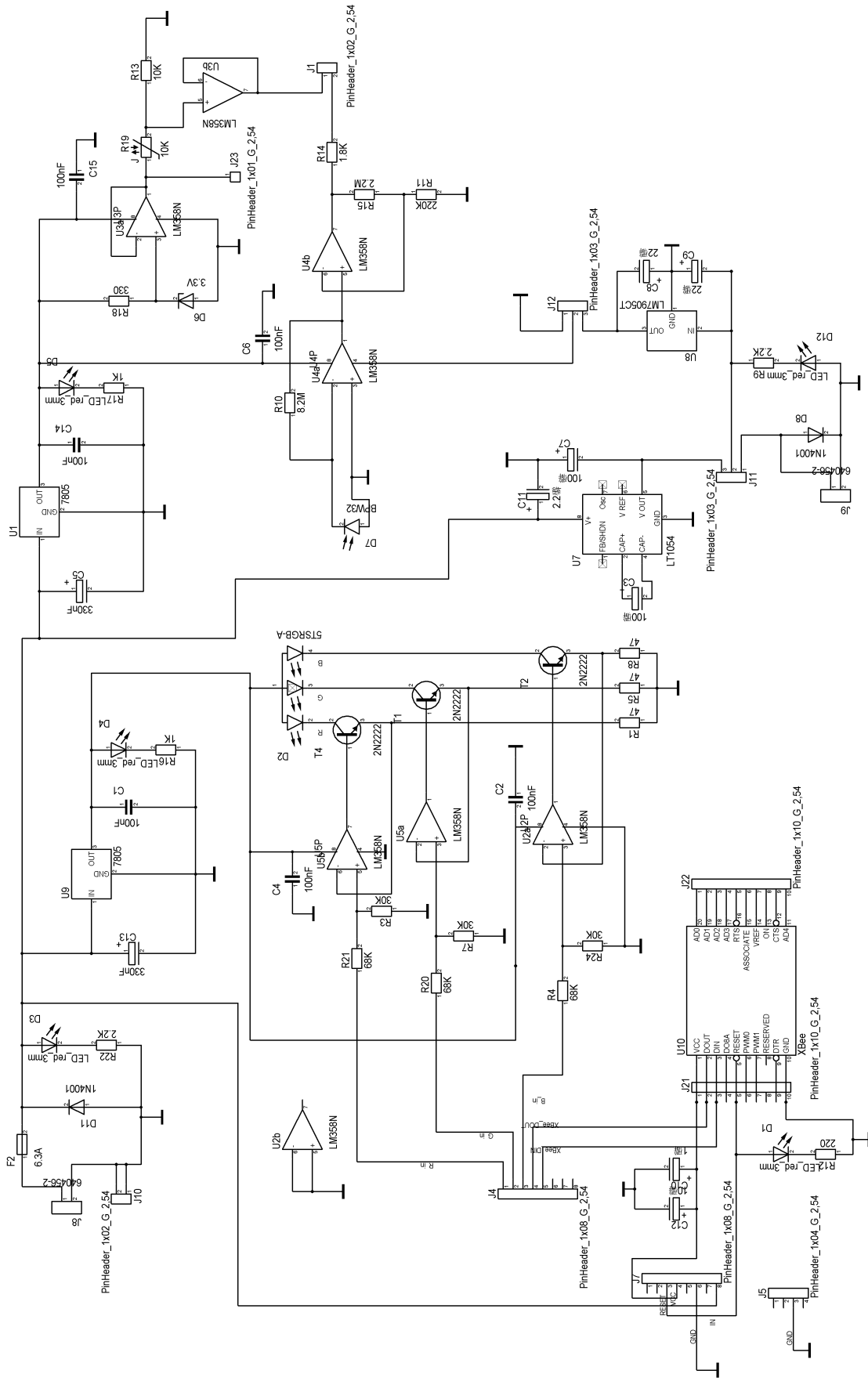


Figure F-3 Schematic of the sensor system's electronic circuit.

Appendix G Register transfer level (RTL) design description schematics

In chapter 5, the core FPGA modules for the sensor system were discussed. The simplified module connections were shown. This section presents the full RTL design description schematics for each module with the top level as listed in Table G-1.

Table G-1 List of module RTL.

Figure G-1	Light pulse generation module RTL design description schematic.
Figure G-2	LCD controller module RTL design description schematics.
Figure G-3	XADC controller module RTL design description schematics.
Figure G-4	Data processing module RTL design description schematics.
Figure G-5	Top level RTL design description schematics.

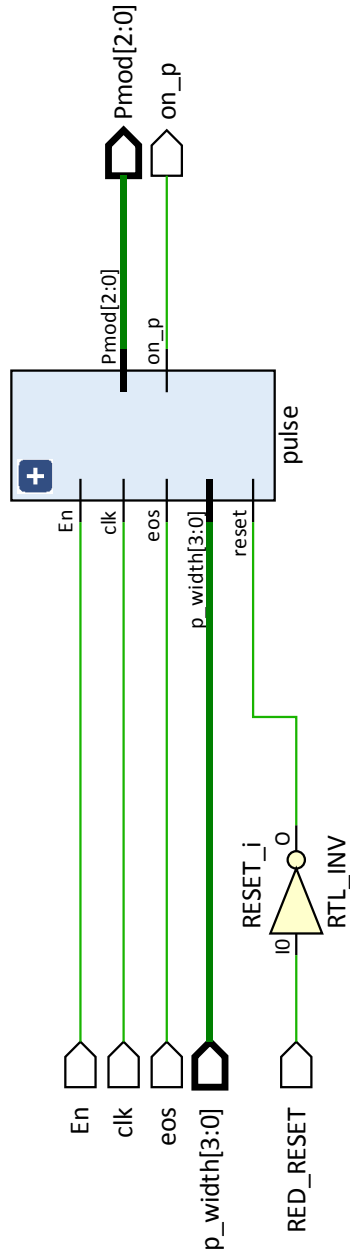


Figure G-1 Light pulse generation module RTL design description schematic.

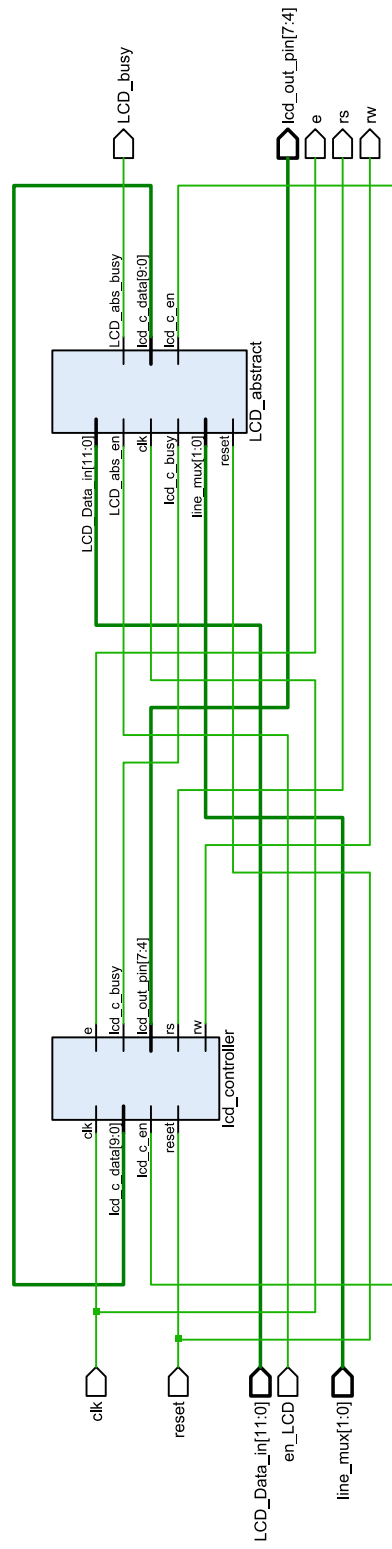


Figure G-2 LCD controller module RTL design description schematic.

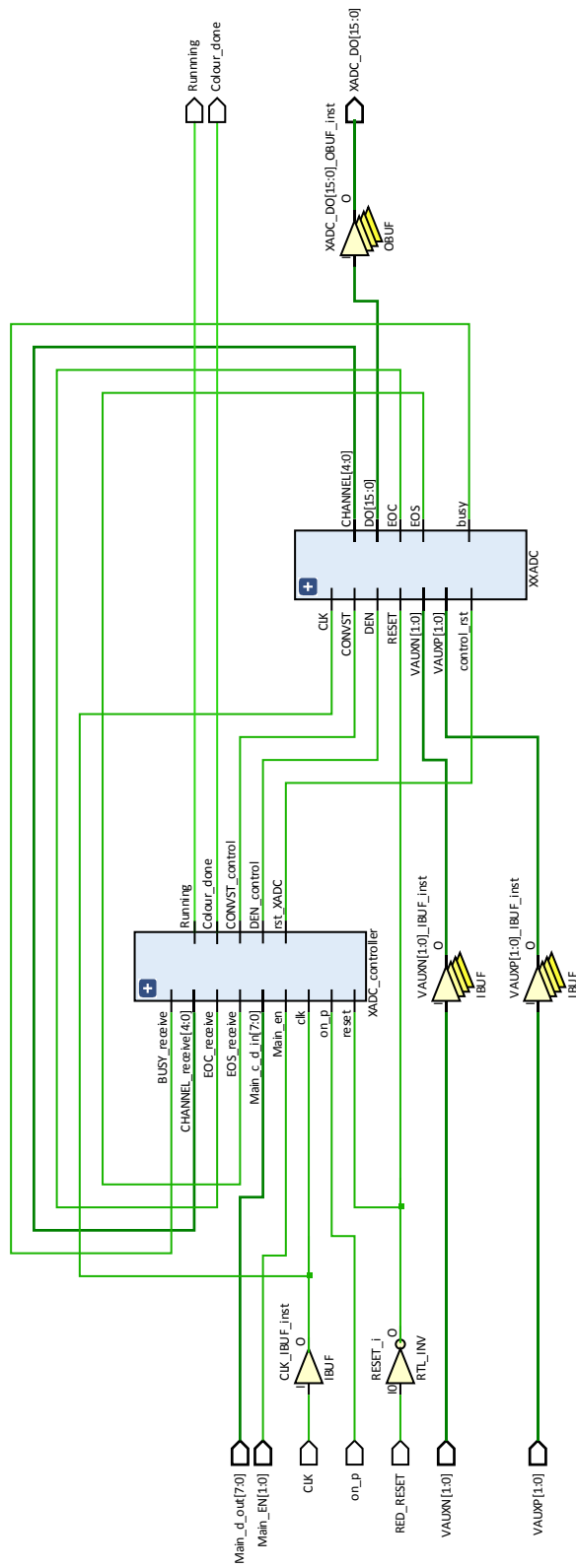


Figure G-3 XADC controller module RTL design description schematic.

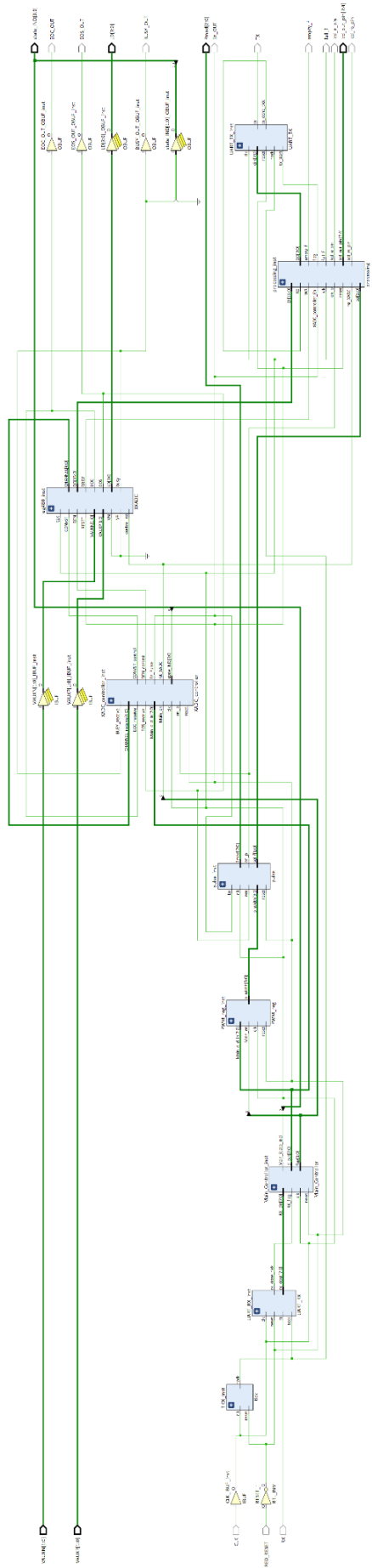


Figure G-5 Top level RTL design description schematic.