KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Software-Defined Multiple-LED Transceiver for Visible Light Communication Networks: Design and Implementation

Jorn Hendrickx

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: elektrotechniek, optie Ingebedde systemen en multimedia

Promotor:
Prof. dr. ir. Sofie Pollin

Assessoren:
Dr. ir. Nobby Stevens
Prof. dr. ir. Tinne Tuytelaars

Begeleiders:
Ir. Jona Beysens
Dr. ir. Qing Wang

Academiejaar 2017 – 2018

© Copyright KU Leuven

Without written permission of the thesis supervisor and the author it is forbidden to reproduce or adapt in any form or by any means any part of this publication. Requests for obtaining the right to reproduce or utilize parts of this publication should be addressed to ESAT, Kasteelpark Arenberg 10 postbus 2440, B-3001 Heverlee, +32-16-321130 or by email `info@esat.kuleuven.be`.

A written permission of the thesis supervisor is also required to use the methods, products, schematics and programmes described in this work for industrial or commercial use, and for submitting this publication in scientific contests.

Zonder voorafgaande schriftelijke toestemming van zowel de promotor als de auteur is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend u tot ESAT, Kasteelpark Arenberg 10 postbus 2440, B-3001 Heverlee, +32-16-321130 of via e-mail `info@esat.kuleuven.be`.

Voorafgaande schriftelijke toestemming van de promotor is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

Preface

The creative path I took in my research and the writing of my thesis has several analogies with quotes from a series of books bundled under the name "The Ultimate Hitchhiker's Guide to the Galaxy" written by Douglas Adams[1].

> "[...] a scientist must also be absolutely like a child. If he sees a thing, he must say that he sees it, whether it was what he thought he was going to see or not. See first, think later, then test. But always see first. Otherwise you will only see what you were expecting. Most scientists forget that."
>
> *So Long, and Thanks for All the Fish*

The creative path was not always smooth and to set about such challenging effort, towards a gratifying ending, is no sinecure. That is why I, first and foremost, want to thank my supervisor Prof. dr. ir. Sofie Pollin for her confidence in me, to let me work on this captivating and entailing subject of Visible Light Communication.

My supervisor Jona Beysens also merits a great share of appreciation and gratitude. He was always ready to help me on my way with wise words, assisted me with the scope of my thesis and the choices that had to be made in the pursuit of inclusiveness, comprehensiveness and wholeness. Indeed, a thesis is never finished and "Oh, I should investigate that as well!".

> "A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools."
>
> *Mostly Harmless*

Of the many people who shared their wisdom and support, I like to mention Achiel Colpaert by name. He provided the necessary support, delving into (Vivado) errors with me and making a generous timesaving effort on my behalf

> "It is a mistake to think you can solve any major problems just with potatoes."
>
> *Life, the Universe and Everything*

[1] (11 March 1952 – 11 May 2001), an English author, scriptwriter, essayist, humorist, satirist and dramatist. [1]

> "He was staring at the instruments with the air of one who is trying to convert Fahrenheit to centigrade in his head while his house is burning down."
>
> *Mostly Harmless*

The quote above captures the feeling often encountered while working on my thesis. Therefore, in times of chaos, I was all the more happy with the unwavering support of my friends at VTK, Stura and LOKO. My roommates, relatives and others not mentioned by name have also my deepest gratitude and appreciation for their support. When needed, they provided me much-needed perspective and moments of clarity.

In particular, I would like to thank Koen Hendrickx, Ph.D. and Emil Loevbak for giving me the opportunity to enrich my English.

> "There is an art, it says, or rather, a knack to flying. The knack lies in learning how to throw yourself at the ground and miss."
>
> *Life, the Universe and Everything*

Finally but most certainly not in the least, I would like to thank my parents, who have supported me and ensured that I was able to conclude my studies to become an engineer, with this manuscript being the icing on the savory cake.

In the end, working on this thesis has borne fruit: the traveled creative path has given me copious knowledge, insights and skills, to be adapted and rehashed in applications to come.

> "They were not the same eyes with which he had last looked out at this particular scene, and the brain which interpreted the images the eyes resolved was not the same brain. There had been no surgery involved, just the continual wrenching of experience."
>
> *So Long, and Thanks for All the Fish*

Jorn Hendrickx

Contents

Abstract

Visible light communication (VLC) is a fairly new and up-and-coming technology. The success of this technology is most certainly related to the vast amount of different possible applications in our daily lives. Furthermore it has become common belief that the congestion problem for the data transport in our digital networks, which is up and coming, can be relieved by communication through the visible light spectrum. Indeed, VLC offers a way out of the crowdedness of the radio frequency (RF) spectrum.

An improvement upon the current state-of-the-art was envisioned to be the focus of this thesis. A transceiver was looked-for, with good performance in throughput (at least 10 Mbit/s) and synchronization between the transmitter and receiver. These specifications needed to be achieved while still offering the necessary flexibility and cost efficiency. Therefore, in this thesis, a software-defined multiple-LED transceiver was designed and implemented in part.

On the hardware level, two development boards were chosen: a ZYBO transmitter and a ZedBoard as receiver. At the software level, it was initially the intention to adapt an already available IEEE 802.15.4 FPGA implementation to fit the IEEE 802.15.7 protocol. This turned out to be unreasonably difficult work for what was needed in this design. Therefore, it was decided to start from scratch and reuse some of the IEEE 802.15.4 code.

In a first implementation, the focus was set on a basic physical layer (PHY) with on-off keying (OOK) modulation and a switch to toggle between the three modes (off, illumination and illumination + communication mode). In a second implementation, a toggle between different clock frequencies was added. In this way, different throughputs can be achieved, depending on the needs of the application. In a final implementation, the previous implementations were added together and the number of output pins was extended to accommodate the use of multiple LED drivers. Also, toggles were added, which can be used to turn specific LED drivers on or off.

Although the software-defined multiple LED transceiver is functional, some much needed upgrade and fine tuning work is still required. Hereto, some recommended additional reading and suggestions for additions to the current system, to make it more usable and flexible, were listed.

This document also contains, in appendices, a manual to set up a virtual machine, install Vivado and compile plus run the code implemented in this thesis.

Samenvatting

Communicatie via zichtbaar licht, (visible light communication, verder VLC genoemd) is een recent opkomende technologie die succes kent door de vele toepassing in het dagelijks leven. Zo loopt er onder andere onderzoek naar het gebruik van VLC in de automobiel- en transportsector, wordt er naar VLC gekeken om binnenshuis locaties te bepalen en wordt VLC gezien als aanvullend communicatie kanaal.
De belangrijkste reden om het gebruik van VLC te starten, is de uitweg die de communicatie via het zichtbaar spectrum biedt voor de steeds drukker wordende interacties via het radiofrequentiespectrum. Er wordt meer en meer mobiele data verbruikt, dat binnenshuis steeds vaker wordt overgedragen van het mobiele netwerk naar de vaste netwerken, zoals Wi-Fi. Daarnaast wordt het Wi-Fi netwerk zelf ook al meer belast door het groeiend aantal toestellen dat hier gebruik van kan maken: smartphones, laptops, tablets, huishoudelijke apparatuur, IoT-sensoren... Beide zorgen ze ervoor dat dit deel van het spectrum overvol zal geraken [2].
Andere redenen waarom het gebruik van VLC favorabel zou zijn, zijn het feit dat VLC gebruik maakt van LEDs en dat VLC voor meer veiligheid zorgt. LEDs zijn goedkoper en efficiënter dan traditionele lichtbronnen en worden daarenboven steeds vaker gebruikt in gebouwen, waardoor de installatiekosten laag kunnen blijven. Daarnaast heeft zichtbaar licht geen elektromagnetische interferentie en kan het niet door muren propageren, wat VLC een meer lokaal karakter geeft.

In hoofdstuk 1 wordt verder een overzicht gegeven van het reeds bestaande onderzoek naar VLC. Zo worden OpenVLC [3] en modBulb [4] geïntroduceerd als goedkope, open source platforms die gebruikt worden in het onderzoek naar VLC. Daarnaast wordt de evolutie geschetst van het onderzoek naar hoge snelheden, gebruik makende van VLC. Hieruit wordt geconcludeerd dat er al veel VLC platformen bestaan, maar dat deze ofwel goedkoop maar niet performant zijn, ofwel zeer duur en met beperkte gebruiksmogelijkheden.
Er moet dus gezocht worden naar een platform dat flexibel is, een redelijke snelheid kan halen over een zekere afstand én betaalbaar blijft. Onder flexibiliteit wordt het kunnen gebruiken van verschillende modulatie technieken, het aansturen van verschillende LED drivers en de mogelijkheid om het systeem als zender en ontvanger te gebruiken, verstaan. Het ligt in de verwachting, om in een volledig functioneel platform, LEDs toegewezen moeten kunnen worden aan specifieke gebruikers. Afhankelijk van de densiteit en beweging van deze gebruikers, zal er dan gekozen kunnen worden welke instellingen voor die situatie het beste werken. Er is dus sprake van

een software-gedefinieerd platform. Verder wordt er gericht op 10 Mbit/s over een afstand van 1,5 meter. Hiervoor zullen de rekenintensieve delen van de code snel moeten worden afgehandeld, waardoor het gebruik van een FPGA het meest logische is. Dit brengt ons bij de onderzoeksvraag van deze thesis, namelijk het ontwerp en de implementatie van een software-gedefinieerde transceiver voor VLC dewelke gebruik kan maken van meerdere LEDs.

In hoofdstuk 2 wordt hiertoe een overzicht gegeven van hoe het platform eruit zou zien.
Wat hardware betreft, wordt er een onderscheid gemaakt tussen de zender en de ontvanger. Voor de zender werd er gekozen voor een ZYBO FPGA, waarbij de Pmod pinnen gebruikt worden om de LED drivers op aan te sluiten. Deze LED drivers, waarvan er tot 16 aangesloten kunnen worden, kunnen in 2 modi werken: een modus waarin er enkel licht wordt uitgezonden en een modus waarin er ook aan communicatie gedaan wordt. Voor de ontvanger werd er gekozen voor een ZedBoard aangezien deze FPGA beschikt over een FMC connector. Deze connector is nodig om de aan hoge snelheid werkende analoog-digitaalomzetter aan te kunnen sluiten.
Wat software betreft, werd gekozen voor Embedded Linux om op de CPU van de FPGA te draaien. Daarnaast zal er ook software op de programmeerbare logica (PL) van de FPGA moeten draaien. Hiervoor zal gebruik worden gemaakt van een gedeeltelijke implementatie van het IEEE 802.15.7 protocol [5]. Meerdere van deze implementaties werden onderzocht, maar uiteindelijk werd er gekozen om te vertrekken van een reeds binnen de onderzoeksgroep beschikbare IEEE 802.15.4 [6] implementatie.

In hoofdstuk 3 worden het IEEE 802.15.7-2011 en het IEEE 802.15.4-2015 protocol in meer detail besproken, alsook worden de verschillen tussen de twee protocollen opgelijst. Daarnaast wordt de IEEE 802.15.4 implementatie besproken en de delen opgetekend. Uit de opgelijste verschillen zal echter blijken dat er meer verschillen tussen de twee protocollen zijn, dan initieel gedacht. Hierdoor zal niet meer geprobeerd worden om de reeds bestaande IEEE 802.15.4 implementatie aan te passen, maar zal er eerder helemaal opnieuw worden begonnen, mits gebruik te maken van enkele delen uit de IEEE 802.15.4 implementatie.

In hoofdstuk 4 wordt vervolgens beschreven hoe de code werd geschreven, waarom bepaalde beslissingen werden genomen en hoe de code werd getest.
Wegens voornamelijk gemakkelijk te implementeren, werd er gekozen voor on-off keying (OOK). Hiertoe werden twee output pinnen, JC1 (V15) en JC2 (W15) en twee input pinnen, JC7 (W14) en JC8 (Y14) voorzien. JC1 kreeg het signaal voor enkel licht en JC2 kreeg het signaal voor communicatie toegewezen. De input pinnen werden niet gebruikt, maar werden initieel voorzien om eventueel te kunnen gebruiken tijdens het testen, namelijk door JC1 aan JC7 en JC2 aan JC8 te verbinden. Om te wisselen tussen de verschillende modi werden eerst de schakelaars gebruikt en later de AXI interface. De code voor deze AXI interface werd overgenomen uit de reeds bestaande IEEE 802.15.4 implementatie.

Na meerdere testen, waaruit bleek dat het in VHDL gemakkelijk is om redeneerfouten te maken aangezien code hier parallel wordt uitgevoerd, bevatte de uiteindelijke implementatie de volgende functionaliteit: het veranderen van de modus via commando's die verstuurd werden in XSDB en zich van de CPU, via de AXI interface naar de PL verplaatsen; het veranderen van de klokfrequentie doormiddel van de stand van de schakelaars; in communicatie modus werd een gehard-coded signaal verzonden; pinnen voor het maximum aantal aan LED drivers werden toegevoegd, alsook gehard-coded toggles waarmee de LED drivers individueel kunnen worden uitgeschakeld.
De correcte uitvoering van de code werd nagegaan door middel van een oscilloscoop die aan de output pinnen werd aangesloten. Voor de laatste implementatie werd er ook een testbank bestand aangemaakt, maar dit werkte uiteindelijk niet.

In hoofdstuk 5 worden uiteindelijk nog enkele aanpassingen voorgesteld die nodig zijn om het VLC systeem effectief te kunnen gebruiken bij onderzoek. Zo berust het merendeel van deze aanpassingen op het uitbreiden van de AXI interface: deze zou gebruikt moeten worden om meer instellingen te kunnen veranderen en wanneer er een data stroom van hoger gelegen OSI lagen op de CPU naar de PL (of omgekeerd) wordt gezonden, zal een hogere doorvoersnelheid nodig zijn. Daarnaast wordt er ook nog voorgesteld om het systeem te laten opstarten vanaf een SD kaartje, in plaats van de huidige manier waarbij de software elke keer via XSDB geladen moet worden. Ook het toevoegen van ontvang functionaliteit zal nog moeten gebeuren.

In appendix A wordt nog een overzicht gegeven van de gebruikte software en hoe deze te installeren en te gebruiken.

Om af te sluiten kan er worden gesteld dat een software-gedefinieerde transceiver voor VLC, die kan gebruik maken van verschillende LEDs, ontworpen is. Om de implementatie ervan te kunnen gebruiken in verder wetenschappelijk onderzoek naar VLC, is er additioneel werk nodig. Zo zal in de verderzetting van het huidige werk zeker de AXI interface verder uitgebreid moeten worden en zal de ontvang functionaliteit nog geïmplementeerd moeten worden.

List of Figures

List of Tables

Nomenclature

AAL	ATM Adaptation Layer
ADC	analog-to-digital converter
ARQ	automatic repeat request
ASK	amplitude shift keying
ATM	Asynchronous Transfer Mode
AXI	ARM Advanced eXtensible Interface
BFSK	binary frequency-shift keying
BPSK	binary phase-shift keying
CCA	clear channel assessment
CPU	central processing unit
CRC	cyclic redundancy check
CSK	color-shift keying
CSMA-CA	carrier sense multiple access with collision avoidance
CSS	chirp spread spectrum
CTM	channel timing management
CVD	color visibility dimming
DME	device management entity
DMT	Discrete Multi-Tone
DPPM	differential pulse-position modulation
DSME	deterministic and synchronous multi-channel extension
DSSS	direct sequence spread spectrum

FMC	FPGA Mezzanine Card
FPGA	Field Programmable Gate Array
FSK	frequency shift keying
FSM	final state machine
GFSK	Gaussian frequency-shift keying
GUI	graphical user interface
HRP	high rate pulse repetition frequency
IE	information element
IoT	Internet of Things
LD	laser diodes
LE	low energy
LECIM	low-energy, critical infrastructure monitoring
LED	light-emitting diode
LLC layer	logical link layer
LMR	land mobile radio
LOS	line of sight
LPC	Low Pin Count
LRP	low rate pulse repetition frequency
MAC	Media Access Control
MCU	microcontroller unit
MIMO	multiple-input, multiple-output
MIO	multiplexed I/O
MPPM	multipulse pulse-position modulation
MSK	minimum shift keying
Msps	million samples per second
MUX	multiplexer
O-QPSK	offset quadrature phase-shift keying

OFDM	orthogonal frequency-division multiplexing
OOK	on-off keying
OPPM	overlapping pulse-position modulation
OS	operating system
PAM	pulse-amplitude modulation
PAN	personal area networks
PCA	priority channel access
PHY	physical layer
PL	programmable logic
Pmod	peripheral module
PS	processing system
PSDU	PHY service data unit
RAM	random-access memory
RCC	rail communications and control
RF	radio frequency
RGB LEDs	Red Green Blue LEDs
RLL	run length limited
RSSI	Received Signal Strength Indicator
RX	receive, receiver, reception
SEM	Single Edge Position Modulation
SFD	start frame delimiter
SNR	signal-to-noise
SPI	Serial Peripheral Interface
SSCS	Service Specific Convergence Sublayer
SUN	smart utility network
TSCH	timeslotted channel hopping
TVWS	television white space

TX	transmit, transmitter, transmission
UWB	ultra-wide band
VLC	visible light communication
VM	virtual machine
VPPM	variable pulse-position modulation
WDM	wavelength-division multiplexing
Wi-Fi	wireless fidelity
ZedBoard	Zynq™ evaluation and development board
ZYBO	Zynq™ board

Chapter 1

Prologue and research question

Visible light communication is a fairly new and up-and-coming technology. Its success lies in the vast amount of different possible applications in our daily lives. Since the practical applications of visible light communication are not common knowledge, this chapter starts with highlighting the advantages and disadvantages of this technology. These chosen practical cases are examples of the use of this technology for the increase of safety in traffic and the day to day activities in our household. In a second and third section, the latest research into these new applications of visible light communication will be reviewed and the research question of this thesis will be formulated.

1.1 Painting some pictures

Imagine yourself driving home from daily work and approaching a traffic light. The LED lights in this traffic light communicate with the LED headlights of your car. Immediately upon establishing the communication, a timer appears on the dashboard of your car, counting down until you can cross the traffic intersection.

Upon arrival at your home, the garage door opens automatically after communication of the same LEDs in your headlights with the light above your garage.

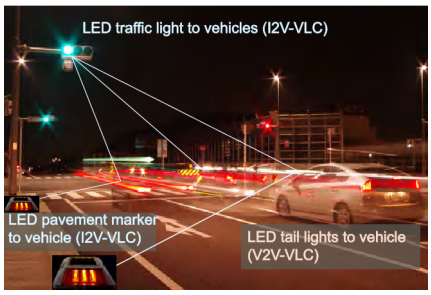


Figure 1.1: *Visible light communications (VLC), to support automotive and smart mobility applications.* Adopted from [7].

Sitting in your sofa on a bright afternoon, you take your tablet and put the shutters down. The instruction for this action was communicated from your tablet via an LED light with the lights in the living room to the central module in your home, realizing a more comfortable climate in your living quarters. Before turning in for the night, you watch a YouTube movie, which is streamed from the lights in your room to your tablet or television.

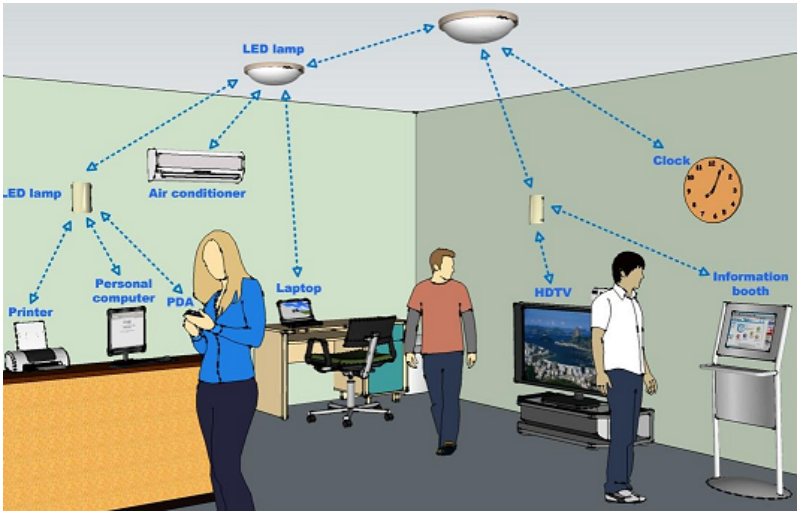


Figure 1.2: Example of VLC, adopted from [8].

Lying in bed with the lights already off, but you want to verify if the coffee machine is set correctly for the next morning. This communication, from your smartphone to the lamp on your nightstand and further to the coffee machine, is also established via Visible Light Communication.

Taken from [5], VLC can be described as follows: "*Visible-light communication (VLC) transmits data by intensity modulating optical sources, such as light-emitting diodes (LEDs) and laser diodes (LDs), faster than the persistence of the human eye. VLC merges lighting and data communications in applications such as area lighting, signboards, streetlights, vehicles, traffic signals, and status indicators.*"

1.2 Why VLC?

There are many use-cases for VLC, both inside and outside the house. In Cisco's Global Mobile Data Traffic Forecast Update [2], it is said that much mobile data activity takes place within user's homes. When broadband and Wi-Fi access points, or operator-owned femto- and picocells, are available, a substantial portion of this traffic, generated by mobile and portable devices, is offloaded from the mobile network to the fixed network. This offload of traffic "*from dual-mode devices (excluding laptops) over Wi-Fi or small-cell networks*", is illustrated in figure 1.3.

In addition to this viewpoint, the demand for wireless data doubles each year, as cited in [9].

Taking these two viewpoints together, it becomes clear that a congestion problem in the digital network for the transport of data is coming. The next subsections will give some indications how and why VLC could provide a solution for this.

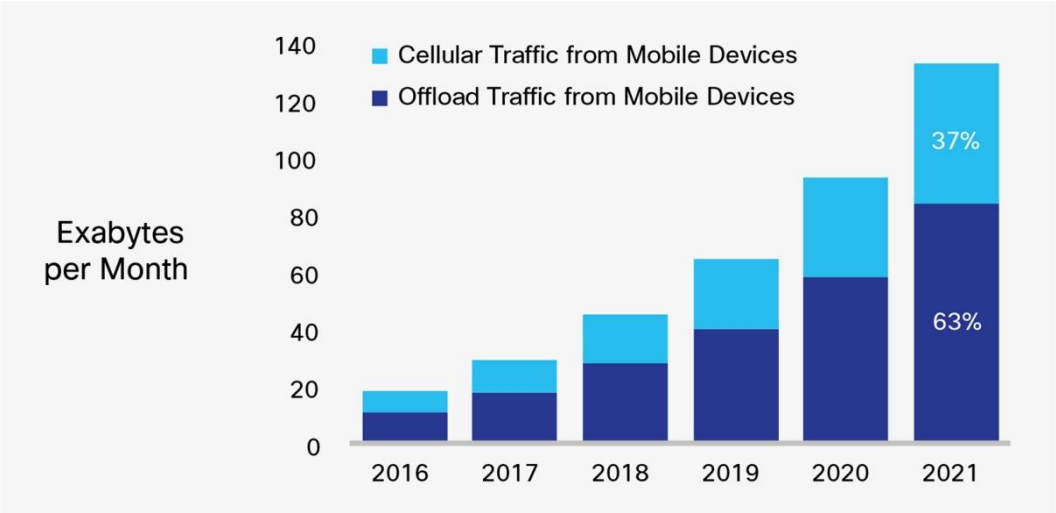


Figure 1.3: *By 2021, 63 Percent of Total Mobile Data Traffic Will Be Offloaded*, adopted from [2].

1.2.1 Relief of spectra

Nowadays, homes are increasingly more packed with different intelligent devices including smartphones, tablets, smart fridges, smart washing machines and other IoT devices. All those devices have in common that they use the Wi-Fi spectrum, which is located at 2.4 GHz and 5 GHz, or other Radio Frequency (RF) spectrums, which are getting more and more crowded. Millimeter wave (mmWave - 28 GHz) [10], infrared light and especially visible light offer a way out of the jam-packed frequencies. The latter of the three suggestions is better known as VLC, which uses the visible spectrum (wavelengths of 390-750 nm [11, p. 146]) and spans the 400 THz to 800 THz region of license free bandwidth.

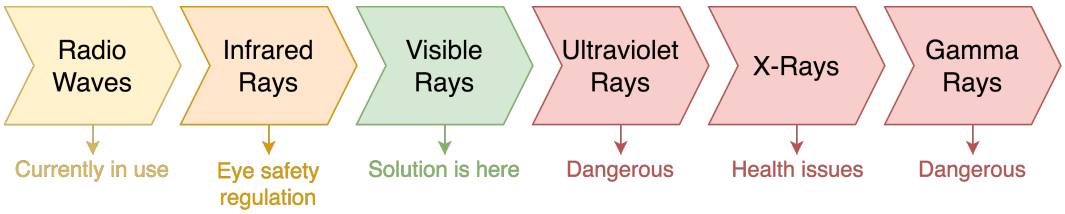


Figure 1.4: An overview of the spectrum and accompanying issues.

1.2.2 Use of LEDs

For VLC, mostly LEDs are used because several advantages are attributed to this lightsource. Several of these advantages are detailed in [12]: When LEDs are compared with traditional light sources, a lower price, a longer life and a higher lighting efficiency are recorded. This higher efficiency is a result of the fact that LEDs have no out-of-visible-band optical spectrum, "*unlike incandescent lamps, which emit*

much light in the infrared spectrum, and fluorescent lamps with additional ultraviolet spectrum" [12]. Additionally, LEDs are more environment-friendly, in part because they have a lower power consumption.

Furthermore other unique characteristics of LEDs make them the prime candidate for VLC. These additional characteristics are also described in [12]: LEDs have a high-speed switching capability, which make them not only useful as light source, but also for communication. Also, it is possible to separate the visible-band optical spectrum by using multiple color-band LEDs, since there are various visible bands of LEDs.

In view of the practical application and the installation of VLC, it is also apparent that communication via LEDs can make use of existing facilities in homes or outdoors, resulting in a lower deployment cost.

1.2.3 Security

Another advantage of using visible light is the lack of electromagnetic interference and the inability to propagate through walls. This characteristic gives VLC a more local spatial character, with its decrease of the risk in crosstalk with other frequencies and its confinement to the room where the original message was launched.

The former is explained by the fact that optical radiation does not interfere with radio waves and therefore VLC can be used in areas in which electromagnetic radiation must be avoided, e.g. airplanes, hospitals, petrochemical and nuclear power plants [10], or to supplement the existing Wi-Fi network in those areas.

The latter offers an additional level of network security and can also be exploited to reduce or eliminate interference between neighboring indoor femto- and/or pico-cells [10].

1.3 State-of-the-art

In this section, first of all, an overview of the existing research concerning VLC is given. This overview contains the open-source VLC platforms and the research into high throughput VLC. Also, some VLC specificities will be reviewed. Finally, this synopsis allows to draw a conclusion on the current developments in VLC and sets the scope of this thesis.

1.3.1 Low-end open-source VLC platforms

In this subsection, two low-end open-source VLC platforms, which are available to the public at the present moment, will be introduced: OpenVLC [3] and modBulb [4]. Both platforms are available at rather low cost with pricings between 15 and 70 euro.

OpenVLC is a platform, running on a BeagleBone Black (± 55 euro), in which the PHY and MAC layers are implemented as Linux drivers. Version 1.1 could handle speeds of around 2.2 Kb/s at distances up to 1 m. Further development to version 1.3 allows a throughput of at least 400 Kb/s at a distance of more than 1 meter, according to [3].

modBulb is a platform which uses an MCU or an FPGA. As MCU the Texas Instruments CC3200 System on Chip (SoC) is chosen, which has a maximum clock frequency of 80 MHz and costs less than 10 euro. As FPGA the Microsemi AGLN250 is chosen, which is used at 20 MHz and costs less than 20 euro. The platform can handle speeds up to 1 Mbit/s at a distance up to 1.8 m and can be updated via Wi-Fi.

In this work, a low-end VLC platform will be created, which can accommodate higher throughputs than the OpenVLC and modBulb platforms. This VLC platform will be open-source with respect to software, as well as hardware.

1.3.2 High throughput VLC platforms

In addition to the aforementioned platforms, which are characterized by relatively low speeds, there are also publications in which specific designs are proposed that can handle much higher speeds. An overview of these platforms can be found in table 1.1 below, with the specifics on the design including: distance between transmitter and receiver, the modulation used and whether white or colored LEDs were used.

The first designs mentioned in the table, are platforms in which LEDs are used that create white light with the aid of phosphorus. Then, different designs are listed that use LEDs in which white light is created by merging red, green and blue colors (RGB LEDs).

In publication [13], it is mentioned that a phosphor-based LED typically consists of a blue LED chip covered by a layer of yellow phosphor, which emits the white light. However, the long relaxation time of the yellow phosphor leads to a limited modulation bandwidth and consequently a lower transmission capacity. In order to avoid this disadvantage, it is recommended to use a blue filter before the photo-diode to increase the modulation bandwidth of the LED.

In [12] it is concluded that RGB LEDs are faster than phosphor LEDs. In addition to this advantage, the RGB LEDs can also be varied in color and intensity, opening a much larger variety of possibilities for modulation.

Phosphor LEDs
In [14], a publication already dating from 2009, a VLC system is proposed which realizes 125 Mbit/s at 5 m distance. In this publication, on-off keying (OOK) is used.

Year	LED	Throughput	Distance	Modulation	Publication
2009	phosphor	125 Mbit/s	5 m	OOK	[14]
2010	phosphor	230 Mbit/s	0.22 - 0.44 m	OOK	[15]
2010	phosphor	513 Mbit/s	0.3 m	DMT	[16]
2011	RGB	803 Mbit/s	0.12 m	DMT / WDM	[18]
2012	phosphor	1 Gbit/s	not mentioned	DMT	[17]
2012	RGB	1.25 Gbit/s	0.1 m	DMT / WDM	[19]
2012	RGB	1.5 - 3.4 Gbit/s	not mentioned	DMT - WDM	[20]
2013	RGB	1.1 Gbit/s	1 m	4 channel MIMO-OFDM	[21]
2014	50 μm gallium nitride	3.22 (3.5) Gbit/s	5 cm	OFDM	[22]

Table 1.1: Overview of publications about high throughput VLC platform designs.

In 2010, a publication [15] reported a throughput of 230 Mbit/s over a distance of 0.22 to 0.44 m, with use of the same OOK technique. In a different publication from 2010, [16], Discrete Multi-Tone (DMT) is used to achieve a higher throughput of 513 Mbit/s over a distance of 0.3 m.

In a later publication from 2012 [17], a throughput of 1 Gbit/s was reported, using DMT and line of sight (LOS). The distance between the transmitter and the receiver was not mentioned.

RGB LEDs
In a publication from 2011 [18], a throughput of 803 Mbit/s was reported over a distance of 0.12 m between transmitter and receiver. As modulation DMT and wavelength-division multiplexing (WDM) were used.

In a publication from 2012 [19], a throughput of 1.25 Gbit/s was reported over a distance of 0.1 m, also using WDM and DMT. In the same year a different publication [20] mentioned, on the other hand, throughputs of 1.5 Gbit/s (using single channel DMT) and 3.4 Gbit/s (using WDM). The distance between the transmitter and the receiver was not mentioned.

Later, in 2013, a publication [21] reported a throughput of 1.1 Gbit/s over a 1 meter distance, using 4 channel multiple-input, multiple-output orthogonal frequency-division multiplexing (MIMO-OFDM).

Special LEDs
In 2014, a publication [22] mentioned throughputs up to 3.22 Gbit/s at 5 cm distance, while [10] even a throughput of 3.5 Gbit/s for the same publication reported. OFDM and a single 50 μm gallium nitride LED were used.

1.3.3 Specificities

To conclude, two applications of VLC will be presented, Epsilon [23] and darkLight [24], along with a brief overview of publications on different modulation techniques.

Epsilon
In [23] (2014) "A Visible Light Based Positioning System" is presented. The system has been tested at distances between 0.4 and 1.1 m, but throughputs were not mentioned. Binary frequency-shift keying (BFSK) and channel hopping were used.

darkLight
In [24] (2016) "Visible Light Communication in the Dark" is presented. In this paper "encoded data into ultra-short, imperceptible (extremely-low luminance) light pulses" and overlapping pulse-position modulation (OPPM) were used, which led to a throughput of 1.6 $Kbit/s$ over a distance of 1.3 m.

Different modulation techniques
In addition to the publications above, there are more publications on modulation techniques, among others: OPPM (1984) [25], multipulse pulse-position modulation (MPPM) (1989) [26], differential pulse-position modulation (DPPM) (1999) [27], OFDM (2015) [28] and color-shift keying (CSK) (2015) [29]. This list is by no means exhaustive.

1.3.4 Conclusion

From the state-of-the art, it can be concluded that the obtained throughput has increased over the years with more technical insights, use of different modulation techniques and the introduction of different LEDs. However, the practical adoption and introduction of the more recent VLC systems would be very expensive and can only be used in specific situations, according the limits in throughput and distance between the transmitter and the receiver.

Also, in more recent years a vast proliferation of different modulation techniques for the VLC systems can be noted. This is most certainly an indication of the growing interest in VLC systems and the application thereof.

In summary, as presented in the above state-of-the-art research, there is an abundance of different VLC systems, which are either low in cost, with a low performance, or either more costly, with a limited application potential.

The listed information on VLC systems above will be used to formulate the research question of this thesis in the next section.

1.4 Research question

In order to formulate the research question, the characteristics of the desired VLC test platform must be listed and defined.

1.4.1 Characteristics

From the state-of-the-art it becomes clear what the characteristics of a good VLC test system would be. The measurement of throughput and a practical usable distance between the transmitter and the receiver are the key physical parameters to be taken into account for this VLC test system. However, flexibility and cost of the VLC test system are additional parameters to be reviewed and to be taken into account for the construction of the test platform.

Flexibility
In order to advance the research in VLC much faster, there is a need for a flexible, but well priced, system. The main focus should be on testing of various modulation techniques and to use different available LED drivers. Another additional point of focus is the possibility to use this VLC test system as a transceiver: both for transmitting and receiving data. Ultimately, in a fully functional platform, it should be possible to assign the LEDs to specific users. Depending on the density and movement of these users, it would then be possible to choose which modules work best for that situation.

This type of flexibility can only be achieved by considering the software as the most essential part of the testing platform. Switching between different settings, including modulation techniques, the number of LEDs used and the operating mode (receiving or transmitting), should be easy and readily available in the software. Hence a **software-defined design** is important. In this design the software must be divided into easy swappable modules or modules which can easily be turned on or off.

Throughput
Another specification is the throughput. In the scope of this work, a throughput of 10 Mbit/s is targeted. To achieve such throughput, the software must work as quickly and efficiently as possible. Thus, it is recommended to have the calculation-intensive parts handled by an **FPGA**. After all, for these parts, an FPGA delivers more performance than a microcontroller [4]. Consequently, parts of the VLC protocol will have to be synthesized for an FPGA. A well suited modulation technique must be chosen too, which will be discussed in chapter 3.

Another way to increase the throughput is increasing the number of LEDs, because combining multiple LEDs will increase the signal strength which in turn will lead to a better signal-to-noise (SNR) ratio. Increasing the number of LEDs can also enable the use of multiple channels at different frequencies when using pulse-amplitude modulation (PAM). For this type of modulation an FPGA is best suited because it

has multiple ports which can be used to connect multiple LED drivers.

Distance
In the scope of this work, a distance of about **1.5 m** will be taken, since this is the common distance between the ceiling (transmitter) and a user (receiver).

Cost
The VLC test system must not be too expensive. Financial resources for this type of research are limited in the early stages of the development. Therefore, testing will be limited to readily available hardware and software. An affordable test system also ensures that other research groups can adopt this system.

1.4.2 Conclusion

The primary goal of this thesis is the construction of a software-defined multiple-LED transceiver. This transceiver must show good performance in speed, throughput and synchronization between the transmitter and receiver, while still offering the necessary flexibility and cost efficiency.

This document is structured as follows. In chapter 2 the hardware and software design of the VLC test system will be reviewed. For the software, a partial FPGA implementation of the IEEE 802.15.7-2011 protocol will be implemented and tested in chapter 4. To this end, a complete description of an in-house available IEEE 802.15.4 implementation, as well as a comparison between IEEE 802.15.4-2015 and IEEE 802.15.7-2011, will be made in chapter 3. Finally, future work will be listed in chapter 5.

Chapter 2

Platform design

In this chapter, a brief overview of the testing platform and its components will be given. The hardware will be reviewed in the first section and in the second section the software will be described.

2.1 Hardware

A VLC test system has two basic components: a transmitter and a receiver. Both components will be discussed in more detail in the following subsections.

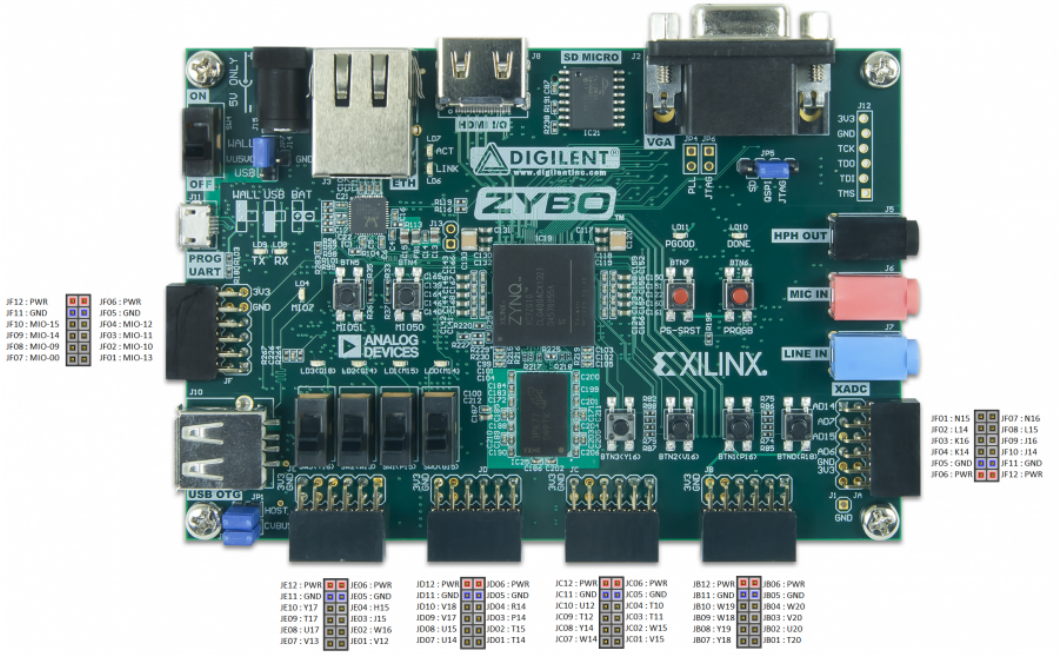


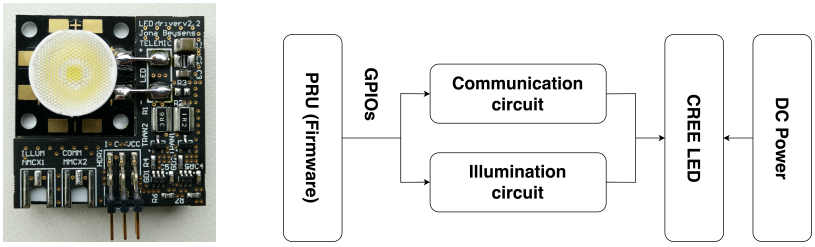
Figure 2.1: Top view of the ZYBO board with the Pmod pins annotated.

2.1.1 Transmitter

The transmitter consists of an FPGA plus CPU, the LED drivers and the LEDs.

As FPGA a ZYBO [30] is chosen, as depicted in figure 2.1. The price of this development board is about 160 euro [31]. This board has a Dual-core ARM Cortex-A9 processor for the central processing unit (CPU) or processing system (PS), at a working speed of 50 MHz, and a Xilinx 7-series Field Programmable Gate Array logic for the FPGA logic, or programmable logic (PL), at a working speed of 125 MHz. In addition, the ZYBO features also an ethernet port, which can be used to provide software updates to the system, and 6 Pmods [32]: 1 MIO connected, 1 XADC and 4 high speed Pmods.

Currently, the LED driver, depicted in figure 2.2 and designed by Jona Beysens, has two operation modes: one which provides only illumination and one which provides both illumination and communication. Here $I_l = 0$, so that the LED can be shut off completely, I_b is the illuminance of the LED, approximately 500 mA, and I_h is approximately 1.2 A, which is chosen in such way that the illuminance of the LED at 50% duty cycle (50% on and 50% off) is equal to the illuminance in *illumination mode*. Each of the two modes has a separate input pin and the third input pin is the ground. In figure 2.3, the two modes are pictured in a graph.



(a) PCB. (b) Block diagram.

Figure 2.2: Design of the TX front-end.

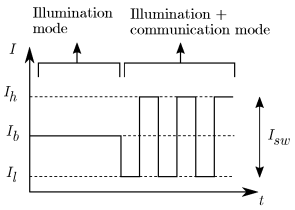


Figure 2.3: Operating modes of the LED driver.

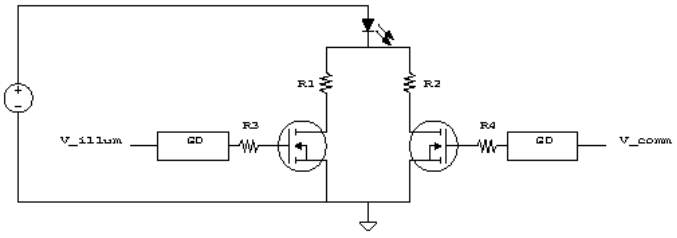
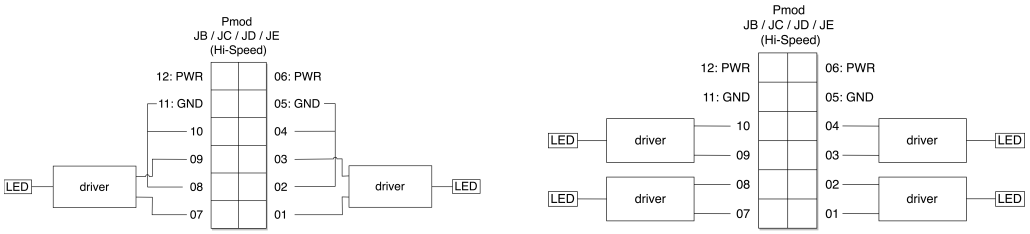


Figure 2.4: LED driver circuit.

As can be seen in figure 2.4, currents I_b and I_h are realized by a circuit consisting of a resistor and a MOSFET transistor in series. To be able to switch quickly, the MOSFET is controlled by a gate driver. The combination of the LED driver and the

CREE XT-E LED [33] achieves a speed of 2 MHz.

The gate drivers are connected to the Pmod ports of the FPGA. Of the 6 Pmods, 1 is connected to the CPU (MIO) and 1 serves as analog-digital-converter, so cannot be used. The remaining 4 Pmods each have 12 available pins, of which 4 are used as ground and power pins. The other 8 pins that can be used in differential or single-ended mode. When used in differential mode one can attach $\frac{4\times\frac{8}{2}}{2} = 8$ 2-pin drivers (figure 2.5a) and when used in single-ended mode one can attach $\frac{4\times8}{2} = 16$ 2-pin drivers (figure 2.5b) to the remaining 4 Pmods.



(a) Differential mode.

(b) Single-ended mode.

Figure 2.5: The connection between the Pmod and the drivers.

The speed of the Pmods has to be taken into consideration. This speed depends in part on the clock speed used to control the Pmod pins. This is described in more detail in chapter 3 and chapter 4. However, in our working model the speed of the LED driver is the limiting factor.

2.1.2 Receiver

The receiver also consists of an FPGA, as well as one or more photodiodes, which are connected via an ADC. In order to provide sufficient throughput, the ADC should be connected to an FMC (FPGA Mezzanine Card) slot.

For the FPGA a board was sought that had a LPC (Low Pin Count) FMC slot. This connector provides 68 pins in single-ended mode or 34 pins in differential mode ([34]). Two options were found: a ZedBoard (figure 2.6) at a cost of about 335 euro [31] or a MicroZed (figure 2.7), with a cost price of 180 euro, combined with a Carrier Card (figure 2.8), with a cost price of 125 euro. Both options use a Zynq™-7000 SoC XC7Z020 as PS.

When a throughput of 10 Mbit/s is set to be the target and OOK is initially used as modulation, the transmitter is required to work at a frequency of 10 MHz. In order to correctly sample the signal at the receiver, 20M samples/s are needed, according to Nyquist. Because the signal is a square wave and not a perfect sine wave, a factor 10 is used instead of the Nyquist factor 2, which leads to 100M samples/s. When other modulation techniques, such as Single Edge Position Modulation (SEM) [38], are introduced, this factor 10 is still justifiable.

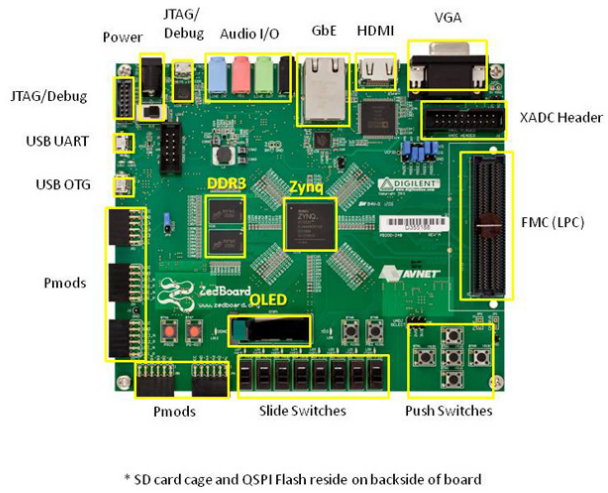


Figure 2.6: Top view of the ZedBoard board with functional overlay, adopted from [35].

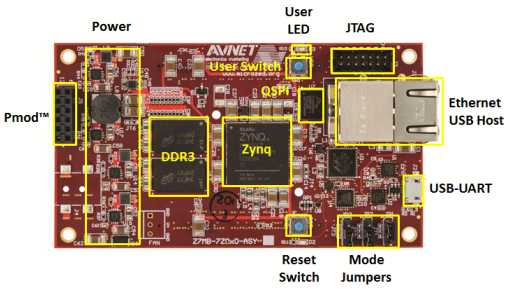


Figure 2.7: Top view of the MicroZed board with functional overlay, adopted from [36].

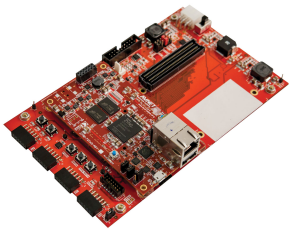


Figure 2.8: Top view of the LPC FMC CarrierCard with a MicroZed board mounted, adopted from [37].

In consequence the ADC should thus be able to process 100 Msps (million samples per second) and each photodiode has to have its own channel on the ADC. Such ADCs are very expensive, because of specifications on the FMC slot. That is why only a generic ADC chip will be used and a custom chip-to-FMC-connection board will be made. This connection can be seen in figure 2.9. However, the construction of the connector falls outside the scope of this thesis.

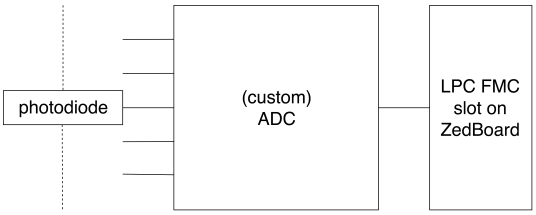


Figure 2.9: The connection between the photodiode, custom ADC and LPC FMC slot on the ZedBoard.

2.2 Software

Regarding software, a distinction can be made between software for the Processing System (PS) and software for the Programmable Logic (PL). The PS software can be Xillinux, Embedded Linux or any other Linux distribution adjusted for embedded systems. The PL software will consist of an adaptation of IEEE 802.15.7.

2.2.1 Xillinux

Xillinux [39] is a Linux distribution which can run on a Zedboard and a ZYBO. Running this operating system without graphics is also possible on a MicroZed. Xillinux provides a demo IP core and the possibility to compose an IP core yourself via a tool on their website. This IP core ensures that the communication between the CPU and FPGA can happen very easily and that the development time of the test platform can be reduced.

First of all, *Getting started with Xillinux for Zynq-7000* [40] was followed to get acquainted with the OS and its installation (building the OS and booting up the ZYBO).

Secondly, *Getting started with Xillybus on a Linux host* [41] was used to learn about memory access in Xillybus by executing, among other things, a "trivial loopback test". Other demo material, described in *Getting started with the FPGA demo bundle for Xilinx* [42] and in *The guide to Xillybus Lite* [43], was also implemented to become familiar with Xillinux. In the latter publication, demo material was available to learn how to access the LEDs, switches and buttons.

Thirdly, *Xillybus host application programming guide for Linux* [44], *Xillybus FPGA designer's guide* [45] and *The guide to defining a custom Xillybus IP core* [46] were read to get more information about best practices in FPGA programming, e.g., synchronous versus asynchronous streams, I/O programming practices, cyclic frame buffers, specific programming techniques, general guidelines (for clocking, data width...), IP cores...

Finally, the demo *FPGA coprocessing for C/C++ programmers (part I-VII)* [47] was carried out and *The guide to Xillybus Block Design Flow for non-HDL users* [48] was read. In this demo and guide, *Vivado High-Level Synthesis (HLS)* was used to compile a function, written in C, into an IP block and subsequently integrated into Xillybus' Block Design flow.

Unfortunately, during this extensive study of Xillinux, it became clear that in the FPGA code, the Pmods are linked to the CPU (MIO). This link makes switching of the Pmod pins happens more slowly than required. Multiple attempts to uncouple the Pmods from the CPU and attaching them to the programmable logic turned out to be unsuccessful, making Xillinux no longer an option in the development of the test platform.

2.2.2 Embedded Linux

Embedded Linux is also a Linux distribution that can be used on a ZYBO (demo [49]) or ZedBoard, just like Xillinux in our initial attempt at finding suitable OS. In this thesis, the following software, which was already available in-house, was used: Buildroot [50] 2016.05 as Linux kernel binary, U-boot [51] to create the bootloader and the Device tree for Zynq-7000 [52]. Details on installing this software on the ZYBO can be found in appendix A.5.

2.2.3 IEEE 802.15.7-2011

For the PL software, first the IEEE 802.15.7-2011 Short-Range Wireless Optical Communication Using Visible Light [5] protocol will be looked at. This protocol already provides a thorough overview of the functionality of a VLC implementation and is therefore a good starting point, when developing an FPGA implementation of said functionalities.

In order to achieve the implementation, it is possible to start from available code used in publications, or from an already in-house available implementation of a similar protocol, IEEE 802.15.4-2015 Low-Rate Wireless Networks [6]. The research on the former approach will be summarized first, but in the end it was the latter approach that was followed in the rest of this work.

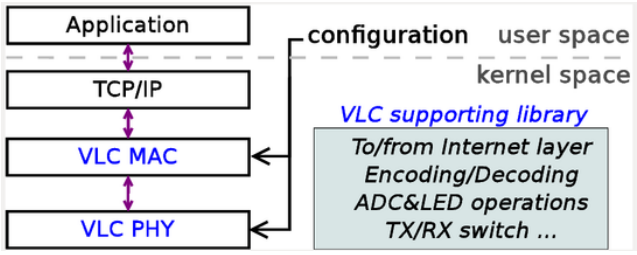


Figure 2.10: Integration of the VLC Linux drivers used in the OpenVLC project, adopted from [53].

First approach

IEEE 802.15.7-2011 implementations are readily available on the internet, but barely in a usable way.

A first example of an implementation is found in the OpenVLC project [54]. Here the PHY and MAC layer are available as Linux drivers, as can be seen in figure 2.10. Because these Linux drivers are written in C and, as can be read in subsection 1.3.1, the achieved throughput of the OpenVLC platform is far below the desired throughput of the platform developed in this work, other starting points were explored.

A second example of an implementation found in publications, is part of the application used in the modBulb project [4, 55], which contains code for the modulator, command, RAM and SPI-slave components (figure 2.11).

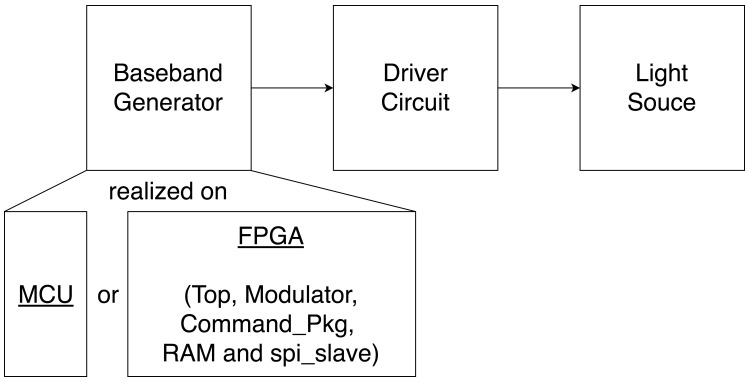


Figure 2.11: "*Abstract architecture. modBulb features a multistage architecture. Different applications can be supported by an appropriate selection of the components.*" (Caption from [4, Figure 2]).

Other examples of FPGA implementation designs can also be found in the literature, including [56] and the ones mentioned in 5.1, but because most of these implementations are not accessible, obtainable or usable for our purpose, another approach is taken.

Second approach

Because the goal was to achieve a (partial) implementation of IEEE 802.15.7, it was recommended, because of the similarities with IEEE 802.15.7, to first look at IEEE 802.15.4, of which an implementation, depicted in figure C.2, was already available in-house. Therefore, in this second approach, a complete description of the available IEEE 802.15.4 implementation will be made, as well as a comparison between IEEE 802.15.4 and IEEE 802.15.7.

From this a list of required changes is constructed in chapter 3, which subsequently are implemented and evaluated in chapter 4.

Because both IEEE 802.15.4-2015 and IEEE 802.15.7-2011 only describe the PHY and MAC layers, going further, the focus will only be on these layers. However, as is mentioned in [11, Chapter 3 - Channel Modeling] and is depicted in figure 2.12, it is worth mentioning that the access from the upper layers to the MAC layer is provided by the LLC layer[1] [57] through the SSCS[2] [58] .

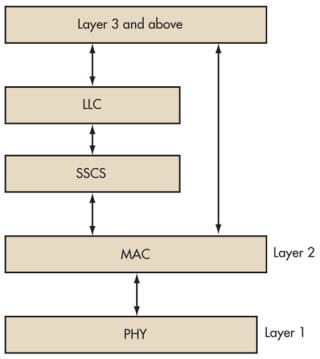


Figure 2.12: Simple architecture of IEEE 802.15.4-2015 and IEEE 802.15.7-2011, adopted from [59].

2.3 Conclusion

The primary goal described in subsection 1.4.2, is the development of a software-defined multiple-LED transceiver. This transceiver needs to be flexible, achieve a high throughput and can be built at a reasonably low cost.

This goal is accomplished by combining a ZYBO as the transmitter and a ZedBoard as the receiver at the hardware level with an adapted IEEE 802.15.4-2015 FPGA implementation to fit the IEEE 802.15.7-2011 protocol, at the software level. The reason to use two different boards is because the receiver has to accommodate a fast ADC and the connection for this ADC was not found on a ZYBO. On the other hand, the ZYBO is cheaper and has as much Pmod connectors as the ZedBoard, which can be used to connect multiple LED drivers. The use of the adapted protocol will save on development time because the apparent similarities between IEEE 802.15.4 and IEEE 802.15.7.

[1]The logical link layer, which is the upper sublayer of the data link layer, provides among others flow control and automatic repeat request (ARQ) [57].

[2]The Service Specific Convergence Sublayer, which is one of the two sublayers of any ATM Adaptation Layer (AAL) and is service dependent, offers assured data transmissions [58].

Chapter 3

Protocols and code

In the first part of this chapter, an overview of the IEEE 802.15.7-2011 protocol will be given. Next, the differences of this protocol with IEEE 802.15.4-2015 will be listed. In a second part, the IEEE 802.15.4 implementation, which in this work will be built upon, will be reviewed.

3.1 IEEE 802.15.7-2011

In this standard [5], the protocol IEEE 802.15.7-2011 is described to be the standard that "*describes the use of VLC for wireless personal area networks (WPAN) and covers topics such as network topologies, addressing, collision avoidance, acknowledgement, performance quality indication, dimming support, visibility support, colored status indication and color-stabilization*".

As mentioned before, this protocol consists of three major parts: the MAC layer, the PHY layer and the device management entity (DME). Furthermore, in this section, these three parts will be briefly introduced. For more detailed information on these three parts, reference [11] can be consulted. To facilitate the reading of these descriptions, figure 3.1 was added.

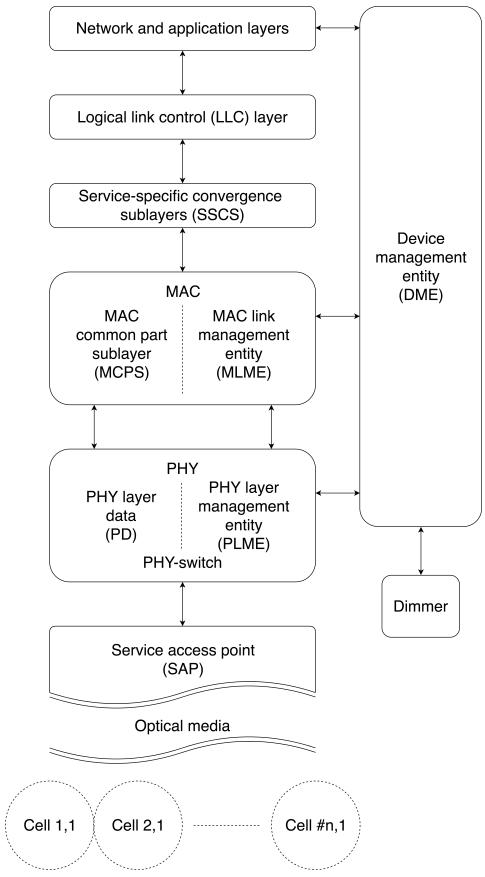


Figure 3.1: VPAN network architecture, adopted from [11].

Modulation	RLL code	Optical clock rate	FEC		Data rate
			Outer code (RS)	Inner code (CC)	
OOK	Manchester	200 kHz	(15,7)	1/4	11.67 kb/s
			(15,11)	1/3	24.44 kb/s
			(15,11)	2/3	48.89 kb/s
			(15,11)	none	73.3 kb/s
			none	none	100 kb/s
VPPM	4B6B	400 kHz	(15,2)	none	35.56 kb/s
			(15,4)	none	71.11 kb/s
			(15,7)	none	124.4 kb/s
			none	none	266.6 kb/s

Table 3.1: PHY I operating modes, adopted from [5].

The MAC layer provides among other things: channel access, guaranteed time slot management, frame validation and acknowledged frame delivery. In IEEE 802.15.7, additional tasks consists of color function[1], color stabilization[2] and dimming support.

The physical layer (PHY) contains the transmitter and receiver functionalities, including: line coding, modulation, error correction coding, and synchronization [11]. In IEEE 802.15.7, three PHY types, PHY I, II and III, are supported. These three types are described in reference [11] and are here summarized as follows: PHY I can be used in outdoor environments at a maximum throughput of 267 kbits/s (table 3.1), PHY II is used indoor at a maximum throughput of 96 Mbit/s (table 3.2) and PHY III, which uses color-shift keying (CSK) and also delivers a maximum throughput of 95 Mbit/s (table 3.3, is designed for VLC systems with multiple light sources and detectors).

The device management entity (DME) adds dimming functionality by providing information to the MAC and PHY layer. On a platform that supports multiple transmitters and/or receivers, the DME controls the PHY switch to select these optical sources and photodetectors [11]. When multiple cells are available, the DME can program the size and position of these cells [11].

[1] "*A function that provides information, such as device status and channel quality, to the human eye via color.*" [5]

[2] "*A control loop for the stabilization of the color emitted by color-shift-keying transmitters.*" [5]

Modulation	RLL code	Optical clock rate	FEC	Data rate
VPPM	4B6B	3.75 MHz	RS(64,32)	1.25 Mb/s
			RS(160,128)	2 Mb/s
		7.5 MHz	RS(64,32)	2.5 Mb/s
			RS(160,128)	4 Mb/s
			none	5 Mb/s
OOK	8B10B	15 MHz	RS(64,32)	6 Mb/s
			RS(160,128)	9.6 Mb/s
		30 MHz	RS(64,32)	12 Mb/s
			RS(160,128)	19.2 Mb/s
		60 MHz	RS(64,32)	24 Mb/s
			RS(160,128)	38.4 Mb/s
		120 MHz	RS(64,32)	48 Mb/s
			RS(160,128)	76.8 Mb/s
			none	96 Mb/s

Table 3.2: PHY II operating modes, adopted from [5].

Modulation	Optical clock rate	FEC	Data rate
4-CSK	12 MHz	RS(64,32)	12 Mb/s
8-CSK		RS(64,32)	18 Mb/s
4-CSK	24 MHz	RS(64,32)	24 Mb/s
8-CSK		RS(64,32)	36 Mb/s
16-CSK		RS(64,32)	48 Mb/s
8-CSK		none	72 Mb/s
16-CSK		none	96 Mb/s

Table 3.3: PHY III operating modes, adopted from [5].

3.2 IEEE 802.15.4-2015

As described in the introduction of this standard [6], it is the third installment of the IEEE 802.15.4 protocol. The goal of this protocol is to enable very low-cost and low-power communications.

In 2003, two PHY layers, which operate in different frequency bands, and a simple MAC layer were defined.
In 2006, two more PHY options besides the enhancements of the MAC layer were introduced, which include: "*improved MAC layer security*", "*MAC frames with an increased version number*", "*support for beacon scheduling [and][...] a shared time base with a data time stamping mechanism*", and "*synchronization of broadcast messages in beacon-enabled personal area networks (PANs)*".
A second revision of this protocol happened in 2011, which added MAC capability to support ranging and four more PHY options.

In the latest version, introduced in 2015, the standard was restructured, six more PHY were added and one MAC amendment was made. In this restructuring, corrigenda and clarifications were also added. Examples of these new features are: enhanced frame formats, low-energy mechanisms, a securable acknowledgment frame that can carry data and an array of new PHY modulation, coding, and band options to support new applications [6].

More amendments were added later on: "Amendment 2: Ultra-Low Power Physical Layer" (2016) [60], "Amendment 4: Higher Rate (2 Mb/s) Physical (PHY) Layer" (2017) [61] and "Amendment 6: Enabling Spectrum Resource Measurement Capability" (2018) [62].

3.3 Differences between the two protocols

After examining these two protocols in more detail, it became clear that there are more differences between the two protocols than initially thought. These differences are listed beneath and are summarized in table 3.4.

		IEEE 802.15.4	IEEE 802.15.7
PHY		O-QPSK, BPSK, ASK, CSS, HRP and LRP UWB, GFSK, MSK, SUN FSK, SUN OFDM, SUN O-QPSK, LECIM DSSS, LECIM FSK, TVWS-FSK, TVWS-OFDM, TVWS-NB-OFDM, RCC LMR and RCC DSSS BPSK	OOK (PHY I and II), VPPM (PHY I and II) and CSK (PHY III)
MAC	channel access	CSMA-CA, TSCH CCA, TSCH CSMA-CA, CSMA-CA with PCA and LECIM ALOHA PCA	slotted random access, with or without CCA visibility support
	starting and maintaining (V)PANs	suport for ranging, DSME, TVWS, CTM and LE transmission, reception and acknowledgement	support for fast link recovery, VLC cell design and mobility, color function and visibility plus dimming
	MAC frame formats and commands	among others differences in the frame control field, the general MAC frame format, the beacon frame format, the data frame format and the command frame format	
			additional command frames for the blinking notification, the dimming notification, the fast link recovery, the mobility notification, the clock rate change notification, the multiple channel assignment, the color stabilization timer notification, the color stabilization information and the CVD disable commands
DME		/	support for dimming, flicker mitigation and clock-rate selection

Table 3.4: Overview of the differences between IEEE 802.15.4 and IEEE 802.15.7.

Firstly, in contrast to IEEE 802.15.7, which supports peer-to-peer, star and broadcast topologies (figure 3.2), IEEE 802.15.4 only supports the first two topologies.

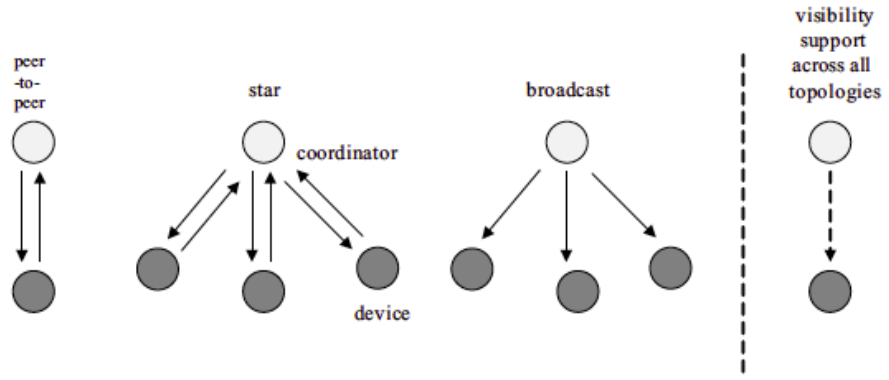


Figure 3.2: Supported MAC topologies, adopted from [5].

3.3.1 Differences on PHY layer level

On the PHY layer level, IEEE 802.15.4 contains specifications for many PHY: O-QPSK, BPSK, ASK, CSS, HRP and LRP UWB, GFSK, MSK, SUN FSK, SUN OFDM, SUN O-QPSK, LECIM DSSS, LECIM FSK, TVWS-FSK, TVWS-OFDM, TVWS-NB-OFDM, RCC LMR and RCC DSSS BPSK.

In contrast, IEEE 802.15.7 contains only specifications for PHY I, II and III. PHY I (table 3.1) uses OOK at an optical clock rate of 200 kHz or VPPM at 400 kHz. It delivers data rates from 11.67 kbit/s to 100 kbit/s for OOK and from 35.56 kbit/s to 266.6 kbit/s for VPPM. PHY II (table 3.2) uses VPPM at 3.75 or 7.5 MHz (data rates from 1.25 Mbit/s to 5 Mbit/s) or OOK at 15, 30, 60 or 120 MHz (data rates from 6 Mbit/s to 96 Mbit/s). PHY III (table 3.3) uses CSK (4-, 8- or 16-CSK) at optical clock rates of 12 MHz or 24 MHz, which lead to data rates from 12 Mbit/s to 96Mbit/s.

3.3.2 Differences on MAC layer level

On MAC layer level, most differences can be found in: channel access, starting and maintaining (V)PANs, the MAC frame formats plus MAC commands, IEs and MAC services.

While channel access in IEEE 802.15.4 contains extra random access methods including: CSMA-CA, TSCH CCA, TSCH CSMA-CA, CSMA-CA with PCA and LECIM ALOHA PCA; IEEE 802.15.7 uses slotted random access, with or without CCA and adds visibility support.

For starting and maintaining (V)PANs, IEEE 802.15.4 contains among others support for ranging (finding the distance between two points), DSME (offers a solution to beacon collision problems), TVWS, CTM and LE transmission, reception and acknowledgement. On the other hand IEEE 802.15.7 contains fast link recovery, VLC cell design and mobility support, color function support (CVD and color stabilization) and visibility plus dimming support.

Other differences between the protocols are the MAC frame formats and commands. In appendix B, comparisons for the frame control field (table B.1), the general MAC frame format (table B.2), the beacon frame format (table B.3), the data frame format (table B.4) and the command frame format (table B.5) are listed. Appendix B also contains a comparison between the MAC commands used in IEEE 802.15.4 and IEEE 802.15.7 (table B.6). In addition, IEEE 802.15.7 contains: command frames for the blinking notification, the dimming notification, the fast link recovery, the mobility notification, the clock rate change notification, the multiple channel assignment, the color stabilization timer notification, the color stabilization information and the CVD disable commands.

The possible information elements also differ between IEEE 802.15.4 and IEEE 802.15.7. A comprehensive and detailed listing of all the differences is deemed unnecessary, but it can be noted that IEEE 802.15.7 adds information fields about dimming, color stabilization and PHY I, II and III. The same holds for the MAC services, where in IEEE 802.15.7 optical-clock-rate selection and in IEEE 802.15.4 services for ranging, TSCH, DSME and others were added.

3.3.3 Other differences

Lastly, as mentioned above, the device management entity (DME) is also part of IEEE 802.15.7. This entity provides especially dimming and flicker mitigation support, as well as support for clock-rate selection, to the MAC and PHY layer.

3.4 Description of the IEEE 802.15.4 implementation

This IEEE 802.15.4 implementation was available in-house and will be built upon in this work. The block diagram of the FPGA design (figure C.2) consists of four main blocks. Two of them, the ZYNQ7 Processing System (processing_system7_0) block and the Processor System Reset (proc_sys_reset_0) block, are readily available blocks in Vivado. The AXI Interconnect (ps7_0_axi_periph) block, which provides the connections between the PS and the PL, and the ieee802154_v1_0 (ieee802154) block are designed by Bertold Van den Bergh.

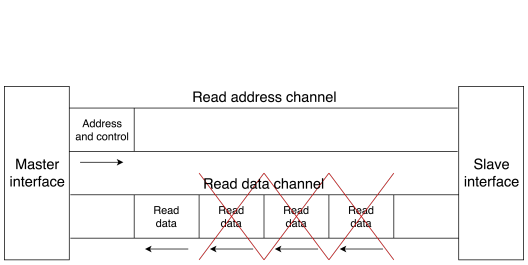
In the next sections, the AXI Interconnect will briefly be discussed and secondly, the IEEE802154 block will be reviewed in detail. In addition to these descriptions, it can also be mentioned that the CPU clock works at 667 MHz, the PL fabric clock (FCLK_CLK0) at 50 MHz and that the reset is an active-high reset signal.

3.4.1 Interface between CPU and programmable logic

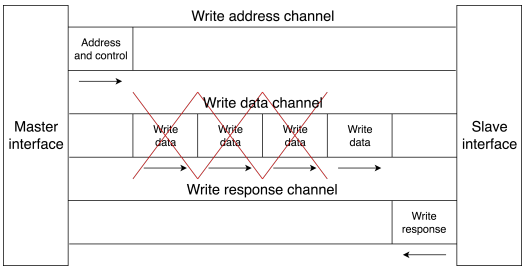
The AXI Interconnect block provides an AXI bus, which is used to connect the PS with the PL by the ZYBO. It is the connection that allows the PL to appear as memory that a program on the PS can read from and write to. In the AXI reference guide [63], three types of interfaces are listed: AXI4, for high-performance memory-mapped requirements, AXI4-Lite, for simple, low-throughput memory-mapped communication (e.g. to and from control and status registers) and AXI4-Stream, for high-speed streaming data. In this implementation, AXI4-Lite is used. A complete block overview can be found in figure C.3.

AXI4-Lite uses five channels: the read address channel, the read data channel, the write address channel, the write data channel and the write response channel. Unlike AXI4, which has a limit of 256 data transfers per burst transaction, AXI4-Lite allows only 1 data transfer per transaction [64, 63], which is adjusted for in figure 3.3. Even though AXI4 has 32 and 64-bit support, 32-bit width is recommended [64].

In this implementation, AXI4-Lite is simplified "*into a simple Address, Data, ReadStrobe, WriteStrobe bus*", which can be found in axi_slave.vhd.



(a) Channel architecture of reads, adopted from [63] and [64].



(b) Channel architecture of writes, adopted from [63] and [64].

Figure 3.3: Channel architectures show how a Read and Write transaction uses the different channels. The red crosses depict the fact that AXI4-Lite only allows 1 data transfer per transaction.

Pmod JC (Hi-Speed)

PWR	12	06	PWR
GND	11	05	GND
frontend_dio4 (IN; 3.3 V)	10 (U12)	04 (T10)	frontend_dio5 (IN; 3.3 V)
frontend_dio1 (INOUT; 3.3 V)	09 (T12)	03 (T11)	frontend_dio2 (IN; 3.3 V)
frontend_miso (IN; 3.3 V)	08 (Y14)	02 (W15)	frontend_sck (OUT; 3.3 V)
frontend_mosi (OUT; 3.3 V)	07 (W14)	01 (V15)	frontend_mosi (OUT; 3.3 V)

Figure 3.4: Pmod JC with pin names and connections to the ieee802154 block.

3.4.2 ieee802154 block and code

In figures C.2 and C.4, the different connections to and from the ieee802154 block are depicted.
The AXI connections, along with the AXI clock and AXI reset are represented. In addition to these connections, an interrupt (intOut) is present.
The remaining connections are those to Pmod JC, which are summarized in figure 3.4.

Below, all the files that make up the ieee802154 block are listed, along with a brief discussion of their function. For clarity, a block diagram (figure 3.5), which visualizes the connections between the different files, has been added.

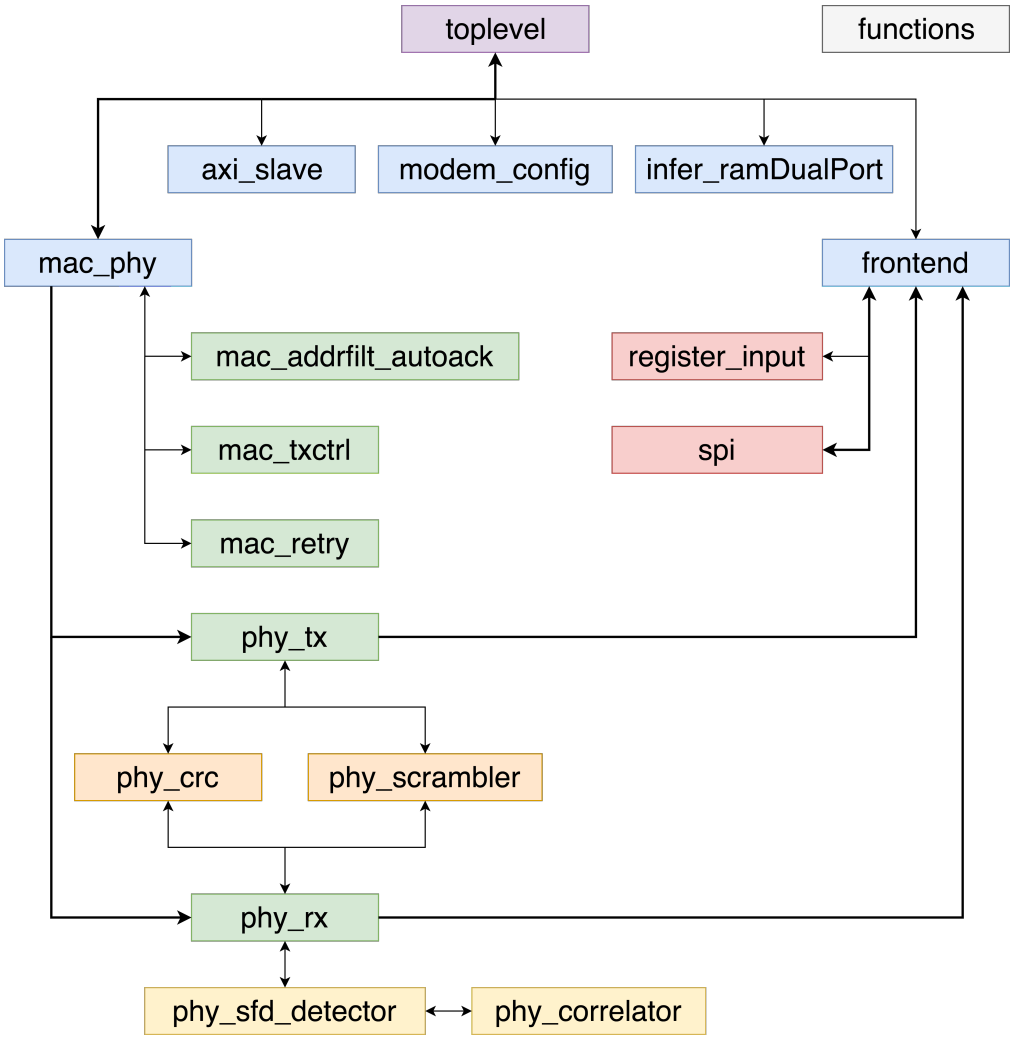


Figure 3.5: Simplified block diagram of the IEEE 802.15.4 code, which is used to visualize the connections between the different files.

toplevel.vhd The toplevel contains the different ports for the global signals (mentioned in figure 3.4), AXI channels (write address channel, write data channel, write response channel, read address channel and read data channel), active high interrupt and the physical pins. The description of the signals and some entities are included as well, in addition to the procedure for the CPU and the decoding of the modem settings from the registers.

axi_slave.vhd The AXI connection consists of two parts: a master and a slave. The master runs on the PS and the slave on the PL. In this file the following was given as commentary: "This AXI slave is not very efficient. Most transactions take more cycles than required. However, the bandwidth is so low that this doesn't matter at all." This will have consequences when a data stream from the PS to the PL will be established.

modem_config.vhd Packages the modem configuration settings (eg., MAC backoff settings, Auto TX and RX settings, use of carrier sense, MAC transmissions...)

infer_ramDualPort.vhd TX and RX packet buffers.

frontend.vhd Controls the SPI link and the RSSI (Received Signal Strength Indicator) measurement. Additionally, two clocks are generated here: a 100 kHz clock and a 1 MHz clock.

register_input.vhd

spi.vhd This block makes use of the following physical pins: frontend_cs, frontend_sclk, frontend_mosi and frontend_miso.

mac_phy.vhd "This module puts the MAC above the PHY." It combines the transmitter (phy_tx.vhd), the receiver (phy_rx.vhd), the packet filter logic (mac_addrfilt_autoack.vhd), CSMA and automatic ACK (mac_txctrl.vhd), the retry logic (mac_retry.vhd) and output packets to higher layers.

mac_addrfilt_autoack.vhd "This block analyzes the received packets. It does address filtering and investigates whether an ACK needs to be sent. This block can in some cases accept invalid frames to save FPGA resources. This is considered acceptable since the higher layer software does a more thorough check."

mac_txctrl.vhd CSMA and automatic ACK.

mac_retry.vhd "This module will trigger the transmission of a packet, and retry it a number of times if it failed."

phy_tx.vhd This block creates the bits, which are send by the frontend block.

phy_crc.vhd This block calculates a cyclic redundancy check (CRC) over the physical layer service data unit (PSDU). This way errors can be detected in the PSDU.

phy_scrambler.vhd This block scrambles or descrambles everything after the start frame delimiter (SFD).

phy_rx.vhd This block processes the received bits, which come from the frontend block.

phy_sfd_detector.vhd This block uses correlators to find the SFD. Firstly, a correlator (phy_correlator.vhd) for every bit in the SFD is initiated. Subsequently, the results of the correlators are added together.

phy_correlator.vhd

functions.vhd The following functions, which are used in all other files, are provided: XOR_REDUCE and TO_STD_LOGIC.

Additionally, there are four testbed files: **mac_tb.vhd**, **phy_tb.vhd**, **phy_tx_tb.vhd** and **spi_tb**.

3.5 Conclusion

In section 3.3, it became clear that there are profoundly more differences between IEEE 802.15.4-2015 and IEEE 802.15.7-2011 than initially thought. First of all, IEEE 802.15.7-2011 has an extra part, namely the device manangement entity. Also, the PHY part differs completely and in the MAC layer part are many more little differences.

Next, the IEEE 802.15.4 implementation was looked at. Main parts were the AXI Interconnect block, which provides the connection between the PS (CPU) and the PL, and the ieee802154 block.

Finally, the IEEE 802.15.4 code was explored, which was reasonably easy to understand on a functional level, but much more difficult on a code level. It became clear that this code was a light version of an IEEE 802.15.4-2015 protocol implementation.

After this profound analysis, it was concluded that it would be easier to start from scratch and reuse some of the IEEE 802.15.4 code, than trying to change this available code to fit IEEE 802.15.7. In the next chapter, the newly developed code is described, along with what worked and what failed.

Chapter 4

Implementation and Evaluation

In this chapter, the general strategy for the testing of the platform will be outlined and then, the individual tests, performed on the platform, will be reviewed.

4.1 General outline of the trials

In this chapter, the focus is on the PHY. The MAC layer will be omitted from the platform in first instance.

Because a throughput of 10 Mbit/s is targeted and looking at table 3.2, it seemed best to focus on OOK with 8b/10b ([65]), to generate a RLL line code, and a minimum clock speed of 30 MHz. Another possibility to generate the targeted throughput, was 4-, 8- or 16-CSK at a minimum clock speed of 12 MHz. However, this solution is more complicated. Therefore this solution took second place in order of execution. A clock signal (FCLK_CLK0) of 50 MHz was chosen, because this could be adopted from the IEEE 802.15.4 implementation, available in-house.

Also, a solution had to be provided to toggle between the different operating modes, which are reviewed in section 2.1.1. A third mode, everything off, was added, besides the already existing *illumination mode* and the *illumination + communication mode*. Two output Pmod pins, JC1 (V15) plus JC2 (W15), and two input Pmod pins, JC7 (W14) plus JC8 (Y14), were provided (see figure C.5). JC1 had the illumination signal and JC2 had the communication signal to the LED driver. The output pins were not yet connected internally, but were provided for future extendability or testing: a loop could be made from JC1 to JC7 and from JC2 to JC8.

In a first approach, the switches on the ZYBO were used to toggle between output and input modes. In a second approach, this functionality was performed with the AXI interface. The switches (see figure 2.1) worked as follows:

- SW0 off and SW1 off: everything off;
- SW0 on and SW1 off: illumination mode;
- SW0 on and SW1 on: illumination + communication mode.

It can be noted that it would be better in terms of timing to only switch one switch to get in *illumination + communication mode*. Here, however, was chosen for switching two switches, because before the switch to *illumination + communication mode* is made, the system is most of the time already in *illumination mode*. Moreover, this choice was also made with the AXI connection in mind: sending a '3' instead of a '4' seemed more pleasant.

The AXI interface, adopted from the IEEE 802.15.4 implementation, consisted of a memory mapped peripheral that presented two 32-bit registers to the host CPU: a read/write register and a read-only register with a 4-bit value [64]. These 4 bits originated from the fact that there were 4 switches/leds, thus every switch/led had a bit and had the ability to be turned on (= 1) or off (= 0) individually. Only the read/write register was used here and was accessed with XSDB (subsection A.5.2) as follows:
read: `mrd -force 0x60000000 1`;
write: `mwr -force 0x60000000 0xY` with **Y** a number from 0 to 31:

- everything off: **Y**= 0;
- illumination mode= **Y**= 1;
- illumination + communication mode: **Y**= 3.

As was mentioned in subsection 3.4.2, it was observed that, in this work, the implementation of the AXI interface was not fast and large enough to support data streams from the PS to the PL and back. Therefore, it can only be used to change settings. In future work, this AXI4-Lite interface should be extended or replaced with a AXI4 or AXI4-stream interface to allow data streams from higher layers on the PS to the PL.

4.2 Discussion of the trials

In this section, each executed test, where a new version number was used and the code was placed in a separate folder, will be listed and reviewed. Every test folder contains the following files: `axi_slave`, `clkdiv`, `functions`, `phy_tx`, `register_bus`, `register_input` and `toplevel`. `axi_slave`, `clkdiv`, `functions`, `register_bus` and `register_input` are adopted from the IEEE 802.15.4 implementation, because the same functionality could be used in this IEEE 802.15.7 implementation. Hence, changes were only made in `toplevel` and `phy_tx`. In the additional file `myCLK`, which is mentioned below and only used in certain trials, code from `nandland.com` [66] is adapted to fit the needs in this IEEE 802.15.7 implementation.

The detailed explanation on how to compile the code can be found in section A.3. The functionality and integrity of the system was checked with an oscilloscope. This device was used to read out the signals on Pmod pins JC1 and JC2, as shown in figure 4.1.

```
testProc: process(s00_axi_aclk)
begin
    if(rising_edge(s00_axi_aclk)) then
        if(switches_sync(0) = '0' and switches_sync(1) = '0' and
        ↪  switches_sync(2) = '0' and switches_sync(3) = '0')
        ↪  then
            do_what <= OFF;
            axiRegister <= "0000";
            light_out_0 <= '0';
            signal_out_0 <= '0';
        elsif(switches_sync(0) = '1' and switches_sync(1) = '0'
        ↪  and switches_sync(2) = '0' and switches_sync(3) =
        ↪  '0') then
            do_what <= LIGHT;
            axiRegister <= "0001";
            light_out_0 <= '1';
            signal_out_0 <= '0';
        elsif(switches_sync(0) = '1' and switches_sync(1) = '1'
        ↪  and switches_sync(2) = '0' and switches_sync(3) =
        ↪  '0') then
            do_what <= SEND;
            axiRegister <= "0011";
            light_out_0 <= '0';
            signal_out_0 <= '1';
        end if;
    end if;
end process;

leds <= axiRegister;
```

Code block 1: Main code in `toplevel` for Trial 1.

```
txProc: entity work.phy_tx port map (clk => s00_axi_aclk,
                                     reset => s00_axi_aresetn,
                                     secondPulse => secondPulse,
                                     light_out => light_out,
                                     signal_out => signal_out,
                                     switches => switches_sync,
                                     leds => leds);
```

Code block 2: Main code in `toplevel` for Trial 2.

4.2.1 Trial 1

Focus: Assigning signals to ports, depending on the position of the switches.
Project folder: ieee802157_test3; part of the code is listed in code block 1.
Design decisions: The AXI interface was not used, the functionality is rather simple (just a constant signal on the output, no alternating signal) and all the own code is put into `toplevel`.

- variable *do_what* was used to save the state of a the final state machine (FSM), used in other trials and variable *axiRegister* is used to put a signal on leds (the name is legacy (from the AXI interface code));
- *light_out_0* was connected to the *light_out* port, which was connected to Pmod pin JC0 and *signal_out_0* is connected to the *signal_out* port, which was connected to Pmod pin JC1;

Status: Working.

4.2.2 Trial 2

Focus: Splitting `toplevel` to `phy_tx`.
Project folder: ieee802157_test4; part of the code is listed in code block 2 and 3.
Design decisions: By splitting the `toplevel`, more structure was created and in later adaptations, the code can be adjusted or extended more easily.
Status: Working.

4.2.3 Trial 3

Focus: The AXI interface.
Project folder: ieee802157_test6; same code as in Trial 2, with the following changes:

- In code block 2, `switches => switches_sync,` was changed to `switches => axiRegisters);` and the line `leds => leds);` was removed.
- In code block 3, the assignments to leds were removed and in `toplevel` `leds <= axiRegisters;` was added.

Design decisions: In a real life scenario, commands to change settings will come from the CPU and not from the switches. Therefore the changes enabled the mode to be changed according to the description in section 4.1.
Status: Working.

4.2.4 Trial 4

Focus: Splitting `phy_tx`.
Project folder: ieee802157_test5 (commented); part of the code is listed in code block 4 and 5.

Design decisions (first try): To add more structure and make the code easier to adjust or extend, the `phy_tx` code from Trial 2 (code block 3) was split. This split allows the assignment to `signal_out` to happen elsewhere. The port `signal_out` in code block 3 was changed to signal `toSignalOut` and the last assignment was changed to `startSending <= '1';`.
Status (first try): Not working. The code was compiled and a bitfile was generated, but the signal on Pmod JC2 was not correct. Later, it was found that the problem was, most likely, attributable to `clkdiv`.
Design decisions (second try): In code block 5, the assignment of '1' to `signal_out` only happens when `startSending` was '1'.
Status (second try): Working.

4.2.5 Trial 5

Focus: Using an FSM.
Project folder: ieee802157_test2 (commented); part of the code is listed in code block 2, 3 and 8.
Design decisions (first try): Working further with Trial 2, code block 3 was changed to the code in code block 6. Additionally, an FSM (code block 7) was tried because it facilitates the readability of the code and makes it easier to add other modes.
Status (first try): Not working. The code was compiled and a bitfile was generated, but the signal on Pmod JC2 was not correct.
Design decisions (second try): In this trial, code block 7 was replaced by code block 8.
Status (second try): Working.

4.2.6 Trial 6

Focus: Changing the clock frequency and creating an alternating signal using a multiplexer (MUX).
Project folder: ieee802157_test7; the important part of the code is listed in code block 10 and 9.
Design decisions: Firstly, it was noticed that `clkdiv` did not work, so `myCLK` was created. The code in `myCLK`, which was partly adopted from a website [66], did not seem to work neither, but when placed in `toplevel`, it did. An example to create a 2 MHz clock frequency from the 50 MHz FPGA clock can be found in code block 10. The multiplexer used for the clock can be found in code block 9. This multiplexer uses the position of the switch as follows:

- every switch on '0': 50 MHz
- SW0 on '1': 10 MHz
- SW1 on '1': 2 MHz
- SW2 on '1': 1 MHz
- SW3 on '1': 1 Hz

The new clock signal is subsequently used to create an alternating signal in `phy_tx`.

Note: It is observed that the 50 MHz clock was too fast to be assigned on the Pmod pins.
Status: Working.

```
testProc: process(clk)
begin
    if(rising_edge(clk)) then
            if(switches(0) = '0' and switches(1) = '0' and
            ↪  switches(2) = '0' and switches(3) = '0') then
                light_out <= '0';
                signal_out <= '0';
                leds <= "0000";
            elsif(switches(0) = '1' and switches(1) = '0' and
            ↪  switches(2) = '0' and switches(3) = '0') then
                light_out <= '1';
                signal_out <= '0';
                leds <= "0001";
            elsif(switches(0) = '1' and switches(1) = '1' and
            ↪  switches(2) = '0' and switches(3) = '0') then
                light_out <= '0';
                signal_out <= '1';
                leds <= "0011";
            end if;
    end if;
end process;
```

Code block 3: Main code in `phy_tx` for Trial 2.

```
send_something: process(clk)
begin
    if(rising_edge(clk) and startSending = '1') then
        toSignalOut <= '1';
        sendDone <= '1';
        startSending <= '0';
    end if;
end process;
signal_out <= toSignalOut;
```

Code block 4: Main code in `phy_tx` for Trial 4 (first try).

```
testProc: process(clk)
begin
    if(rising_edge(clk)) then
        if(switches(0) = '0' and switches(1) = '0' and
        ↪  switches(2) = '0' and switches(3) = '0') then
            light_out <= '0';
            startSending <= '0';
            leds <= "0000";
        elsif(switches(0) = '1' and switches(1) = '0' and
        ↪  switches(2) = '0' and switches(3) = '0') then
            light_out <= '1';
            startSending <= '0';
            leds <= "0001";
        elsif(switches(0) = '1' and switches(1) = '1' and
        ↪  switches(2) = '0' and switches(3) = '0') then
            light_out <= '0';
            startSending <= '1';
            leds <= "0011";
        end if;
    end if;
end process;

signal_out <= '1' and startSending;
```

Code block 5: Main code in `phy_tx` for Trial 4 (second try).

```
testProc: process(clk)
begin
    if(rising_edge(clk)) then
        if(switches(0) = '0' and switches(1) = '0' and
        ↪  switches(2) = '0' and switches(3) = '0') then
            do_what <= OFF;
        elsif(switches(0) = '1' and switches(1) = '0' and
        ↪  switches(2) = '0' and switches(3) = '0') then
            do_what <= LIGHT;
        elsif(switches(0) = '1' and switches(1) = '1' and
        ↪  switches(2) = '0' and switches(3) = '0') then
            do_what <= SEND;
        end if;
    end if;
end process;
```

Code block 6: Part of the main code in `phy_tx` for Trial 5 (first try).

```
    fsmProc: process(clk)
    begin
        if(rising_edge(clk)) then
            if(reset = '1') then
                do_what <= OFF;
                sendDone <= '1';
            else
                case do_what is
                    when OFF =>
                        if(sendDone = '0') then
                            do_what <= SEND;
                            axiRegister <= "1011";
                        else
                            light_out <= '0';
                            axiRegister <= "0000";
                            report "System is OFF";
                        end if;
                    when LIGHT =>
                        if(sendDone = '0') then
                            do_what <= SEND;
                            axiRegister <= "1011";
                        else
                            light_out <= '1';
                            axiRegister <= "0001";
                            report "Only light";
                        end if;
                    when SEND =>
                        sendDone <= '0';
                        startSending <= '1';
                        axiRegister <= "0011";
                        report "Send something";
                end case;
            end if;
        end if;
    end process;

    leds <= axiRegister;
```

Code block 7: The FSM in `phy_tx` for Trial 5 (firsty try).

```
    leds <= "0000" when (do_what = OFF) else
            "0001" when (do_what = LIGHT) else
            "0011" when (do_what = SEND);
    light_out <= '1' and TO_STD_LOGIC(do_what = LIGHT);
    signal_out <= '1' and TO_STD_LOGIC(do_what = SEND);
```

Code block 8: Part of the main code in `phy_tx` for Trial 5 (second try). `TO_STD_LOGIC` is a function from `functions` that takes a `boolean` type and returns a `std_logic` type.

```
newCLK <= r_toggle50MHz when switches =
↪  std_logic_vector(to_unsigned(0,switches'length)) else
        r_toggle10MHz when switches =
        ↪  std_logic_vector(to_unsigned(1,switches'length)) else
    r_toggle2MHz when switches =
    ↪  std_logic_vector(to_unsigned(2,switches'length)) else
    r_toggle1MHz when switches =
    ↪  std_logic_vector(to_unsigned(4,switches'length)) else
    r_toggle1Hz;
```

Code block 9: The multiplexer used in `toplevel` for Trial 6.

```
        fpgaClock : positive := 50000000;
        constant c_2MHz : natural := fpgaClock / 2000000 / 2;
        signal r_cnt2MHz : natural range 0 to c_2MHz;
        signal r_toggle2MHz : std_logic := '0';

        clk2MHzProc: process(s00_axi_aclk) is
        begin
                if(rising_edge(s00_axi_aclk)) then
                            if(r_cnt2MHz = c_2MHZ-1) then
                                 r_toggle2MHz <= not r_toggle2MHz;
                                 r_cnt2MHz <= 0;
                        else
                            r_cnt2MHz <= r_cnt2MHz + 1;
                     end if;
                    end if;
        end process clk2MHzProc;
```

Code block 10: Code used to create a 2 MHz clock from a 50 MHz clock frequency in `toplevel` for Trial 6.

4.2.7 Trial 7

Focus: Using a hardcoded string of bits as message.
Project folder: ieee802157_test8; part of the code is listed in code block 11.
Design decisions: It was chosen to convert a bit_vector (which would be the message send from a higher layer) and convert this to a `std_logic_vector`. Subsequently, an iteration was carried out over the length of the `std_logic_vector`, starting from the smallest index.
Status: Working.

```
signalProc: process(clk,newCLK)
    variable bitsToSend: bit_vector(21 downto 0) :=
    ↪ "10101100111000110010 10";
    variable bitsToSendVector: std_logic_vector(bitsToSend'length-1
    ↪ downto 0) := to_stdlogicvector(bitsToSend);
    variable counter: integer := bitsToSendVector'length;
begin
    if(rising_edge(newCLK)) then
        if(counter = 0) then
            counter := bitsToSendVector'length;
        else
            toSignalOut <= bitsToSendVector(bitsToSendVector'length
            ↪ - counter);
            counter := counter - 1;
        end if;
    end if;
end process signalProc;

signal_out <= toSignalOut and TO_STD_LOGIC(do_what = SEND);
```

Code block 11: Part of the main code in `phy_tx` for Trial 7.

4.2.8 Trial 8

Focus: Adding ports for 16 LED drivers.
Project folder: ieee802157_test8_ext; an overview of the block diagram can be found in figure C.6.
Design decisions: It was chosen to connect the LED drivers in single-ended mode (see figure 2.5b). Each LED driver has a toggle to switch the driver on or off. The decision was taken to hardcode those toggles on '1', but in the future, they could be changed via the AXI interface.
Note: Also, a test bench was created, but it did not work.
Status: Working.

4.3 Conclusion

In this chapter, several trials were created and executed with successes and some fails.

The correct execution of the code was validated with an oscilloscope. It was noticed that, contrary to sequential programming languages, including Java or Python, programming in VHDL, in which execution happens in parallel, entails more challenges. For example, code was compiled correctly, but the desired result was not achieved due to erroneous reasoning during programming.

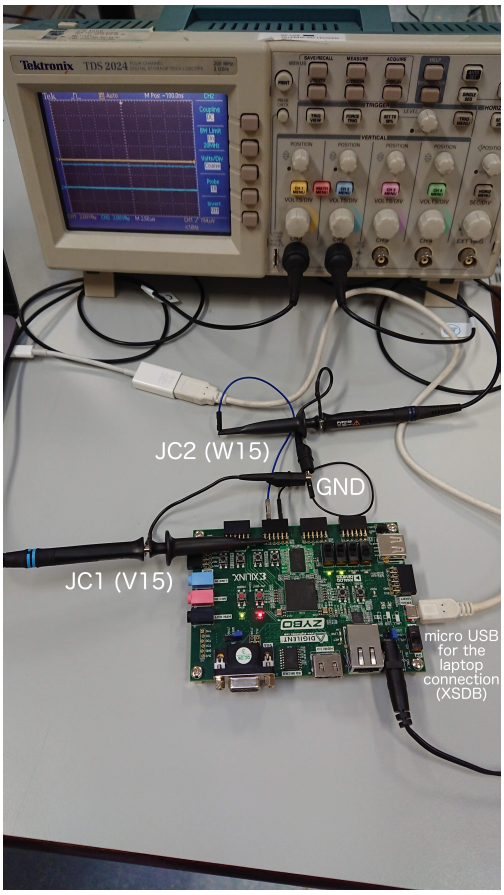


Figure 4.1: Picture of the test setting.

At the beginning of the project, a constant signal was put on Pmod pins. Subsequently, the switches and an AXI interface were used to change the mode of the system. At this juncture, LEDs were used as a visual control mechanism to see in which mode the system is working. Furthermore, it was possible to adjust the clock frequency. This change was verified with an alternating signal, which was put on a Pmod pin. In a final implementation, first, all the functionality of the previous implementations was put together. Second, the number of output pins was extended to accommodate the use of multiple LED drivers and hardcoded toggles were added to turn specific LED drivers on or off. Finally, a hardcoded bitstring was “sent”.

Furthermore, a test bench was created to make it possible to check the correct execution of the code when there is no oscilloscope available, but it did not work.

Chapter 5

Future work

The accomplishments, described in this thesis, do not finalize the implementation of the VLC test system. Nevertheless, these accomplishments are the basis for future development work to be taken up in consecutive theses. Therefore, this chapter will list recommended next steps to finish the implementation, started in this work. However, these next steps will largely benefit from the recommended reading presented in the first section.

5.1 Additional reading material

During the research for this work, interesting literature emerged that was not examined or covered in this work. This literature will be briefly listed here.

First of all, in this thesis, references were made to the book "*Visible Light Communications - Theory and Applications*" [11]. This book contains chapters which could be useful as background information for future work, including: "*Chapter 3: Channel Modeling*", "*Chapter 4: Modulation Schemes*", "*Chapter 5: IEEE 802.15.7: Visible Light Communication Standard*", "*Chapter 6: Techniques for Enhancing the Performance of VLC Systems*" and "*Chapter 14: OFDM-Based VLC Systems FPGA Prototyping*".

In addition to this book, new publications appear every day. Out of this vast amount of new information, the following publications are suggested to be beneficial:

- "*Design and implementation of IEEE 802.15.7 VLC PHY-I transceiver*" (2014) [67]
- "*A Fully Integrated IEEE 802.15.7 Visible Light Communication Transmitter With On-Chip 8-W 85% Efficiency Boost LED Driver*" (2016) [68]
- "*Single Edge Position Modulation as a Dimming Technique for Visible Light Communications*" (2016) [38]
- "*In Light and In Darkness, In Motion and In Stillness: A Reliable and Adaptive Receiver for the Internet of Lights*" (2018) [69]

5.2 Expanding the functionality of the current IEEE 802.15.7 implementation

In the current format, the VLC test system is not yet usable for experimental measurements. However, when adapted to gain more flexibility for research purposes, it will be. Therefore, some functionality needs to be added. This additional functionality is described in the subsections below.

5.2.1 Adding more flexibility

The current implementation can only accommodate the use of all the LED drivers simultaneously, so here is room for improvement in flexibility.

A more flexible controlling of the toggles for the LED drivers would largely enhance the suppleness of the test system. Moreover, an approach to generate the ability to control what can be sent by each LED driver individually, would bring the flexibility of the VLC test system to another level.

Another addition to increase the flexibility of the VLC test system, could be the supplementary use of a RLL line code on top of the OOK modulation. Also, other modulations could be added in the PHY to increase the agility of the test system.

5.2.2 Starting from on the SD card

In the current status of the VLC system, the OS is loaded via XSDB. This OS should be loaded from the SD card to facilitate testing with the platform. In this way, it will also be easier to connect with the platform and experiment with C code, running on the CPU, which connects to the IEEE 802.15.7 implementation on the PL via the AXI interface.

Also, a GUI could be written and run on the host system, which is connected (via ethernet or Wi-Fi) with the platform. This GUI could be used to change settings, for example with buttons to turn on or off certain LED drivers, or to choose the clock frequency or modulation technique.

5.2.3 Adding a receiver

In order to accommodate a different type of functionality, the platform should be expanded with a receiver modus. It should be possible to switch between transmitting and receiving mode through the proverbial flipping of a hardware or software switch.

5.2.4 Expanding the AXI interface

These extra settings and expansions for the VLC test platform, as well as a stream of data sent from the higher layers on the PS to the PL, or vice versa, require the AXI interface to be expanded upon. Normally, AXI4-Lite allows transfers of

only 1 word per transaction [63], which at a clock rate of 50 MHz should give 800 Mbit/s. However, in this work, the implementation of the AXI4-Lite interface is not as efficient. So on the one hand, the AXI interface should be extended to allow more commands, and on the other hand, when streaming data and higher throughputs are needed, the AXI4-Lite interface should, depending on the application, be switched to a AXI4 or a AXI4-stream interface, which allow up to 256 words per transfer (about 205 Gbit/s).

5.3 Combination with IEEE 1905.1

In a more distant future, it could be envisioned to combine this VLC test platform with a IEEE 1905.1 ("*IEEE Standard for a Convergent Digital Home Network for Heterogeneous Technologies*" [70]) backbone. This would lead the way to a more integrated indoor VLC based communication system and should expand the possibilities for managing the system. Because IEEE 1905.1 makes abstraction of the physical layer, it can leverage the performance, coverage and mobility benefits of multiple interfaces, as is mentioned in [71]. Benefits are not only better coverage and throughput, but also increased capacity by load balancing different streams over different links and increased robustness of transmissions by switching streams from one link to another in case of link degradation [71].

Chapter 6

Conclusion

The intent of this thesis was to design and implement a software-defined multiple-LED transceiver for Visible Light Communication networks.

Firstly, in chapter 1, an introduction to VLC was given, followed by an in-depth overview of the state-of-the-art, and concluded with the research question of this thesis.
The state-of-the-art overview contained open-source VLC platforms, high throughput VLC platforms, papers on different modulation techniques and two applications of VLC. It was concluded that there are an abundance of different VLC systems, which are either low in cost, with a low performance, or either more costly, with limited application potential, meaning a middle ground needed to be found.
For the research question, four characteristics of a good VLC platform arrised:

Flexibility Flexibility to use various modulation techniques, multiple LEDs, which could be assigned to specific users, and to use the system in receiving or transmitting mode. Hence, a **software-defined design** is desired.

Throughput Because the target is set at a throughput of 10 Mbit/s and performance to make the corresponding calculations sufficiently fast was important, an *FPGA* design was chosen.

Distance A functional system provides communication between the ceiling and the devices of users. Therefore, the distance between the transmitter and the receiver is considered to be **1.5 m**.

Cost The system must be priced reasonably.

Next, in chapter 2, a hardware and software design was proposed.
On the hardware level, a ZYBO was chosen as the development board for the transmitter and a ZedBoard as the development board for the receiver. The reason to use two different boards is because the receiver has to accommodate a fast ADC and the connection for this ADC was not found on a ZYBO. On the other hand, the ZYBO is cheaper and has as much Pmod connectors as the ZedBoard, which can be used to connect multiple LED drivers.

At the software level, an IEEE 802.15.4-2015 FPGA implementation would be adapted to fit the IEEE 802.15.7-2011 protocol, because it was thought that IEEE 802.15.4 and IEEE 802.15.7 shared many similarities.

In chapter 3, IEEE 802.15.7-2011 and IEEE 802.15.4-2015 were investigated in more detail and the differences between the two protocols were listed. It was found that there are more differences than initially thought.
Also in chapter 3, the available IEEE 802.15.4 implementation was described in detail. It was noticed that the implementation was not fully IEEE 802.15.4-2015 compliant, but the nice overview of which part of the code could be reused in the IEEE 802.15.7-2011 implementation was welcomed.
In conclusion, it seemed easier to start from scratch and reuse some of the IEEE 802.15.4 code, than changing this code to fit IEEE 802.15.7.

Subsequently, in chapter 4, several trials were done, but because everything in VHDL happens in parallel, the desired results were not always achieved due to erroneous reasoning during programming.
An implementation is achieved which can toggle between the different modes, using the hardware switches or an AXI interface, and an implementation which can toggle between different clock frequencies. Also, as much Pmod pins as possible were provided, together with hardcoded toggles. This way, later on, LED drivers could be turned on or off by changing these toggles.

Finally, to accomplish a usable system, additions to the current system have to be made. These additions were, together with some recommended extra reading, listed in chapter 5 as suggestions for future work.

In conclusion, a software-defined multiple LED transceiver for Visible Light Communication networks was designed, but the implementation needs additional work to be usable in further research in VLC. For example, in future theses, particularly the AXI interface should be extended or replaced and the receiver functionalities should be added.

"Protect me from knowing what I don't need to know. Protect me from even knowing that there are things to know that I don't know. Protect me from knowing that I decided not to know about the things that I decided not to know about. Amen."

Mostly Harmless
Douglas Adams

Appendices

Appendix A

Software

In this thesis, specific software has been generated and used which requires specific installation guidelines. In this chapter, these guidelines are described and certain specificities are addressed.

A.1 Setting up a virtual machine

The major part of this work was done on a virtual machine (VM) on a MacBook Pro. Here VirtualBox and VirtualBox Extension Pack [72] were used to create and run a VM with an up-to-date version of Ubuntu 16.04.5 LTS 64-bit. In Ubuntu, VirtualBox' *Guest Additions* was installed.

During this installation it was chosen to *create a new virtual hard disk*. It is recommended to assign the hard disk 55-60 GB because the installation of Ubuntu (recommended 25 GB, minimum requirement 5-10 GB) and Vivado (35-45 GB) requires a lot of disk space.

Subsequently, it is recommended to give the virtual machine as much RAM, video memory and CPU cores as possible. This way the virtual machine will not lag and slow down the actions.

System - Motherboard - Base memory ±12 GB was chosen;
System - Processor - CPU cores 4 cores are selected;
Display - Screen - Video memory size 128 MB, which means this setting was maxed out.

Finally, some extra options can be enabled in the settings of VirtualBox.

General - Advanced - Shared Clipboard *Host to guest* is selected, but either option is possible;
System - Motherboard *Enable I/O-APIC* is checked;
System - Acceleration - Hardware virtualization *Enable nested paging* is checked;
Display - Screen - Acceleration *Enable 3D-acceleration* is checked;

Ports - USB *Enable USB-controller* and *USB 2.0 (EHCI) Controller* are checked. All the USB-devices will be visible in the virtual machine;

Shared folders a shared folder is added to share the files (mostly code) needed on the virtual machine. For this, it is chosen to use a Dropbox folder so that the files stay up-to-date.

To connect USB devices (such as a ZYBO), follow the next steps (which can also be found at [73]:

- In the terminal, add your user to the group *vboxusers*: `sudo adduser USERNAME vboxusers`;
- Logout and login again;
- Connect the USB device to the host and wait for it to be recognised;
- In the virtual machine, left or right click on the USB icon in the status bar at the bottom of the screen (or in the menu, go to **Devices - USB**) and select the device.

A.2 Installing Vivado

The installation of Vivado can be performed as follows:

- On the site of Xilinx [74], the latest version of Vivado (Vivado HLx 2018.2: WebPACK and Editions - Linux Self Extracting Web Installer) can be found. Here Vivado v2017.4 (64-bit) is used. Download this file;
- In a terminal, go to the folder where the file was downloaded: `cd Downloads/`;
- Assign execute permission to the .bin file: `sudo chmod +x filename.bin`;
- Start the installer: `./filename.bin`;
- Choose *Vivado HL WebPACK*. When trying some demo's of Xillinux (see subsection 2.2.1), *Vivado HL Design Edition* should be chosen, because *Vivado HLS* will be needed to compile C-files;
- In the customization step, only *Devices - Production Devices - SoCs - Zynq-7000* is necessary, so the rest can be unchecked to safe some space. With everything checked the download will take up almost 8 GB and the Disk Space Required will be around 37 GB.
- Here, Vivado is installed in `/opt/Xilinx/Vivado/<version>/`, which will be shortened as `<Vivado Installation dir>`.

Installation of the ZYBO and ZedBoard board files. Windows users can look here for instructions: https://reference.digilentinc.com/learn/software/tutorials/vivado-board-files/start.

- Download the board files from https://github.com/Digilent/vivado-boards/archive/master.zip (or from the repository [75]) and unpack them;
- Merge the `/new/board_files/` folder with the `<Vivado Installation dir>/data/boards/board_files/` folder;
- Merge the `/old/board_parts` folder with the `<Vivado Installation dir>/data/boards/board_parts/` folder;

- (Re)start Vivado with `vivado &`. The board files should now be available.

Installation of the Zybo cable drivers:

```
cd <Vivado Installation dir>/data/xicom/cable_drivers/lin64/
↪  install_script/install_drivers/
sudo ./install_digilent.sh
```

Or alternatively, download and install the *Digilent Adept 2* drivers from https://reference.digilentinc.com/reference/software/adept/start?redirect=1#software_downloads.

A.3 Compiling the code

- Start Vivado and open `Vivado/Vivado.xpr`;
- Upgrade the CPU and reset logic blocks if Vivado asks;
- Go to *Tools -> Create and Package New IP...*;
- Select *Package a specified directory*;
- Select the ieee802157_vx directory you want to test;
- Keep clicking next until the wizard is completed;
- Go to the *Identification* tab and fill in a name for the block, check that the right files are in the source and that there are no errors, and go to the *Review and Package* tab to package the block (*Package IP*);
- In the main project,
  - If there is already a ieee802157 block, click on *Add IP*, search for the block you just created and replace the block already there;
  - If there is no ieee802157 block yet, add your own block.
    - Right click on the ports and select *Make external*;
    - Connect the clock (to FCLK_RESET0_N) and the reset (to peripheral_arestn[0:0];
    - Next, go to *Open Elaborated Design* and connect the leds, switches and scalar ports to the right pins;
- Click on *Generate Bitstream*.
- Go to `Vivado/Vivado.runs/impl_1` and use XSDB to load *Main_wrapper.bit* on the ZYBO.

A.4 Operating Systems

- Background information can be found in Part C (Operating Systems & System Integration) from The Zynq Book [76].
- When trying Xillinux, follow *Getting started with Xillinux for Zynq-7000* [40].
- Here, Embedded Linux is used. All the files, as well as a guide, were provided by Bertold Van den Bergh.

A.5 Connection with ZYBO and loading OS

To start a serial connection with the ZYBO, minicom or picocom can be used. The latter is used here.

First, take out the SD card, put the *Power Select Jumper* (jumper $JP5$) on *SD*, the *Programming Mode Jumper* (jumper $JP7$) on *WALL* and connect the power cable to the *Power Jack* ($J15$) and the micro-usb cable to the *Shared UART/JTAG USB port* ($J11$). Next, turn on the ZYBO ($SW4$ to *ON*) and add the USB device to the VM (see section A.1).

Now, make first a serial connection with Minicom (or Picocom), follow the steps under XSDB to load the OS and files onto the ZYBO and return to the Minicom screen to observe the startup and to execute commands.

A.5.1 Minicom (or Picocom)

- `sudo apt-get install minicom` or `sudo apt-get install picocom`;
- Use `ls /dev/` to see if ttyUSB0 or ttyUBS1 are available (x in the next step is 0 or 1, mostly 1);
- `sudo minicom -D /dev/ttyUSBx` or `sudo picocom -b 115200 /dev/ttyUSBx` to connect with the ZYBO;
- Press **CTRL-A**, followed by the **Z** key to open the help menu. Subsequently, choose **O** to *configure* Minicom;
- Use the arrows to select *Serial port setup*, press **ENTER** and make sure the settings are as follows:
  - **Serial Device**: /dev/ttyUSB1
  - **Bps/Par/Bits**: 115200 8N1
  - **Hardware Flow Control**: No
  - **Software Flow Control**: No
- To enter input, it could be necessary to toggle *local echo* on by pressing **CTRL-A** and subsequently **E**;
- The settings can be closed by pressing **ESC**;
- Minicom can be closed by first pressing **CTRL-A** and subsequently **X**.

A.5.2 XSDB

XSDB comes with Vivado. In the recap below, the text in italic and within quotation marks is adopted from [64].

- First, go to the folder with the necessary files (Main_wrapper.bit, rootfs.cpio.uboot, system.dtb, u-boot and uImage);
- Use `xsdb` to start XSDB and `connect` to connect with the ZYBO;

- Source the ps7_init script:

```
source ./Vivado_ieee802154/Vivado.srcs/sources_1/bd/Main/ip/
↪ Main_processing_system7_0_0/ps7_init.tcl
```

- Run `ps7_init` "*to start the CPU and DDR controller*" and `ps7_post_config` "*to turn on the FPGA fabric level shifters*".

  For only the bit-file: `fpga -f Main_wrapper.bit`;

  For the OS: Run the following command to "*select core 1, reset it, init zynq, load uboot, load flattened devicetree, load rootfs (as initramfd), start core 1, select core 2, start core 2 (runs a busyloop until Linux inits it), load fpga bitfile*"

```
target 2; rst; ps7_init; ps7_post_config; dow ./u-boot; dow
↪ -data ./uImage 0x7000000; dow -data ./system.dtb 0x6000000;
↪ dow -data ./rootfs.cpio.uboot 0x12000000; con; target 3;
↪ con; fpga -f Main_wrapper.bit
```

- XSDB can be closed by using `exit`.

In the MINICOM screen, the startup process can be followed.

Appendix B

Frame formats and commands

Bits: 0–2	3	4	5	6	7	8	9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	AR	PAN ID Compression	Reserved	Sequence Number Suppression	IE Present	Destination Addressing Mode	Frame Version	Source Addressing Mode

(a) IEEE 802.15.4, adopted from [6].

Bits: 0–1	2–5	6–8	9	10	11	12–13	14–15
Frame Version	Reserved	Frame Type	Security Enabled	Frame Pending	Ack Request	Dest Addressing Mode	Source Addressing Mode

(b) IEEE 802.15.7, adopted from [5].

Table B.1: Format of the frame control field.

Octets: 1/2	0/1	0/2	0/2/8	0/2	0/2/8	variable	variable	variable	2/4
Frame Control	Sequence Number	Destination PAN ID	Destination Address	Source PAN ID	Source Address	Auxiliary Security Header	IE	Frame Payload	FCS
		Addressing fields					Header IEs	Payload IEs	
MHR							MAC Payload		MFR

(a) IEEE 802.15.4, adopted from [6].

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Destina-tion VPAN Identifier	Destination Address	Source VPAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							MSDU	MFR

(b) IEEE 802.15.7, adopted from [5].

Table B.2: General MAC frame format.

Octets: 2	1	4/10	variable	2	variable	variable	variable	2/4
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Superframe Specification	GTS Info	Pending address	Beacon Payload	FCS
MHR				MAC Payload				MFR

(a) IEEE 802.15.4, adopted from [6].

Octets: 2	1	4/10	0/5/6/10/14	2	variable	variable	0/1	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Superframe Spec	GTS fields (Figure 47)	Pending address fields (Figure 48)	cellSearch Length	Beacon Payload	FCS
MHR				MSDU					MFR

(b) IEEE 802.15.7, adopted from [5].

Table B.3: Beacon frame format.

Octets: 2	0/1	variable	variable	variable		variable	2/4
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	IEs		Data Payload	FCS
				Header IEs	Payload IEs		
MHR				MAC Payload			MFR

(a) IEEE 802.15.4, adopted from [6].

Octets: 2	1	(As defined in 5.2.2.2.1)	0/5/6/10/14	variable	2
frame control	Sequence Number	Addressing fields	Auxiliary Security Header	Data Payload	FCS
MHR				MSDU	MFR

(b) IEEE 802.15.7, adopted from [5].

Table B.4: Data frame format.

Octets: 2	0/1	variable	variable	variable		1	variable	2/4
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	IE		Command ID	Content	FCS
				Header IEs	Payload IEs			
MHR				MAC Payload				MFR

(a) IEEE 802.15.4, adopted from [6].

Octets: 2	1	(As defined in 5.2.2.4.1)	0/5/6/10/14	1	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Command Frame Identifier	Command Payload	FCS
MHR				MSDU		MFR

(b) IEEE 802.15.7, adopted from [5].

Table B.5: Command frame format.

Command ID	Command name (802.15.4)	Command name (802.15.7)
0x01	Association Request	Association request
0x02	Association Responce	Association response
0x03	Disassociation Notification	Disassociation notification
0x04	Data Request	Data request
0x05	PAN ID Conflict Notification	VPAN ID conflict notification
0x06	Orphan Notification	Beacon request
0x07	Beacon Request	Coordinator realignment
0x08	Coordinator realignment	GTS request
0x09	GTS request	Blinking notification
0x0a	TRLE Management Request	Dimming notification
0x0b	TRLE Management Response	Fast link recovery
0x0c	Reserved	Mobility notification
0x0d	Reserved	GTS Response
0x0e	Reserved	Clock rate change notification
0x0f	Reserved	Multiple channel assignment
0x10	Reserved	Band hopping
0x11	Reserved	Color stabilization timer notification
0x12	Reserved	Color stabilization information
0x13	DSME Association Request	CVD disable
0x14	DSME Association Response	Information element
0x15	DSME GTS Request	Reserved
0x16	DSME GTS Response	Reserved
0x17	DSME GTS Notify	Reserved
0x18	DSME Information Request	Reserved
0x19	DSME Information Response	Reserved
0x1a	DSME Beacon Allocation Notification	Reserved
0x1b	DSME Beacon Collision Notification	Reserved
0x1c	DSME Link Report	Reserved
0x1d-0x1f	Reserved	Reserved
0x20	RIT Data Request	Reserved
0x21	DBS Request	Reserved
0x22	DBS Response	Reserved
0x23	RIT Data Response	Reserved
0x24	Vendor Specific	Reserved
0x25-0xff	Reserved	Reserved

Table B.6: MAC commands used in IEEE 802.15.4 and IEEE 802.15.7, adopted from [6] and [5].

Appendix C

Block diagrams

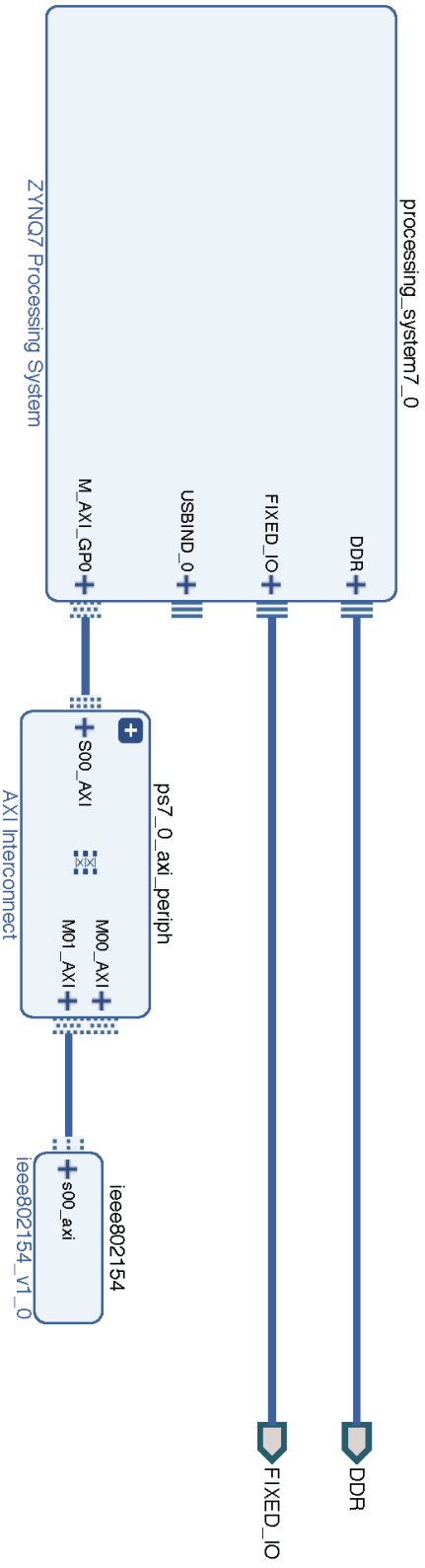


Figure C.1: Top level block diagram of the IEEE 802.15.4 implementation.

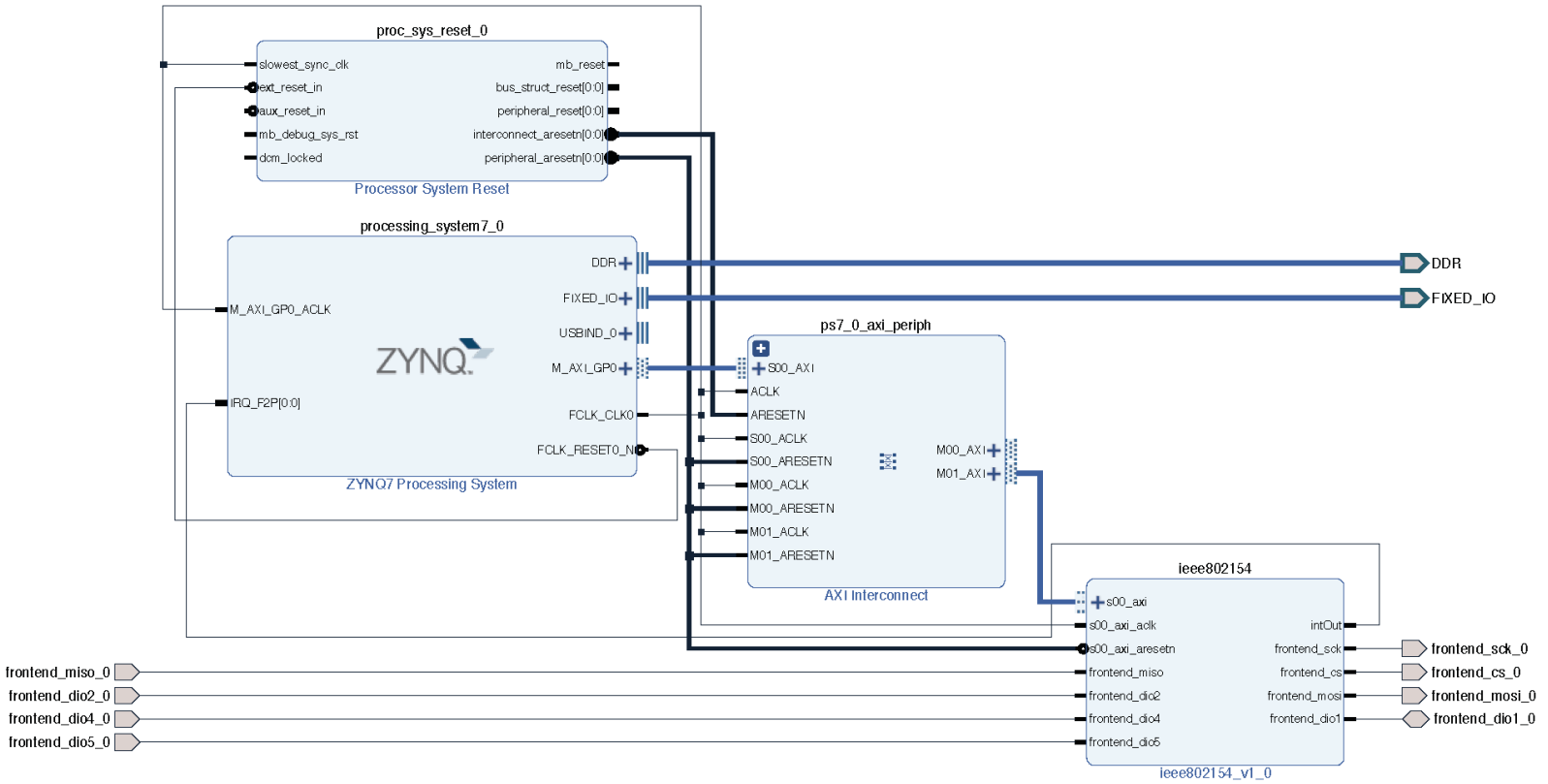


Figure C.2: More detailed top level block diagram of the IEEE 802.15.4 implementation.

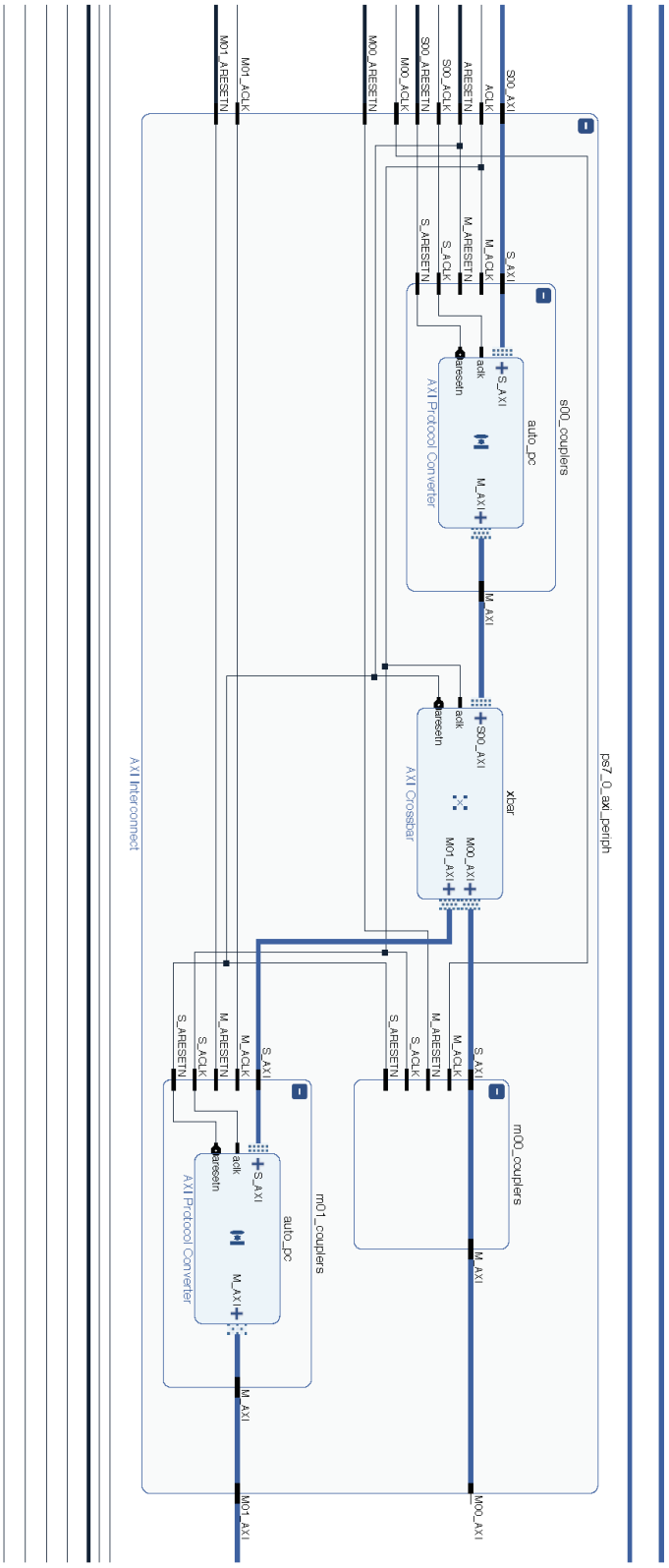


Figure C.3: Detailed AXI Interconnect block of the IEEE 802.15.4 implementation.

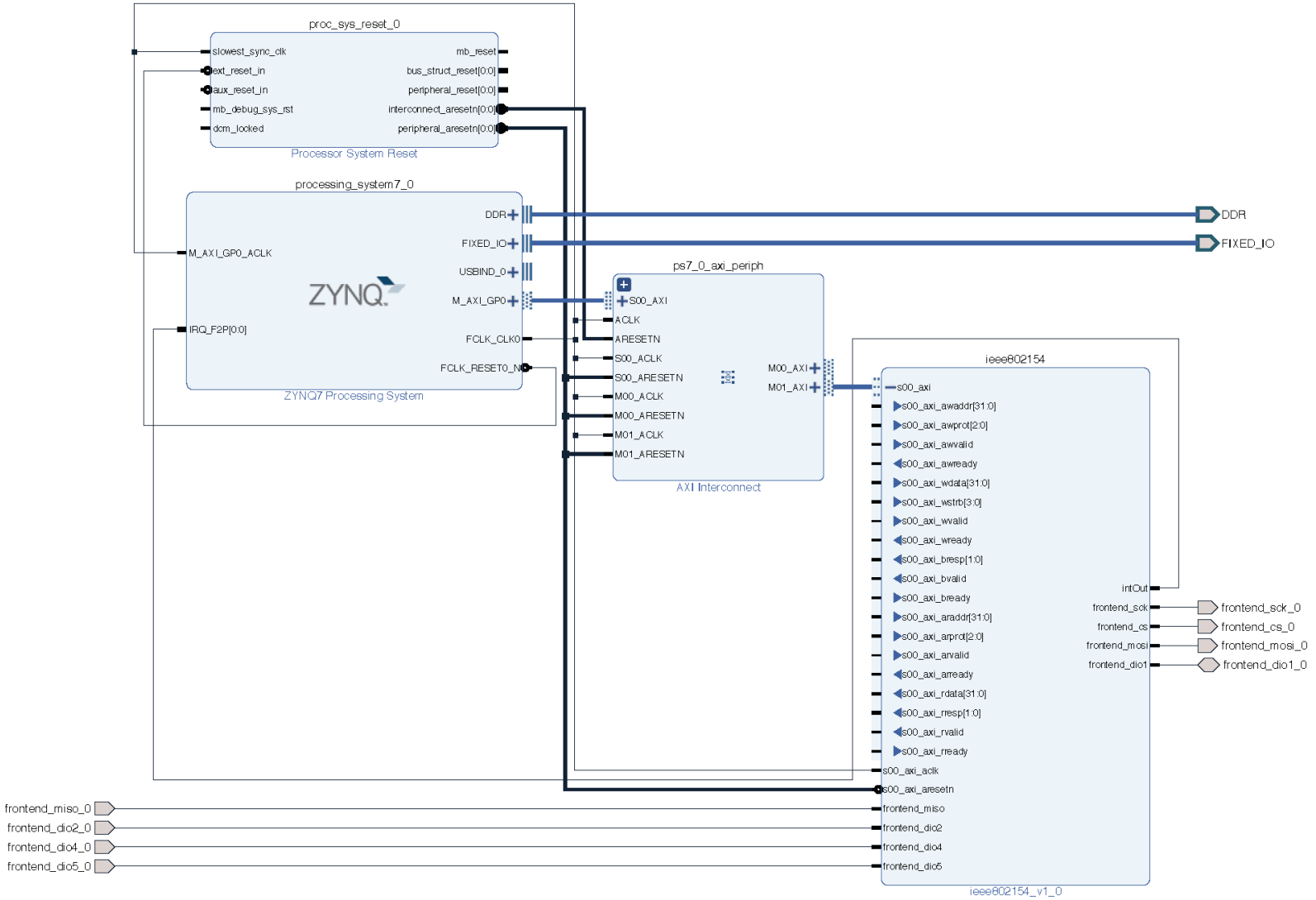


Figure C.4: Top level block diagram of the IEEE 802.15.4 implementation with detailed ieee802154_v1_0 block.

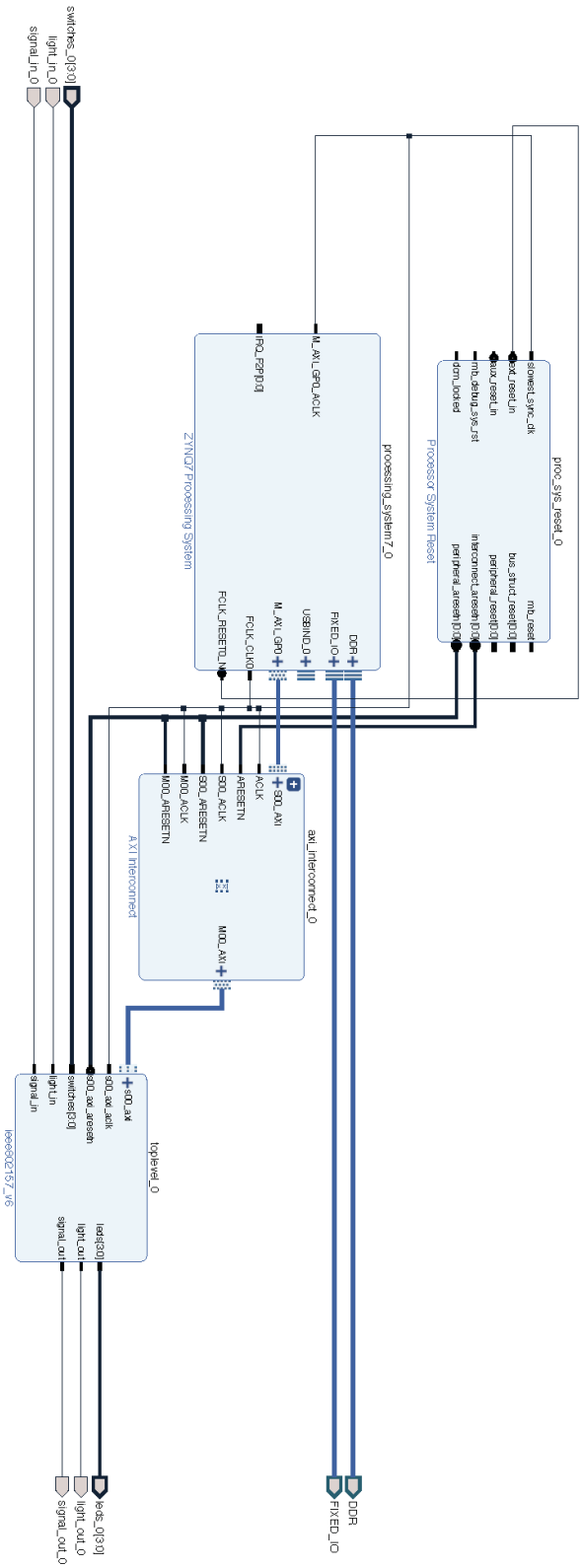


Figure C.5: Top level block diagram of the IEEE 802.15.7 implementation.

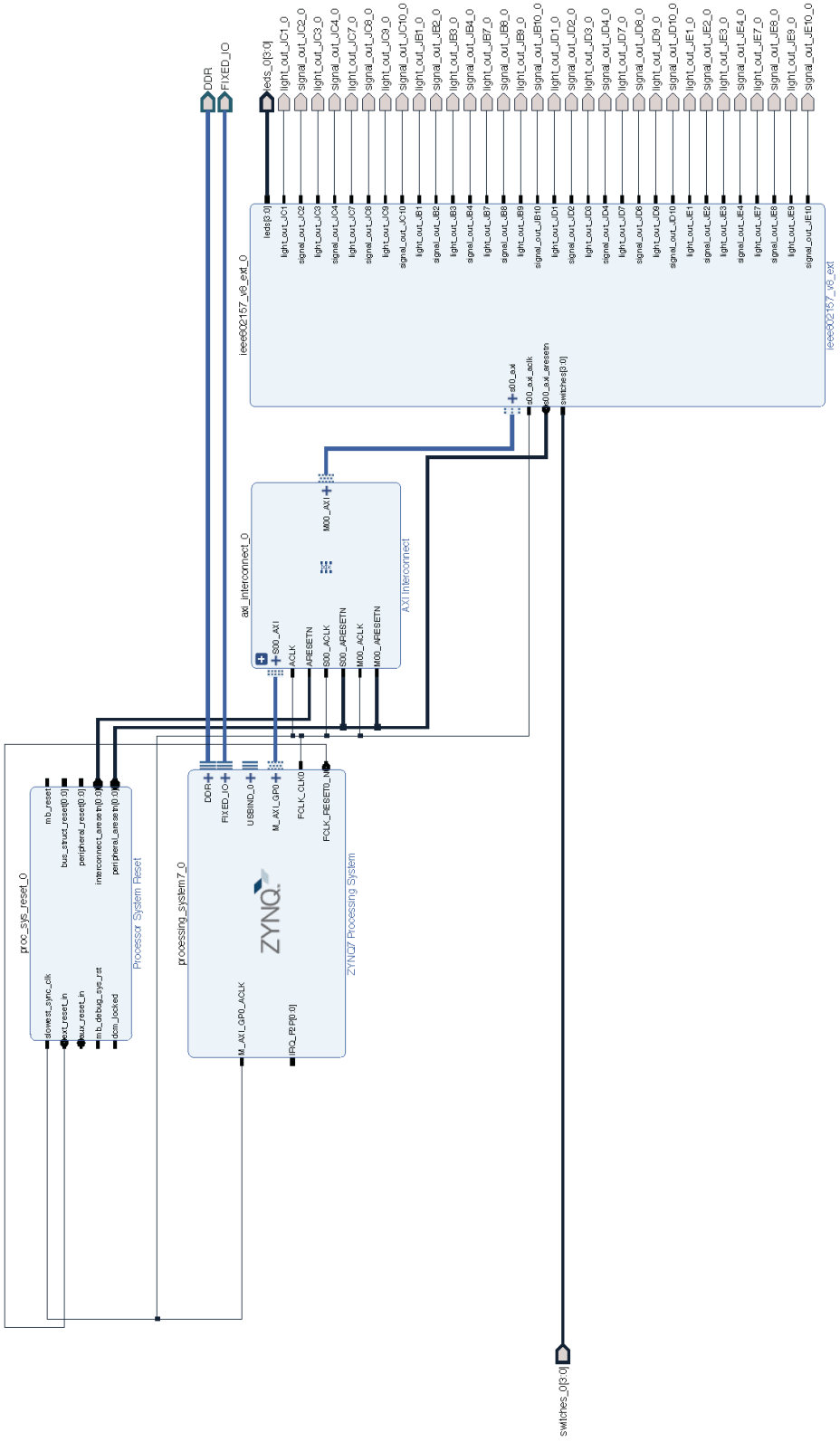


Figure C.6: Top level block diagram of the extended IEEE 802.15.7 implementation.

Bibliography

[1] Wikipedia, "Douglas adams." URL: https://en.wikipedia.org/wiki/Douglas_Adams. Last accessed: 2018-08-03.

[2] "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper." https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. Last accessed: 2018-07-29.

[3] Q. Wang, D. Giustiniano, and D. Puccinelli, "Openvlc: software-defined visible light embedded networks," in *Proceedings of the 1st ACM MobiCom workshop on visible light communication systems*, VLCS '14, pp. 15–20, ACM, September 2014.

[4] K. Hewage, A. Varshney, A. Hilmia, and T. Voigt, "modbulb: a modular light bulb for visible light communication," in *Proceedings of the 3rd Workshop on visible light communication systems*, VLCS '16, pp. 13–18, ACM, October 2016.

[5] "IEEE standard for local and metropolitan area networks part 15.7: Short-range wireless optical communication using visible light," *IEEE Std 802.15.7-2011*, pp. 1–309, Sept 2011.

[6] "IEEE standard for low-rate wireless networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, April 2016.

[7] T. Yamazato, N. Kawagita, H. Okada, T. Fujii, T. Yendo, S. Arai, and K. Kamakura, "The uplink visible light communication beacon system for universal traffic management," *IEEE Access*, vol. 5, pp. 22282–22290, 2017.

[8] "Visible light communication - vlc & purevlc™." http://3.bp.blogspot.com/-ys7U-L-rI9c/UmFEqhG8SyI/AAAAAAAAAiE/Vg0EgECVit0/s1600/VLC.png. Last accessed: 2018-07-29.

[9] K. Soni, "Li-fi : Future technology in wireless communication." https://www.slideshare.net/KapilSoniHaxinos/li-fi-light-fidelity-33632967. Last accessed: 2018-07-29.

[10] D. Tsonev, S. Videv, and H. Haas, "Light fidelity (li-fi): towards all-optical networking," in *Broadband Access Communication Technologies VIII*, vol. 9007, p. 900702, International Society for Optics and Photonics, 2014.

[11] Z. Ghassemlooy, M.-A. Khalighi, and D. Wu, *Visible Light Communications: Theory and Applications*. CRC Press, 2017.

[12] T. Komiyama, K. Kobayashi, K. Watanabe, T. Ohkubo, and Y. Kurihara, "Study of visible light communication system using rgb led lights," in *Proceedings of the SICE Annual Conference*, pp. 1926–1928, Society of Instrument and Control Engineers (SICE), 2011.

[13] J.-Y. Sung, C.-W. Chow, and C.-H. Yeh, "Is blue optical filter necessary in high speed phosphor-based white light led visible light communications?," *Opt. Express*, vol. 22, pp. 20646–20651, Aug 2014.

[14] J. Vučić, C. Kottke, S. Habel, K. Langer, A. Nerreter, K.-D. Büttner, and J. Walewski, "125 mbit/s over 5 m wireless distance by use of ook-modulated phosphorescent white leds," in *European Conference on Optical Communication, ECOC*, 2009.

[15] J. Vučić, C. Kottke, K. Habel, K.-D. Langer, S. Nerreter, A. Büttner, and J. W. Walewski, "230 mbit/s via a wireless visible-light link based on ook modulation of phosphorescent white leds," in *2010 Conference on Optical Fiber Communication, Collocated National Fiber Optic Engineers Conference, OFC/NFOEC 2010*, 2010.

[16] J. Vučić;, C. Kottke, S. Nerreter, K. Langer, and J. W. Walewski, "513 mbit/s visible light communications link based on dmt-modulation of a white led," *Lightwave Technology, Journal of*, vol. 28, pp. 3512–3518, December 2010.

[17] A. M. Khalid, G. Cossu, R. Corsini, P. Choudhury, and E. Ciaramella, "1-gb/s transmission over a phosphorescent white led by using rate-adaptive discrete multitone modulation," *IEEE Photonics Journal*, vol. 4, pp. 1465–1473, Oct 2012.

[18] J. Vučić, C. Kottke, K. Habel, and K.-D. Langer, "803 mbit/s visible light wdm link based on dmt modulation of a single rgb led luminary," in *2011 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference, OFC/NFOEC 2011*, 2011.

[19] C. Kottke, J. Hilt, K. Habel, J. Vučić, and K.-D. Langer, "1.25 gbit/s visible light wdm link based on dmt modulation of a single rgb led luminary," in *European Conference on Optical Communication, ECOC*, 2012.

[20] G. Cossu, A. M. Khalid, P. Choudhury, R. Corsini, and E. Ciaramella, "3.4 gbit/s visible optical wireless transmission based on rgb led," *Opt. Express*, vol. 20, pp. B501–B506, Dec 2012.

[21] A. H. Azhar, T. A. Tran, and D. O'Brien, "A gigabit/s indoor wireless transmission using mimo-ofdm visible-light communications," *IEEE Photonics Technology Letters*, vol. 25, pp. 171–174, Jan 2013.

[22] D. Tsonev, H. Chun, S. Rajbhandari, J. J. D. McKendry, S. Videv, E. Gu, M. Haji, S. Watson, A. E. Kelly, G. Faulkner, M. D. Dawson, H. Haas, and D. O'Brien, "A 3-gb/s single-led ofdm-based wireless vlc link using a gallium nitride *murmLED*," *IEEE Photonics Technology Letters*, vol. 26, pp. 637–640, April 2014.

[23] L. Li, P. Hu, C. Peng, G. Shen, and F. Zhao, "Epsilon: A visible light based positioning system," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, (Seattle, WA), pp. 331–343, USENIX Association, 2014.

[24] Z. Tian, K. Wright, and X. Zhou, "The darklight rises: visible light communication in the dark," in *Proceedings of the 22nd Annual International Conference on mobile computing and networking*, MobiCom '16, pp. 2–15, ACM, October 2016.

[25] I. Bar-David and G. Kaplan, "Information rates of photon-limited overlapping pulse position modulation channels," *Information Theory, IEEE Transactions on*, vol. 30, pp. 455–464, May 1984.

[26] H. Sugiyama and K. Nosu, "Mppm: a method for improving the band-utilization efficiency in optical ppm," *Lightwave Technology, Journal of*, vol. 7, pp. 465–472, March 1989.

[27] D.-S. Shiu and J. Kahn, "Differential pulse-position modulation for power-efficient optical communication," *Communications, IEEE Transactions on*, vol. 47, pp. 1201–1210, August 1999.

[28] M. Aminikashani, W. Gu, and M. Kavehrad, "Indoor positioning in high speed ofdm visible light communications," May 2015.

[29] P. Hu, P. Pathak, X. Feng, H. Fu, and P. Mohapatra, "Colorbars: increasing data rate of led-to-camera communication using color shift keying," in *Proceedings of the 11th ACM Conference on emerging networking experiments and technologies*, CoNEXT '15, pp. 1–13, ACM, December 2015.

[30] "Zybo store." http://store.digilentinc.com/zybo-zynq-7000-arm-fpga-soc-trainer-board/. Last accessed: 2017-12-04.

[31] J. Johnson, "Comparison of zynq boards." http://www.fpgadeveloper.com/2014/03/comparison-of-zynq-boards.html. Last accessed: 2017-12-04.

[32] "Zybo reference manual." https://reference.digilentinc.com/reference/programmable-logic/zybo/reference-manual#pmod_ports. Last accessed: 2017-12-04.

[33] "Specifications for the xlamp xt-e white." http://www.cree.com/led-components/products/xlamp-leds-discrete/xlamp-xt-e-white. Last accessed: 2018-08-02.

[34] Wikipedia, "Fpga mezzanine card." https://en.wikipedia.org/wiki/FPGA_Mezzanine_Card. Last accessed: 2018-07-18.

[35] "Reference manual for the zybo - pmod pins." https://reference.digilentinc.com/_detail/zybo/zybopins.png?id=reference%3Aprogrammable-logic%3Azybo%3Areference-manual. Last accessed: 2017-12-04.

[36] "Front image of the microzed." http://zedboard.org/sites/default/files/product_spec_images/front_view_overlay.png. Last accessed: 2018-07-12.

[37] "Image of an fmc carrier with microzed mounted." http://zedboard.org/sites/default/files/product_spec_images/PIC-AES-MBCC-FMC-G-with-MicroZed-112513.jpg. Last accessed: 2017-12-04.

[38] N. Stevens and L. D. Strycker, "Single edge position modulation as a dimming technique for visible light communications," *Journal of Lightwave Technology*, vol. 34, pp. 5554–5560, Dec 2016.

[39] Xillybus Ltd., "Xillinux: A linux distribution for zedboard, zybo, microzed and sockit." http://xillybus.com/xillinux. Last accessed: 2018-07-18.

[40] Xillybus Ltd., "Getting started with Xillinux for Zynq-7000." http://xillybus.com/downloads/doc/xillybus_getting_started_zynq.pdf. Last accessed: 2018-08-03.

[41] Xillybus Ltd., "Getting started with Xillybus on a Linux host." http://xillybus.com/downloads/doc/xillybus_getting_started_linux.pdf. Last accessed: 2018-08-03.

[42] Xillybus Ltd., "Getting started with the FPGA demo bundle for Xilinx." http://xillybus.com/downloads/doc/xillybus_getting_started_xilinx.pdf. Last accessed: 2018-08-03.

[43] Xillybus Ltd., "The guide to Xillybus Lite." http://xillybus.com/downloads/doc/xillybus_lite.pdf. Last accessed: 2018-08-03.

[44] Xillybus Ltd., "Xillybus host application programming guide for Linux." http://xillybus.com/downloads/doc/xillybus_host_programming_guide_linux.pdf. Last accessed: 2018-08-03.

[45] Xillybus Ltd., "Xillybus FPGA designer's guide." http://xillybus.com/downloads/doc/xillybus_fpga_api.pdf. Last accessed: 2018-08-03.

[46] Xillybus Ltd., "The guide to defining a custom Xillybus IP core." http://xillybus.com/downloads/doc/xillybus_custom_ip.pdf. Last accessed: 2018-08-03.

[47] Xillybus Ltd., "FPGA coprocessing for C/C++ programmers." http://xillybus.com/tutorials/vivado-hls-c-fpga-howto-1. Last accessed: 2018-08-03.

[48] Xillybus Ltd., "The guide to Xillybus Block Design Flow for non-HDL users." http://xillybus.com/downloads/doc/xillybus_block_design_flow.pdf. Last accessed: 2018-08-03.

[49] Commanderfranz, "Embedded linux tutorial - zybo." https://www.instructables.com/id/Embedded-Linux-Tutorial-Zybo/. Last accessed: 2018-07-18.

[50] https://buildroot.org/. Last accessed: 2018-07-30.

[51] "Code of u-boot on github." https://github.com/DigilentInc/u-boot-Digilent-Dev. Last accessed: 2018-07-30.

[52] "Code of the zybo device tree on github." https://github.com/Digilent/linux-Digilent-Dev/blob/master/arch/arm/boot/dts/zynq-zybo.dts. Last accessed: 2018-07-30.

[53] "Openvlc components." http://www.openvlc.org/openvlc.html. Last accessed: 2017-12-16.

[54] "Code of the openvlc project on github." https://github.com/openvlc/openvlc. Last accessed: 2017-12-16.

[55] "Code of the modbulb project on github." https://github.com/modBulb/modBulb. Last accessed: 2017-12-16.

[56] H. Guo, K. L. Man, Q. Ren, Q. Huang, V. Hahanov, E. Litvinova, and S. Chumachenko, "FPGA Implementation of VLC Communication Technology," in *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 586–590, March 2017.

[57] Wikipedia, "Logical link control." https://en.wikipedia.org/wiki/Logical_link_control. Last accessed: 2018-07-30.

[58] Javvin, *Network Dictionary*. Javvin Technologies, Inc, 2007. ISBN13: 9781602670006.

[59] L. Frenzel, "What's the difference between ieee 802.15.4 and zigbee wireless?." http://www.electronicdesign.com/what-s-difference-between/what-s-difference-between-ieee-802154-and-zigbee-wireless. Last accessed: 2018-07-18.

[60] "IEEE standard for low-rate wireless networks amendment 2: Ultra-low power physical layer," *IEEE Std 802.15.4q-2016 (Amendment to IEEE Std 802.15.4-2015 as amended by IEEE Std 802.15.4n-2016)*, pp. 1–52, April 2016.

[61] "IEEE standard for low-rate wireless networks amendment 4: Higher rate (2 mb/s) physical (phy) layer," *IEEE Std 802.15.4t-2017 (Amendment to IEEE Std 802.15.4-2015 as amended by IEEE Std 802.15.4n-2016, IEEE Std 802.15.4q-2016, and IEEE Std 802.15.4u-2016*, pp. 1–25, April 2017.

[62] "IEEE standard for low-rate wireless networks amendment 6: Enabling spectrum resource measurement capability," *IEEE Std 802.15.4s-2018 (Amendment to IEEE Std 802.15.4-2015 as amended by IEEE Std 802.15.4n-2016, IEEE Std 802.15.4q-2016, IEEE Std 802.15.4u-2016, IEEE Std 802.15.4t-2017, IEEE Std 802.15.4v-2017, and IEEE Std 802.15.4-2015/Cor 1-2018)*, pp. 1–51, June 2018.

[63] Xilinx, "Axi reference guide." https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf, March 2011.

[64] B. Van den Bergh, "Using FPGAs for SDR." Presentation (PDF available).

[65] Wikipedia, "8b/10b encoding." URL: https://en.wikipedia.org/wiki/8b/10b_encoding. Last accessed: 2018-08-11.

[66] R. (www.nandland.com), "Tutorial: Your first fpga program: An led blinker." URL: https://www.nandland.com/vhdl/tutorials/tutorial-your-first-vhdl-program-part1.html. Last accessed: 2018-08-12.

[67] F. Che, B. Hussain, L. Wu, and C. P. Yue, "Design and implementation of ieee 802.15.7 vlc phy-i transceiver," in *2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pp. 1–4, Oct 2014.

[68] F. Che, L. Wu, B. Hussain, X. Li, and C. P. Yue, "A fully integrated ieee 802.15.7 visible light communication transmitter with on-chip 8-w 85efficiency boost led driver," *Journal of Lightwave Technology*, vol. 34, pp. 2419–2430, May 2016.

[69] Q. Wang, D. Giustiniano, and M. Zuniga, "In light and in darkness, in motion and in stillness: A reliable and adaptive receiver for the internet of lights," *IEEE Journal on Selected Areas in Communications*, vol. 36, pp. 149–161, Jan 2018.

[70] "IEEE standard for a convergent digital home network for heterogeneous technologies," *IEEE Std 1905.1-2013*, pp. 1–93, April 2013.

[71] Wikipedia, "Ieee 1905." URL: https://en.wikipedia.org/wiki/IEEE_1905. Last accessed: 2018-08-14.

[72] "Download page of virtualbox." https://www.virtualbox.org/wiki/Downloads. Last accessed: 2018-07-18.

[73] "Set up usb for virtualbox." https://help.ubuntu.com/community/VirtualBox/USB. Last accessed: 2018-08-04; Last edited on: 2015-04-28 08:04:24 by penalvch.

[74] "Download page of vivado." https://www.xilinx.com/support/download.html. Last accessed: 2018-07-18.

[75] "The vivado board files on github." https://github.com/Digilent/vivado-boards/. Last accessed: 2018-07-18.

[76] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC*. Strathclyde Academic Media, first ed., 2014.