DESIGN AND SYSTEM IMPLEMENTATION OF PULSE POSITION MODULATION (PPM) BASED CODING SYSTEMS

by

Zohaib Ali Farhat

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

School of Computing and Engineering

University of Huddersfield

UK

May 2021

Abstract

The physical layer or modulation scheme plays a key role in a communication system, where performance features like Bit-Error Rate (BER), bandwidth efficiency and sensitivity are all dependent on the type of modulation scheme used. Currently, there are numerous modulation schemes for any given communication system, requiring the designer to decide which modulation scheme to apply. Many researchers propose different modulation schemes as the optimal for a given system with the aid of mathematical models and equations. However, there is minimal evidence on the practical implementation and testing of difference schemes to fully justify the selection. The scope of this research is to practically analyse, compare and validate the performance of different Pulse Position Modulation (PPM) schemes, where PPM is preferred for modulating and demodulating the signal in optical communications. This work presents, for the first time, practical analysis and comparison of different PPM techniques including Digital PPM (DPPM), Multipulse PPM (MPPM), Offset PPM (OPPM), Dicode (DiPPM) and Duobinary PPM (DuoPPM). The system implementation of these PPM techniques was carried under identical operational system conditions using Hardware-in-loop (HIL) approach, to validate the performance.

A visible light communication (VLC) system incorporating a high power commercial 20 W LED was used for the implementation of PPM schemes. An FPGA was used to encode the message into PPM formats and to transmit it over the LED. A comprehensive comparison was performed between several PPM schemes in terms of BER, power estimation and bandwidth utilisation. Additionally, a new modified form of MPPM, called modified MPPM (MPPM64), was proposed in this study, which improves the bandwidth utilisation of the communication system by 14.28%. Furthermore, a new error correction method for OPPM, called Priority Decoding, was proposed to improve the BER of OPPM. Experimentation revealed the improved performance of OPPM by achieving 10 times fewer errors with BER of less than 10^{-8}.

A testbench was developed, which enables the user to apply any PPM from the given PPM schemes to a communication system. This testbench can be used to evaluate the performance of the communication system and to find the suitable PPM scheme which will deliver the best performance. Error correction techniques including Parity check and Cyclic Redundancy Check (CRC) were implemented to improve system performance. MPPM64 was implemented with Parity check and CRC achieving 0.4 m in transmission distance at identical transmission speeds when compared with the original scheme. Maximum Likelihood Detection (MLSD) was implemented with DiPPM and DuoPPM, showing 40% and 37.4% theoretical improved performance, respectively. In the practical implementation of MLSD, a 10 times achievement was recorded in BER at 1.5 m, outperforming original DiPPM and DuoPPM schemes.

Determining the most appropriate PPM scheme for a VLC or any given system is not straight forward as it depends on many system parameters. However, this work enables the user to identify the most appropriate scheme for any given VLC system. This work enables different system parameters such as BER, transmission distance, power estimation and bandwidth utilization to be taken into account when determining the most appropriate setup. A detailed comparison is shown to guide modulation scheme selection in optical applications based on different parameter limitations.

Acknowledgement

I would like to offer my sincere acknowledgement to my supervisor Dr Peter Mather, for his experience, patience and assistance which helped me to reach the required level for this research. I would also like to thank my second supervisor Dr Martin Sibley for his continued support.

I would like to thank my colleges Yuxiang, Mostafa, Wahab, faculty members, research administration and other research colleges, who have made my experience wonderful during my time at the university.

I would particularly like to thank my parents, my wife and my siblings. Special thanks and gratitude to my mother and father and for their high expectations and faith in me. I would like to especially thank my wife, Haleema, who was very supportive, motivating and encouraging me during my research.

COPYRIGHT

I. The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the “Copyright”) and he s/has given The University of Huddersfield the right to use such copyright for any administrative, promotional, educational and/or teaching purposes.

II. Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the University Library. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.

III. The ownership of any patents, designs, trademarks and any and all other intellectual property rights except for the Copyright (the “Intellectual Property Rights”) and any reproductions of copyright works, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

List of Publications

- Farhat, Z. A., Ahfayd, M. H., Mather, P. J., & Sibley, M. J. "Improved BER for offset pulse position modulation using priority decoding over VLC system," Wireless Days (WD), Manchester, United Kingdom, 24-26 April 2019, pp. 1-4.
  Contribution: Proposed a new method of error correction, called Priority decoding, for Offset PPM; which improved the overall performance of Offset PPM.

- Ahfayd, M. H., Farhat, Z. A., Sibley, M. J., Mather, P. J., & Lazaridis, P. I. "Selection of high power LEDs for Li-Fi applications," 25th International Conference on Telecommunications (ICT), Saint Malo, France, 26-28 June 2018, pp. 170-174.
  Contribution: Contributed in setting up the hardware circuitry, specifically the comparator circuitry, and identified the 20 W and 30 W LEDs outperform high and low power LEDs.

- Farhat, Z. A., Ahfayd, M. H., Mather, P. J., & Sibley, M. J. "Practical implementation of duobinary pulse position modulation using FPGA and visible light communication,"*IEEE 15th Student Conference on Research and Development (SCOReD),* Putrajaya, Malaysia, 13-14 December 2017, pp. 253-256.
  Contribution: Design and implementation of Duobinary PPM modulation scheme on the FPGA. Identified Intersymbol interference as the main source of error.

- Ahfayd, M. H., Farhat, Z. A., Sibley, M. J., Mather, P. J., & Lazaridis, P. I. "Visible light communication based system using high power LED and dicode pulse position modulation technique," *25th Telecommunication Forum (TELFOR),* Belgrade, Serbia, 21-22 November 2017, pp. 1-4.
  Contribution: Design and implementation of Dicode PPM modulation scheme on the FPGA.

Contents

List of Figures

List of Tables

List of Abbreviations

AMI:	Alternate Mark Inversion
BER:	Bit-Error Rate
DAPIM:	Dual Amplitude and Pulse Interval Modulation
DPWM:	Discontinuous Pulse Width Modulation
DAPPM:	Differential Amplitude and Pulse Position Modulation
DH-PIM:	Dual-Header Pulse Interval Modulation
DPIM:	Digital Pulse Interval Modulation
DPIWM:	Digital Pulse Interval and Width Modulation
DiPPM:	Dicode PPM
DPPM:	Digital PPM
DuoPPM:	Duobinary PPM
FA:	False Alarm
HD:	Heterodyne Detection
HIL:	Hardware-in-the-loop
IMDD:	Intensity Modulation with Direct Detection
ISI:	Inter-symbol Interference
ITU:	International Telecommunication Union
LOS:	Line-of-sight
MDPIM:	Modified Digital Pulse Interval Modulation
MPPM:	Multiple Pulse Position Modulation
MLSD:	Maximum Likelihood Sequence Detection
NRZ:	Non-Return to Zero
OOK:	On-Off Keying

OPPM:	Offset PPM
PAM:	Pulse Amplitude Modulation
PCM:	Pulse Code Modulation
PD:	Photodiode
PFM:	Pulse-Frequency Modulation
PIWW:	Pulse Interval Width Modulation
PPM:	Pulse Position Modulation
PSM:	Pulse Shift Modulation
PRBS:	Pseudo-Random Binary Sequence
RZ:	Return to Zero
SWFM:	Square-wave Frequency Modulation
SNR:	Signal to Noise Ratio
TIA :	Trans-impedance preamplifier
VLC:	Visible Light Communication
VPPM:	Variable PPM

Chapter 1
Introduction

1.1 Introduction

Data communication plays an important role in daily life through emails, texts, social media and calls etc. Humans exchange data daily and the technology used in the transmission of information from one point to another is called a communication system. There are mainly two types of communication systems i.e., analogue and digital. In an analogue communication system, a continuous signal is transmitted from A to B; examples of analogue signals are voice and video signals such as Television signals. A signal in the form of an analogue signal is multiplied with a carrier signal, with different frequency, and transmitted. Analogue communication has seen a decline in recent years due to cost and space, as it requires relatively larger physical receivers and transmitters. In the digital communication system, a stream of 1's and 0's, often called a symbol, is transmitted. The sequence of 1's and 0's typically are formatted to represent information consisting of audio, video, textual or a combination of all. However, digital communication uses standard electronics and software for the transmission and reception of data which are relatively cheaper and easy to replace. Applications of digital communication systems mainly include optical fibre and optical wireless systems. [1, 2].

From the electromagnetic spectrum, given in table 1-I, the sources of Radio waves and Microwave include AM radio towers, TV and FM radio towers, mobile phones and towers [3]. Electromagnetic spectrum with frequencies ranging from 3 kHz to 300 GHz, also called Radio Frequency (RF) waves, are used in analogue communication systems. Conventional RF bands are used for broadcasting commercial television, radio and wireless communications. There are a limited number of RF bands available for broadcasting and communication purposes. Therefore, RF is becoming expensive and overcrowded. If two systems are using the same frequency band, over a shared communication channel such as air, both systems will suffer from distortion and noise due to superposition and destructive interference. Hence, RF bands are highly regulated by national laws and international telecommunication union (ITU), preventing interference between bands [4, 5]. However, optical wireless systems

remain unregulated leading to becoming a trending topic in the field of communications.

Table 1-I Electromagnetic spectrum [3]

Common name	Frequency range
Radio and Microwave	3 Hz to 300 GHz
Infrared	300 GHz to 430 THz
Visible light	430 THz to 790 THz
Ultraviolet	790 THz to 30 PHz
X-ray	30 PHz to 30 EHz
Gamma rays	>30 EHz

Optical systems use laser or light-emitting diodes (LEDs) to transmit information, by varying the intensity of the signal. LEDs have a frequency range of 430 THz to 790 THz [3]. Despite the frequencies in hundreds of THz, the possible biological effects on the human body due to optical communications may include tissue damage and damage to the immune system, which are far below the threshold levels when compared with the RF systems. These possible biological effects due to optical systems occur when an optical transmitter is applied with very high power, which is not usually applied near humans. On other hand, continuous or daily exposure to electromagnetic or RF waves may lead to chromosomal damage or DNA errors which may result in cancer [6-8]. Some optical systems based on LEDs, also known as Visible Light Communication (VLC) system, can be used for illumination purposes. On other hand, RF systems are only used for communication purposes. Optical wireless communication consisting of laser beams can provide extremely high-speed transmission rates in line of sight (LOS) systems when compared with RF [9]. In LOS, RF waves or optical energy travels in a direct path from the transmitter to the receiver. Considering above mentioned advantages of optical systems over RF, researchers are trying to develop an alternative system that can replace RF.

There is a continuous demand to achieve higher data rates and longer transmission distance in optical communications. Researchers seek to improve the transmission rates of communication systems in every possible way including integrated circuits, signal modulation techniques and physical links used for

transmission. To reduce interference, researchers propose different methods for efficient transmission of information including modulation techniques, receiver designs and channel optimisations [2]. Usually, an ideal channel in any optical system with no external interference or noise and with LOS can achieve optimal performance. It is not possible to achieve a perfect noiseless channel with zero losses in the real world, due to interference from surroundings like weather or walls of a building, this can reduce signal intensity. A modulation scheme plays an integral role in the performance of an optical system, a modulation scheme must use any given channel and achieve the optimum possible performance. Several modulation schemes are used with optical systems such as Pulse Amplitude Modulation (PAM), Pulse Position Modulation (PPM), Pulse Interval modulation (PIM) and Orthogonal frequency division multiplexing (OFDM) [10]. PPM is considered an attractive and widely used method of signal modulation in optical communications. There are several variations of PPM, e.g., Digital PPM (DPPM), Multi-pulse PPM (MPPM) etc. A PPM scheme uses the available bandwidth of the channel to achieve good performance in terms of sensitivity and error rates. VLC and other optical systems are susceptible to ambient light noise, high signal attenuation, and Intersymbol interference (ISI). ISI results in the deformation of the signal when one symbol or pulse interferes with the adjacent symbol or pulse. High optical power levels are one way to overcome these errors [11]. However, high optical power levels are not desirable in optical systems, for example, a portable system with batteries or an energy-efficient system running on low power. High optical power is also not desirable for the human eye as this could result in retina damage, hence, due to health and safety reasons only limited optical power can be applied to an optical system [12, 13].

DPPM is considered the most appropriate modulation method in optical systems, due to the low mark:space ratio and low average power of the transmitted signal [12, 14]. In PPM, a message is transmitted by positioning a single pulse or pulses in fixed time duration. Signal modulation plays a prime role in the overall performance of an optical communication system. A typical PPM scheme is given in figure 1-1. A message or data is encoded into the PPM format at the transmitter end before transmission and decoded back into the original message at the receiver end.

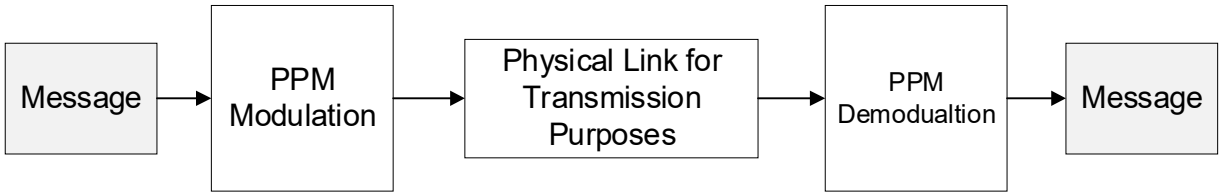


Figure 1-1 A typical PPM based communication system

1.2 Different types of PPM schemes

DPPM was proposed to utilize the bandwidth available in optical fibre and wireless systems. DPPM showed 5-11 dB improvement in the sensitivity when compared with pulse code modulation (PCM) [15]. The level of sensitivity reveals the weakest signal which will be detected by the receiver, the sensitivity of a receiver can be increased by improving the bandwidth and reducing the noise level [16]. The PCM represents the sampled analogue signal in a digital form. DPPM has been widely investigated by researchers, the only drawback DPPM offers is the large bandwidth expansion. PPM trades the bandwidth for improved sensitivity and low average power. To transmit m-bits in DPPM format, 2^m bits are required [17-21]. With the demand for data growing fast, DPPM is not suitable where limited bandwidth is available. If m=3, 8-bits are required to transmit 3-bits of data. To overcome the large bandwidth expansion, researchers have been proposing alternate PPM schemes with improved bandwidth utilization. The ideal modulation scheme must offer low average power, improved sensitivity and a reduced bandwidth expansion. However, each proposed PPM scheme inevitably comes with its advantages and disadvantages. DPPM was the earliest scheme to be investigated, which offered better average power and sensitivity than On-Off Keying (OOK) [17]. OOK is another widely investigated scheme for optical systems. OOK is the simplest form of PAM, a binary one is represented by a high voltage level, and binary zero with a low voltage level. To overcome the large bandwidth expansion and minimise average power many alternate PPM schemes were proposed. To name a few well-known PPM schemes, a list is given below:

- Multiple PPM (MPPM) [22]
- Digital Pulse Interval Modulation (DPIM) [11]
- Dual Header Pulse Interval Modulation (DH-PIM) [23]
- Offset PPM (OPPM) [24]
- Dicode PPM (DiPPM) [25]

- Duobinary PPM (DuoPPM) [26]
- Variable PPM (VPPM) [27]
- Overlapping PPM [28]

Furthermore, hybrid schemes have been proposed based on the above mentioned schemes, a hybrid scheme involves the combination of multiple modulation schemes. For example, a hybrid modulation called hybrid binary phase-shift keying modified MPPM (hybrid BPSK-modified MPPM) was proposed to improve the performance of a binary phase-shift keying (BPSK) and MPPM for optical fibre systems [29]. A hybrid PPM scheme is usually more complicated than the parent PPM scheme and results in a complex modulation/demodulation process and transmitter/receiver circuits [30]. Usually, each scheme is proposed and recommended based on the theoretical analysis. Since the DPPM proposal in the 80s, there has been a great interest in overcoming the large bandwidth expansion problems of DPPM, this has led to the discovery of many alternate and hybrid PPM schemes. However, there is very little evidence focusing on practicality and implementation concerning real-world scenarios [31-34].

1.2.1 Motivation and Novel aspects of the research

The motivation for this research comes from the insufficient published evidence of existing schemes in terms of practical implementation and measurements. The practical implementation is principally overlooked, and proposals are being based on theoretical calculations and equations. This research is an undertaking to fill this gap, along with Hardware-in-the-loop (HIL) experimentation approach which can be used to evaluate the performance of the given PPM schemes without the physical link. The most effective way to test and evaluate a PPM scheme is to implement it over a physical link. However, in some cases practical implementation is not possible in laboratory based environment, due to limited resources or to reduce cost and development time. HIL approach can be applied in such cases before implementing it over the practical system. HIL can also be applied to enhance the quality of testing, reduce development time and reduce time-to-market. In this research, a testbench serves the purpose of implementing the individual PPM schemes, enabling the performance measurement of the schemes under uniform system parameters to be evaluated. Furthermore, this testbench will be able to simulate the characteristics of a

real physical link such as inducing targeted or random errors using a pseudo-random binary sequence generator (PRBS).

In this research, a range of PPM schemes were studied and investigated over an identical communication system configuration using a developed testbench. This resulted in an enhanced understanding of PPM schemes and VLC circuitry, enabling the performance features of a given digital communication system to be investigated and validated.

Several of the PPM schemes, to the author's knowledge, were first time practically implemented and studied in a research environment. Meanwhile, this implementation also provided support in the investigation and improvement of the practical VLC circuitry. A novel PPM scheme, called Priority decoding, for error correction in OPPM, was also proposed. Furthermore, a modified form of MPPM called modified MPPM (MPPM64), was proposed which requires reduced bandwidth for data transmission, resulting in improved system efficiency. Error correction using MLSD, CRC and parity check techniques are also investigated and results analysed.

1.3 Aims

Modulation schemes play key role in the performance of any communication system and PPM schemes have always been traditionally identified as the most appropriate schemes for optical communication systems [35]. This research aims to develop a test bench to investigate, measure and enhance the performance of different PPM schemes. The developed test bench will be practically implemented over a VLC system incorporating a High Power LED light. For enhancement of PPM schemes, different error correction techniques are implemented such as CRC, Parity check, MLSD and Priority decoding.

One of the key aspects of the research was practically implementing and comparing different schemes on an identical system to identify the optimal scheme for a given VLC or any optical system. The developed test bench is scalable to add more modulation schemes and features for more comprehensive analysis in future.

1.4 Objectives

In achieving the research aim following objectives were identified:

- Investigate and gather information on different modulation schemes, with a particular focus on PPM modulation schemes used for optical communication systems.
- Identify the gaps in PPM studies regarding practical implementation.
- Investigate different PPM schemes and study the advantages and disadvantages of the schemes.
- Develop VHDL codes for different PPM schemes including DPPM, MPPM, OPPM, DiPPM and DuoPPM, to encode and decode PCM into respective PPM format.
- Design a synchronization technique using VHDL, to synchronise the incoming data from the receiver.
- Develop error detection and counting system using VHDL, to identify the total number of corrupted bits, one of the most important information when comparing modulation schemes.
- Convert developed codes into a testbench, to allow comparison and validation under identical conditions. Practically, implement, test, debug and verify the correct operation of testbench over the VLC system using FPGA.
- Develop and implement error correction VHDL codes such as MLSD, parity check and CRC, to improve the overall efficiency of the communication system.
- Using the developed testbench, investigate and improve VLC hardware circuitry.
- Examine and identify the optimal PPM scheme for the VLC system implementations.
- Identify the optimal PPM schemes in terms of error rate and complexity.

1.5 Novel contributions

The novel contributions of this research are;

- First-time practical implementation and practical analysis of OPPM, DiPPM and DuoPPM schemes.
- Implementing and evaluating the performance of multiple PPM schemes over a VLC system under identical system conditions.
- Proposing a modified version of the MPPM scheme to reduce bandwidth expansion.

- Proposing a novel method called Priority decoding to improve the BER of the OPPM scheme.
- Developing testbench using HIL approach, to test and evaluate the performance features of PPM schemes.
- Identifying VLC system component performance limitations and identifying the main sources of induced system errors.
- Future research direction recommendations have been made on the VLC system, test strategies and the novel proposed enhanced PPM schemes.

1.6 Thesis Structure

- **Chapter 1** – gives an overview of the research aims and objectives, and identifies the gaps in previous research. This chapter also identifies the novelty of this research.
- **Chapter 2** – gives the background information and history of communication systems. The importance of a modulation scheme in a communication system is discussed, including modulation schemes for optical systems.
- **Chapter 3** – Includes an introduction to the PPM scheme and its system model. This chapter includes a study of several types of PPM schemes identified and practised in this research. Two proposed novel schemes are discussed including modified MPPM64 and Priority decoding for OPPM.
- **Chapter 4** – gives a detailed overview of a VLC system and its key components. The design of the VLC setup used in this research is discussed in this chapter.
- **Chapter 5** – includes designing and system implementation of different PPM schemes on FPGA over the VLC setup. Block diagrams of each modulation schemes and the test bench are included in this chapter.
- **Chapter 6** – includes Results and Discussions.
- **Chapter 7** – contains conclusions and recommendations for future work.

Chapter 2
Background and Literature Review

2.1 Introduction

The reason behind why humans have become the most developed species is that humans produce data, store it on paper or in electronic gadgets, and trade. Particularly, the trade and dispersion of data have changed the way of life fundamentally. For example, before the phones were invented, to visit a place, one needed to plan the visit cautiously, from flight to the hotel booking and physically gather all this data about the location beforehand. After the invention of mobile phones, identical tasks can be done in minutes. Moreover, with the availability of Internet and mobile device user can check email or stock price wherever and whenever. This radical change in way of life is because of high-speed remote correspondence.

In early ages, smoke signals were used to transmit a message from one place to another place. However, the range was limited to visual distance and highly depended on weather conditions. There were also other forms of communication such as drums, carrier pigeons and the semaphore flags. These old forms of communications were highly unreliable. A Scottish physicist named C.J. Maxwell described the idea of electromagnetic waves between 1861 and 1862. To prove Maxwell's idea, a German physicist built an experimental setup in 1887, this proved the existence of electromagnetic waves. A French scientist, E. Brainly in the 1890s, invented a device consisting of a tube with two electrodes, this could detect electromagnetic waves. A revolution came in the communications field when a transistor was developed by Bell Labs. In 1954, a transistor-based radio system was developed, this was a portable wireless receiver [1, 36-38].

C. E. Shannon from Bell Labs, a scientist proposed the information theory which includes processing, extraction and utilisation of information. Shannon presented the idea of measuring the information and adopted the concept of Entropy from thermodynamics. Entropy in information theory means the level of uncertainty of a random variable. Shannon also defined the concept of channel capacity and designed the communication architecture. Channel capacity is defined as the maximum rate of reliable communications over a noisy channel. The communication architecture is

shown in figure 2-1, and this is used by all communication systems i.e., Wireline and Wireless communication systems [36, 39].

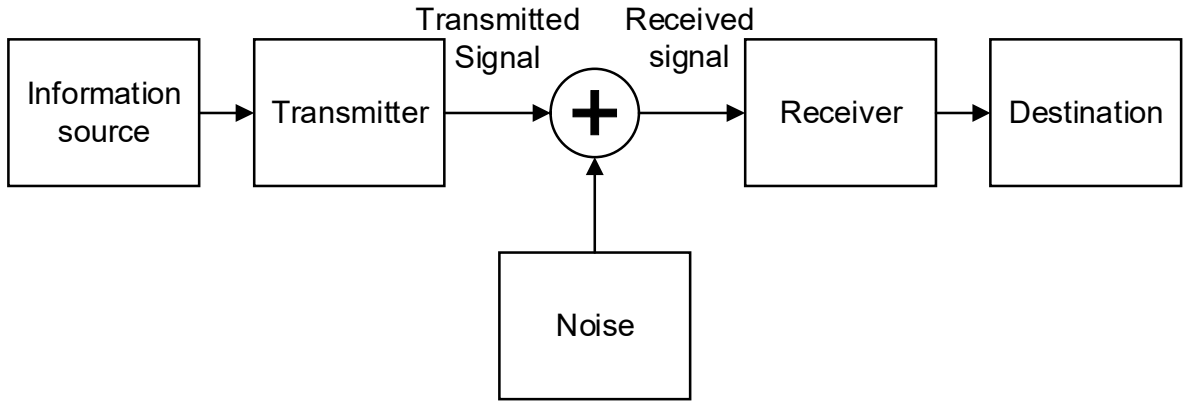


Figure 2-1 Communication architecture by Shannon [36]

2.2 Types of communication systems

Depending on the nature of the channel the communications systems can be categorized into two categories i.e., Wireline and Wireless systems. The wireless system can be further subdivided into different categories including RF, microwave and optical wireless communication systems. If a communication system is compared based on the signal, it can be categorized into two: Analogue and Digital communication systems. An optical wireless system using a digital signal as a source can be also called a digital communication system. Optical systems use infrared (IR) and light-emitting diodes (LEDs) for transmission and are becoming more and more popular due to immunity to electromagnetic interference and low cost [40, 41].

2.3 Importance of modulation in communication systems

Modulation, in simple terms, can be defined as changing or modulating the properties of a data signal using a carrier signal, where the carrier signal is higher in frequency as compared to the data signal. For example, If two signals with frequencies take 2 kHz and 2 MHz are modulated with two different carrier frequencies of 3 MHz and 5 MHz, respectively. To recover the original signal the receiver must be tuned to 3 MHz and 5 MHz. The size of the antenna is mathematically related to the modulation frequency f where $f = c / \lambda$. For effective transmission, the antenna size should be at least one-quarter of the wavelength of the transmitted signal. The transmission power

of the antenna is also related to the frequency of the carrier and is given by $P = l/\lambda$ [40].

Where λ represents wavelength, f represents frequency, c represents the speed of light, P represents the power and the length of the antenna is denoted by l.

In a digital communication system, transforming the symbols into waveforms is called modulation. The symbols must be converted into waveforms or pulses before the transmission process. The goal of the demodulator is to convert the waveform signal into a symbol [42]. Modulation is a key stage for both types of communication systems including analogue and digital system and it determines the efficiency of the communication system.

2.4 Optical Communications and its advantages

Optical wireless communications (OWC) are becoming more and more popular and have become a topic of great interest for researchers. OWC systems use IR and LEDs to transfer data. A wide range of spectrum of 300GHz to 790 THz is available in OWC systems and remains unregulated, unlike RF. OWC systems have the capability of data transmission with theoretical speeds of 670 THz, which is 10,000 times larger than the RF spectrum [43]. The LEDs or laser diodes have fast switching characteristics to achieve higher transmission speeds. On the receiver end, a photodiode (PD) is used to detect and convert the optical signal into an electrical signal. Where the transmission distance is only a few meters and/or the system is implemented indoors, LEDs are typically preferred. The LED provides the benefit of illumination along with data transfer and fast switching characteristics. LEDs are becoming popular in our daily life as compared to fluorescent bulbs due to energy efficiency [44]. This sharing of resources can reduce carbon footprint, save power and reduce the cost of operation [45].

Optical communications are preferred over microwave because of cost, size and higher speeds. OWC require low-cost front ends such as LEDs and PDs. In inter-satellite-link (ISL), an optical system is preferred, due to long-distance and high data rate requirements. The main advantage of optical systems over RF and microwave is the antenna size. In optical systems, the antenna sizes are much smaller. Figure 2-2 shows the difference between aperture sizes in RF and optical systems for a 60 GHz crosslink. In terms of terminal weight required for a 60 GHz system, OWC systems are

relatively smaller than RF systems [46, 47]. In figure 2-3, a comparison of weight is shown between OWC and RF systems.

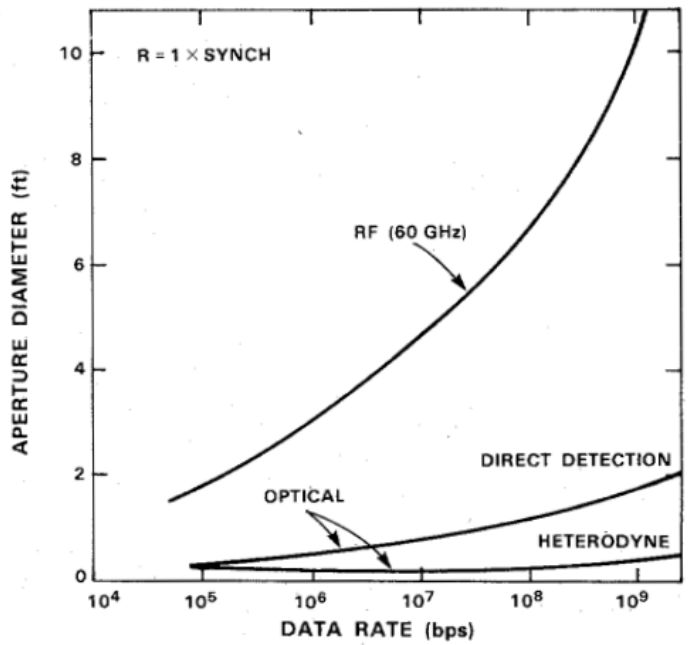


Figure 2-2 Aperture diameter vs data rate of different systems. Reproduced from: [46]

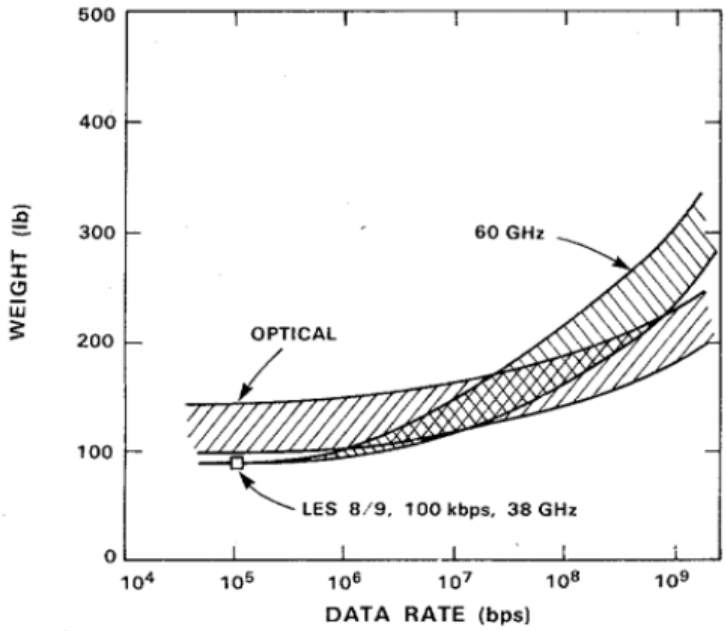


Figure 2-3 Weight comparison of different systems vs data rate. Reproduced from: [46]

2.5 Basic elements of a communication system

The basic elements of a communication system include a transmitter, a transmission channel and a receiver [48, 49].

2.5.1 The transmitter

A transmitter, within a communication system, is responsible for configuring data into a modulated signal via a carrier signal, in order to be transmitted over a channel using an appropriate antenna. The modulation process involves changing the frequency of the data signal to that of a locally generated sinusoidal signal. A similar local oscillator is required on the receiver side, to enable reliable information transfer the transmitter and receiver sinusoids must be accurately synchronized [48].

2.5.2 The transmission channel

A transmission channel is a path between a transmitter and receiver, it may refer to a physical cable, air or space. Communication occurs through the exchange of data over the channel. All communication systems suffer from channel noise, this includes Additive White Gaussian Noise (AWGN), interference and fading.

AWGN is the most fundamental assumption made in a communication system about the noise altering the transmitted signal. ‘Additive’ refers to the addition of noise to the transmitted signal, ‘White’ refers to all of the frequencies with equal strength and “Gaussian” is referred to as amplitude distribution [48, 50].

2.5.3 The receiver

A receiver is the opposite of a transmitter, a transmitter modulates the signal and a receiver demodulates the signal. It carries the same amount of information to recover the original signal. The noise introduced by the channel can impact the information which can be recovered at the receiver [48, 50].

The receiver must be synchronized with the transmitter to recover the original message and it is a fundamental requirement in any communication system. The phenomenon of the receiver determining at which time the incoming signal needs to be sampled is called timing synchronization, it involves signal frame or slot synchronizations. For synchronization of the carrier, the receiver adopts the frequency of the received signal which involves integer/fraction frequency estimation [51]. Furthermore, the synchronization should also include the correction of some errors

within a received signal, such as any timing offsets, carrier frequency offset, phase noises and sampling frequency offset caused by the following: channel delay, Doppler effect and local oscillator mismatch, phase rotation of channel and mismatch between DAC and ADC, respectively [36].

2.6 Basic modulation techniques for digital communication systems

For short-distance transmission, for example, a VLC system requires LOS and the use of baseband modulations techniques. Baseband modulation techniques are often called line coding. While in long-distance communication systems, passband modulation is preferred in which a message signal is used to alter the amplitude, frequency or phase of a carrier signal. Passband modulation is also known as Carrier modulation.

2.6.1 Baseband modulation tree

Baseband modulation is a type of internal modulation, and it is divided into two types: Isochronous and Anichronous. A modulation technique that has a fixed length of symbol is called Isochronous scheme e.g., Pulse position modulation (PPM). Anichronous schemes have different lengths of symbols e.g., Digital Pulse Interval Modulation (DPIM). Many of the schemes given in Figure 2-4 are not suitable for optical fibre and wireless systems due to power efficiency or complexity. However, PPM and DPIM are the widely used and investigated schemes for optical communication systems and have a low peak to average optical power ratio [52, 53]. One of the simplest modulation used in OWC or free-space optics (FSO) communication systems is On-Off Keying (OOK), in which binary 1 is represented by a positive signal and 0 is represented as no signal. OOK uses intensity modulation and direct detection (IMDD) receivers [54]. An IMDD receiver is a type of receiver in which the intensity of the transmitted signal is varied and the receiver recovers the signal by differentiating between two different intensity levels. IMDD is the preferred method of detection in optical communication systems. IMDD receivers are usually low cost and less complicated as compared to heterodyne (HD) receivers. In HD or coherent detection systems, a local oscillator is required which is phase-locked with the transmitter to recover the signal [17, 45, 55-57]. OOK has been studied and compared with PPM and pulse interval modulation (PIM) by many researchers for optical systems due to its simplicity and IMDD detection method. However, due to its

highest average power when compared with PPM and PIM schemes; it is not considered suitable for optical systems [12, 14]. There are three widely used and proposed techniques for IMDD systems including PPM, DPIM and OOK. However, each has its drawbacks and advantages.

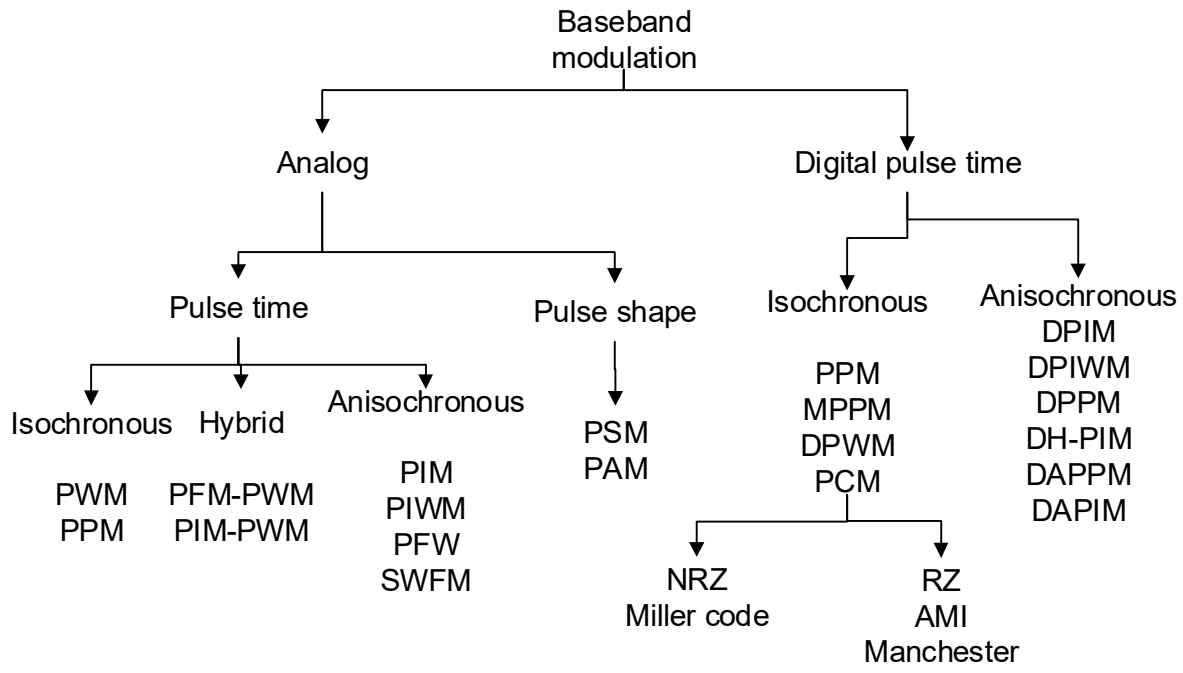


Figure 2-4 Baseband modulation tree [53]

2.7 Pulse Position Modulation

The PPM signal is formed by positioning a single pulse in every slot of 2^M total slots, where M is the number of bits in the message. The transmission rate of the system can be calculated by $M/2^M$ [10]. In optical systems, PPM is widely used and investigated due to its power efficiency and improved sensitivity when compared with OOK. PPM has been given the most attention in the optical communications field for research due to low average power consumption and the IMDD method [58-60]. One of the earliest PPM scheme is called Digital PPM (DPPM) also known as conventional PPM. In the 1980s, Garrett started investigating PPM with IMDD and HD methods and showed that the IMDD method with PPM shows better sensitivity when compared with the HD method. Sensitivity can be defined as minimum received optical power at a specific bit-error rate (BER) [17, 61]. BER is the ratio of the number of corrupted bits over total transmitted bits. HD is also known as a coherent detection method and involves the detection of phase and frequency, which makes the receiver and

modulation method complex and expensive. A local carrier is required with phased locked in HD receivers [62]. Many studies are proposing the IMDD for optical communication systems, PPM can be easily implemented using the IMDD method [35, 55]. IMDD for optical communication systems requires a simple PD and amplifier circuitry. This receiver setup is less complicated and easy to design when compared to designing receivers for HD systems such as antennas. Therefore, with the advantage of IMDD method and low average power, PPM is considered the optimal scheme for IMDD optical communications by many researchers [63].

In 2007, Fan compared the application of PPM and Pulse width modulation (PWM) for optical communication systems. It was concluded that PPM has a great advantage in power efficiency when compared with the PWM. It was also shown that PPM exhibits uniform spectral distribution [64]. In 2014, Jiang analysed OOK, DPIM and PPM schemes for ground to satellite laser communications and concluded that DPIM and PPM outperform the OOK in terms of power efficiency [65]. Another study was conducted for use of MPPM for underwater OWC systems. Several models were developed for different water types, receiver types and data rate requirements. This showed that underwater implementation of PPM based schemes is also possible [66]. From the previously mentioned advantages of PPM, it can be concluded that PPM is far more accepted for optical systems than OOK, PWM and DPIM.

2.7.1 System model of PPM

The PPM signal generation is shown in figure 2-5. In PWM, there is a wasted portion of the waveform, thus transmitted power is always wasted and conveys no information. To reduce the power in the PWM signal and improve the system efficiency, the leading edge can be differentiated to generate a pulse width train and the position of those pulses will be directly proportional to the modulating signal [67]. It must be noted that there are different ways to generate the PPM signal, only one is discussed in this thesis.

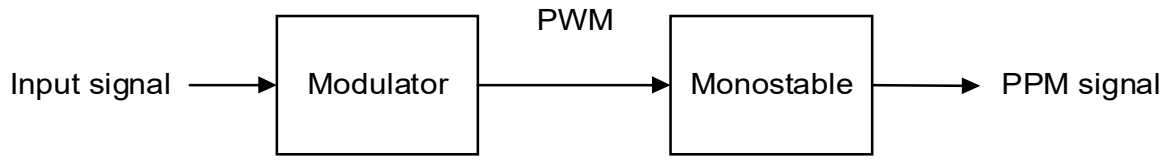


Figure 2-5 PPM system model. Reproduced from [67]

The difference between a few signal formats is given in figure 2-6. In PAM, the signal is formatted by varying the amplitude of the signal. In PWM the signal is formatted by varying the width of the signal. In PPM, the signal is formatted by varying the position of a pulse.

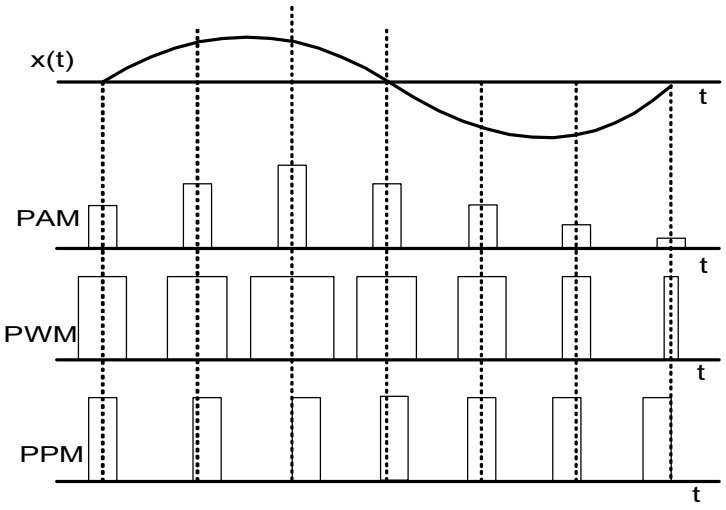


Figure 2-6 Different types of signal format compared to PPM [68]

Let x(t) denotes the PPM optical signal intensity, and the transmitted pulse shape is given by p(t), then x(t) can be represented as [64]:

$$x(t) = LP \sum_{k=0}^{K} Plk(t - kT) \qquad 2.1$$

Where LP is a factor for $x(t)$, T is the symbol interval. $Plk(t)$ represents the family of the pulse shapes and can be expressed as:

$$plk(t) = \begin{cases} 1 \; for \; t \in \left[\frac{l-1}{t}, \frac{lT}{L}\right], for \; l \in 1,2,..,L \\ 0 \quad elsewhere \end{cases} \qquad 2.2$$

Where T is divided into L intervals, l signifies the position on an interval during the $k'th$ symbol.

2.8 Types of errors in PPM schemes

All PPM schemes are affected by three types of errors including Wrong-slot (WS), False Alarm (FA) and Erasure [69, 70]. Inter-symbol Interference (ISI) can also affect the BER of the systems, which occurs when a pulse is detected in more than one slots, due to slower rise and fall times of the circuitry [17, 71]. Figure 2-7 shows

the difference between different errors sources in PPM based communication systems. Any given error sources can be responsible for errors in a system resulting in a higher BER of the system. For example, a circuit with noise will have more FA errors than Erasures. It must be noted that a symbol represents multiple bits, which means a single corrupted bit can introduce multiple errors.

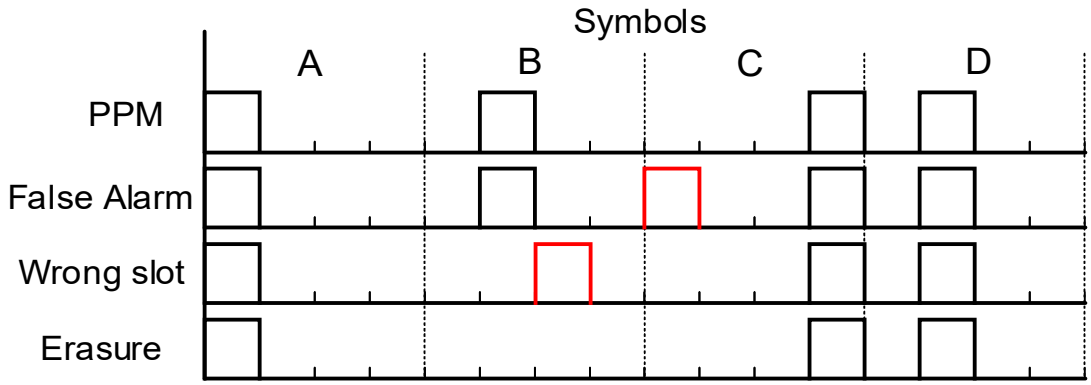


Figure 2-7 Error sources in PPM schemes [53]

2.8.1 False Alarm

This type of error arises when a pulse is detected in the next consecutive symbol due to noise in the system resulting in signal threshold crossing. In figure 2-7, a new pulse has appeared in the symbol C, resulting in a FA error. A single FA can potentially corrupt up to one symbol at a time.

2.8.2 Wrong Slot

This type of error arises when a pulse in a symbol is delayed, thus changing the whole format of the symbol and resulting in multiple errors. In figure 2-7, a pulse in the symbol B has moved from its original location resulting in a WS error. This error can occur due to the slower response of the system circuitry. If a pulse slides from one symbol to another, this can result in the corruption of multiple symbols, thus producing multiple errors.

2.8.3 Erasure

This types of error arises when the voltage detected for a pulse at the receiver is below the threshold, and the pulse remains undetected. From figure 2-7, in symbol B, a pulse disappears and remains undetected or below the threshold for detection which causing the erasure error. An erasure error can corrupt only a single symbol although it can result in multiple errors.

2.8.4 Intersymbol Interference

From figure 2-8, two signals are shown, A- input signal before the ISI effect; B- output signal after ISI effect. The broadening of the pulses in signal B causes ISI, this degrades the signal and becomes difficult to demodulate the signal at the receiver side. Thus, resulting in higher BER of the system [72].

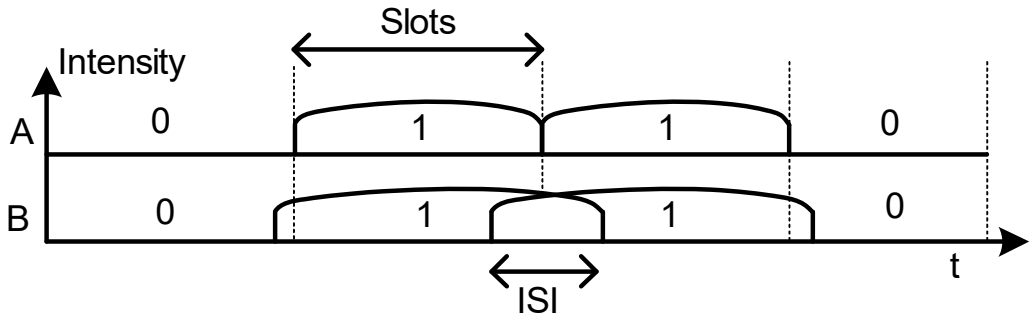


Figure 2-8 Intersymbol interference: A- signal before ISI; B- signal after ISI effect. Reproduced from [72]

2.9 Standard for choosing modulation schemes

The standard for selecting an appropriate modulation scheme is simple and the following criteria must be adopted: low power consumption, bandwidth utilization and system complexity or cost. Bandwidth utilization or bandwidth expansion are often used in communication systems to show the efficiency of the system. It refers to the appending of extra bits or binary sequence to create a symbol, which is then transmitted over the channel. The higher the bandwidth expansion the less the message or data is being transmitted, this means higher speed or a channel with large bandwidth will be required to transmit the data. Low power is the key in optical systems since the transmission power for an optical system is limited due to many factors, including eye and skin safety. If a communication system is small and portable then it puts a strain on the battery [53].

When comparing or selecting a modulation scheme from multiple modulation schemes usually the following criteria is used: Power efficiency, Bandwidth comparison, transmission capacity and Error probability. However, one can also include system implementation for comparison. Since theoretical studies or investigations do not provide details in practicality of the system in the real world. A detailed study was conducted to compare OOK, DPIM and PPM for optical communication systems, as these schemes are widely investigated and used in optical

systems [12]. OOK showed the lowest power efficiency, DPIM and PPM offer improved power efficiency at the expense of bandwidth expansion when compared with OOK. To increase the amount of information transmitted per slot in PPM, the optical pulse can be made narrow, thus, increasing transmission efficiency. However, this also results in bandwidth expansion. Figure 2-9 shows the optical pulse and bandwidth expansion issue, as you increase N the optical pulse narrows resulting in bandwidth expansion [22]. As a result transmission capacity is restricted, consequently, low bandwidth utilisation [73]. It was concluded in the study that when comparing in terms of transmission capacity and bandwidth efficiency DPIM is the most efficient scheme for optical communication systems. It must be noted that DPIM offers variable line rate since the DPIM format has variable symbol size which makes implementation complex. However, at increased symbol length PPM outperforms all [12, 23].

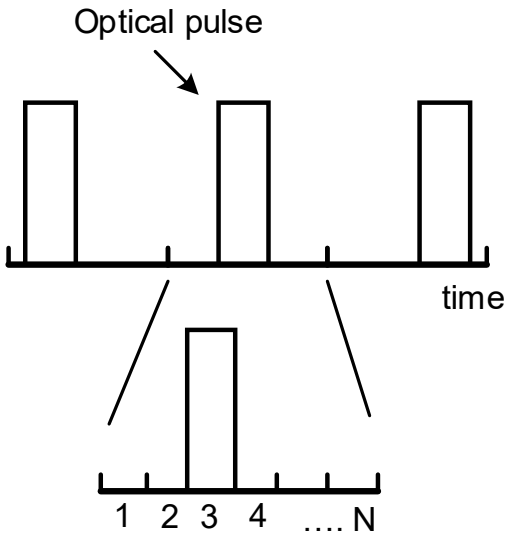


Figure 2-9 Optical pulse. Reproduced from [22]

2.10 Summary

In this chapter, a basic overview of a communication system was discussed and how it developed throughout history. The transmitter, receiver and channel are the key parts of a communication system. Optical systems are gaining more popularity due to the ability of data transfer at high speeds and immunity to electromagnetic interference. The fundamental model of a communication system and fundamental noise in a communication channel was discussed. Furthermore, the importance and the use of a modulation scheme was highlighted and how it impacts the overall

efficiency of the system. It can be concluded that DPIM and PPM are the strongest candidates for any optical system based on the IMDD system. However, it must be noted that DPIM has variable symbol size, hence, implementation is complicated when compared with PPM.

Chapter 3

Pulse Position Modulation Schemes

3.1 Introduction

There are many variants of the PPM scheme, however, it is not possible to consider all the PPM schemes in this study due to limited time and resources. Following PPM schemes have been selected for this study:

- Digital PPM
- Multiple PPM
- Offset PPM
- Dicode PPM
- Duobinary PPM

Along with above mentioned schemes, author has proposed two novel PPM schemes named Priority decoding for Offset PPM and Modified Multiple PPM (MPPM64). The former is proposed to reduced the errors in OPPM and the latter is modified version of original multiple PPM (MPPM74) with lower bandwidth expansion.

3.2 Digital PPM (DPPM)

In DPPM, 2^M slots are required to transmit M-bits of the message, which means when M=3, 8 bits of message can be transmitted, increasing the bandwidth by 2.67 times the size of the actual message. If M=4, 16 bits are required increasing the bandwidth by 4 times of the actual message. Table 3-I shows the DPPM format where M=3. It compromises the bandwidth of the system for a signal-to-noise ratio. It also has a lower mark: space ratio as compared to other conventional modulation schemes like OOK [74, 75]. From previously mentioned features and advantages, PPM gained popularity and the National Aeronautics and Space Administration (NASA) intended to use PPM in free-space optical links. The infrared data association (IrDA) classified PPM (M=4) as standard for 4 Mbps serial communication, due to its good average power in optical channels [20]. The only major drawback PPM offers is large bandwidth expansion, which makes its practical implementation costly with a lower data rate, which led to discoveries of other PPM schemes [71], discussed later in this chapter.

Table 3-I DPPM symbol format

Dataword (M=3)	DPPM format
000	0000 0001
001	0000 0010
010	0000 0100
100	0000 1000
100	0001 0000
101	0010 0000
110	0100 0000
111	1000 0000

3.3 Multipulse PPM (MPPM)

In 1989, Sugiyama and Nosu proposed MPPM with the benefit of lower bandwidth expansion when compared to DPPM. In MPPM, to transmit k pulses per codeword, $\binom{N}{k}$ combinations can be formed, and the data transmitted per codeword will be $\log_2\binom{N}{k}$ bits. Where N represents the size of a symbol and *k* represents the number of pulses [22]. For example, if *k*=2 and N=7, the total combinations can be 21 with a dataword size of 4. However, in DPPM with a similar dataword size,16 bits of the codeword is required. Message bits are referred to as Dataword, and the respective symbol for dataword is referred to as codeword. Table 3-II shows the MPPM format conversion. MPPM was studied for a VLC system for dimming control and it was found that it outperforms Variable OOK (a variation of OOK) and Variable PPM (variation of PPM) in achieving higher spectral efficiency requiring less optical power [76]. Another study was conducted to compare MPPM with DPPM, it was found that when N is fixed MPPM can outperform conventional DPPM with the benefit of improved bandwidth utilization. MPPM offers improved bandwidth utilization when compared to DPPM, however, it does expand the bandwidth of the system by at least 1.75 times the size of dataword.

Table 3-II MPPM symbol format

Dataword	MPPM (7,4)
0000	1100000
0001	1010000
0010	1001000
0100	1000100
0101	1000010
0110	1000001
0111	0110000
1000	0101000
1001	0100100
1010	0100010
1011	0100001
1100	0011000
1101	0010100
1110	0010010
1111	0010001

3.4 Offset PPM (OPPM)

Sibley proposed a version of DPPM called Offset PPM (OPPM), which has half the line rate of DPPM. It showed that it offers better sensitivity when compared with DPPM. Sibley showed theoretical results at a rate of 1 Gbps and concluded that OPPM offers 3.1 dB increase in sensitivity [24]. Table 3-III shows the OPPM format conversion. Another theoretical study was conducted comparing OPPM with DPPM and OOK, it was shown that OPPM offers 3.27 dB sensitivity improvement over DPPM when both are operating at a normalized channel bandwidth of 3 and 6 coding levels. It was also showed that OPPM significantly outperforms OOK with a sensitivity advantage of 10 dB [61]. OPPM can be considered a feasible alternate for DPPM and MPPM. It offers improved bandwidth utilization when compared with DPPM and MPPM, by increasing the bandwidth of the system by at least 1.33 times the size of dataword.

Table 3-III OPPM symbol format

Dataword	OPPM format
000	*0* 000
001	*0* 001
010	*0* 010
011	*0* 100
100	*1* 000
101	*1* 001
110	*1* 010
111	*1* 100

3.5 Dicode PPM (DiPPM)

DiPPM was proposed by Sibley [25] in 2003, Dicode and DPPM signalling were combined to develop the DiPPM, which shows improved bandwidth utilization. DPPM has the lowest mark:space ratio, however, DiPPM can be combined with error correction techniques to overcome lower sensitivity issues. MLSD and RS codes can be used to further improve the sensitivity issue of DiPPM [77, 78]. In DiPPM, a transition from 0 to 1 is represented by a set pulse (S) and a transition from 1 to 0 is represented by a Reset (R) pulse. S pulse is always placed in the slot where the clock signal is high, similarly, the R pulse is placed where the clock signal is low. When a signal is constant it is represented by no pulse (N). Table 3-IV shows the DiPPM data format [25]. Figure 3-1 shows the PCM to DIPPM format conversion without guard pulses [79]. A study was conducted proposing DiPPM for optical fibre and wireless systems due to its lower line rate and IMDD characteristics [15]. When implemented without guard slots the line rate becomes equal to the data rate.

Table 3-IV DiPPM symbol format

Data	DiPPM
00	No Pulse (N)
01	Set (S)
10	Reset (R)
11	No Pulse (N)

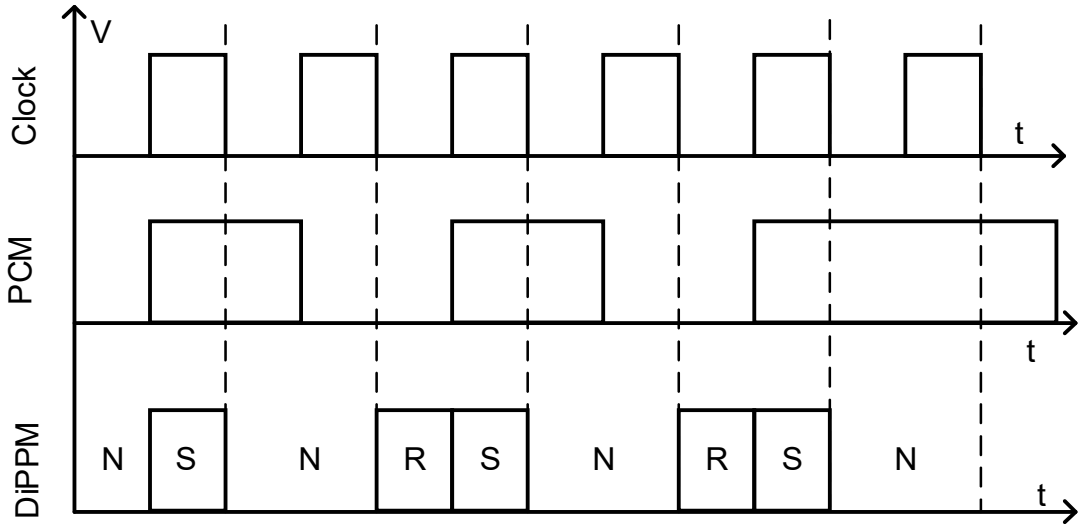


Figure 3-1 PCM to DiPPM conversion [79]

3.6 Duobinary PPM (DuoPPM)

DuoPPM is the opposite of DiPPM in signalling format where constant pulses are generated when the signal is at constant high or low, and when the signal is transitioning there is no pulse. When the signal is constant 1, a set (1) pulse is placed and when the signal is constant low a reset (0) pulse is placed. Table 3-V shows the DuoPPM format [26]. DuoPPM was first time practically implemented by Farhat to investigate the possible application over VLC setup [80]. Figure 3-2 shows the PCM to Duobinary signal conversion. Similar to DiPPM, DuoPPM offers improved bandwidth utilization than OOK and DPPM. It has twice the line rate of Non-Return to Zero (NRZ) OOK. To further improve the sensitivity of DuoPPM it can be used with MLSD [26]. When implemented without guard slots the line rate becomes equal to the data rate.

Table 3-V DuoPPM symbol format

Data	**DuoPPM**
00	Pulse in slot 0
01	Change (C)
10	Change (C)
11	Pulse in slot 1

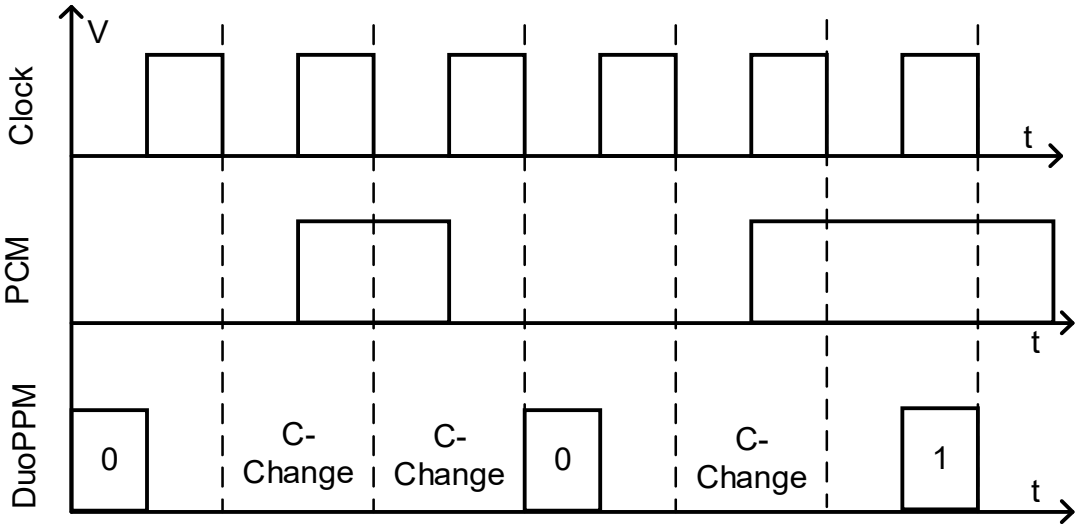


Figure 3-2 PCM to DuoPPM conversion [80]

3.7 Priority Decoding for OPPM- a novel method for improving BER in OPPM

A novel method for error correction – priority decoding for OPPM is proposed in this research. The OPPM requires half the line rate as compared to DPPM, hence it was proposed to utilize the bandwidth available in optical channels. OPPM converts 3-bit size of dataword into 4-bit size of codeword, as given in figure 3-3. There are a total of 16 (2^4) codeword combinations available, however, only 8 of the codewords are valid for OPPM, with the remaining 8 being invalid. As PPM schemes suffer from FA, Erasure, WS and ISI errors [12], which corrupt the received codeword and a single error can result in multiple errors in the decoded data. Priority decoding is proposed to remove invalid codewords at the receiver. Many schemes usually request retransmission of a packet when an invalid codeword or packet is received, which means a lower overall line rate and compromised system bandwidth, however priority decoding is configured so that no retransmission of the data packets or codewords is required. Furthermore, the motivation for Priority decoding comes from designing an error correction technique that does not compromise the bandwidth and transmission rate of the system. This novel scheme attempts to convert an invalid codeword into a valid one at the receiver.

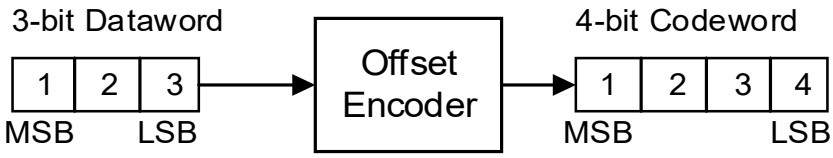


Figure 3-3 OPPM conversion format

From table 3-VI, the maximum number of 1's in OPPM is two, when priority decoding detects more than two 1's, the system can readily identify this as a corrupted codeword. OPPM also has no more than one '1' in the last three bits of the codeword, which can also be used to detect an invalid codeword. Referring to figure 3-3, assume bit 3 has the highest priority followed by bit 2 and 4, respectively. If a bit with the highest priority is detected as '1', the remaining bits are set to 0's i.e., bit 2 and 4. However, if the MSB bit is corrupted, a valid codeword becomes another valid codeword. Hence, it is not possible to detect the error in this case. Priority decoding requires a simple digital logic for implementation.

From table 3-VII, three invalid codewords are received at the decoder and a comparison between the original and the Priority decoder is shown. As it can be seen the original OPPM will receive invalid codewords and produce errors. However, the OPPM converts these invalid codewords into valid codewords. When the MSB is corrupted, the priority decoder is unable to detect the error such as in the case of *1* 001, which is a limitation of this system.

Table 3-VI PCM to OPPM and DPPM conversion

PCM (Dataword)	**OPPM codeword**	**DPPM codeword**
000	*0* 000	0000 0001
001	*0* 001	0000 0010
010	*0* 010	0000 0100
011	*0* 100	0000 1000
100	*1* 000	0001 0000
101	*1* 001	0010 0000
110	*1* 010	0100 0000
111	*1* 100	1000 0000

Table 3-VII OPPM with Priority Decoding method

Corrupted word	Original OPPM decoder	OPPM decoder with Priority decoding
0 110 (invalid)	*0* 110 (invalid)	*0* 100 (valid)
1 011 (invalid)	*1* 011 (invalid)	*1* 010 (valid)
1 001 (invalid)	*1* 001 (invalid)	*1* 001 (invalid)

3.7.1 Limitations

Priority decoding has the capability to correct ISI, FA and WA, however, it can not correct erasure errors. In the case of erasure errors, a valid codeword becomes another valid codeword which is a limitation of the priority decoding. Furthermore, It must be noted that the priority order can be changed, however, it will not increase or decrease the overall error correction percentage or BER of the system. Since the total number of combinations for each priority location is the same, resulting in the same overall probability for each priority location.

3.7.2 Evaluation methodology

Priority decoding will be designed and implemented on an FPGA, for practical analysis. It will be evaluated by comparing the performance features of the original OPPM against the Priority decoding for OPPM such as BER and error probabilities. This simple comparison yet very effective will give a reasonable indication of Priority decoding capability to correct errors.

3.8 Modified MPPM- a novel modified form of MPPM to reduce bandwidth expansion

The original MPPM also referred as MPPM74, has a codeword size of 7, as shown in table 3-VIII, it appends three more bits to the dataword before transmitting the signal. The total number of available codewords in the original MPPM are 128 (2^7), however, only 16 codewords are valid. In this research, a novel modified form of MPPM is proposed to reduce the bandwidth expansion, called Modified MPPM64. In Modified MPPM64, the codeword size is reduced to 6, thus reducing the total combinations to half i.e., 64. Having fewer invalid codewords means the probability of receiving an invalid codeword is halved. Another advantage of having a smaller

codeword means that a similar amount of data can be transmitted at lower line rates resulting in fewer errors in the system.

To achieve a smaller 6-bit codeword for modified MPPM, two pulses were removed from the relevant codeword for 0000b and the codeword was converted to all zeros as given in table 3-VIII. For example, to achieve a data rate of 14 MHz, MPPM74 requires a line rate of 24.5 MHz. On other hand, Modified MPPM64 offers a similar data rate at the line rate of 21 MHz.

Table 3-VIII MPPM and Modified MPPM comparison

PCM	Original MPPM74	Modified MPPM64
0000	1100000	000000
0001	1010000	110000
0010	1001000	101000
0011	1000100	100100
0100	1000010	100010
0101	1000001	100001
0110	0110000	011000
0111	0101000	010100
1000	0100100	010010
1001	0100010	010001
1010	0100001	001100
1011	0011000	001010
1100	0010100	001001
1101	0010010	000110
1110	0010001	000101
1111	0001100	000011

3.8.1 Limitations

Reducing the size of the codeword effectively decreases the mark to space ratio of Modified MPPM64, which is thus a limitation of this scheme. However, despite the low mark to space ratio the low line rate feature of this novel MPPM can offer reliable data transmissions. Another limitation of MPPM64 could be all 0s in a codeword, having all zeros can compromise the synchronisation mechanism of the system, thus, resulting in errors or loss of data. Many systems are synchronised using

clock edge pulses, if there are no pulses in the codeword, the system can effectively become asynchronous, which could result in loss of data.

3.8.2 Evaluation methodology

MPPM64 will be practically implemented on an FPGA to measure and compare its performance to the original MPPM74. The modified MPPM64 will be evaluated by comparing its performance features to the original MPPM such as BER, error probabilities and bandwidth utilisation factor.

3.9 Summary

In this chapter, the system model of the PPM was presented. Different types of PPM were discussed with advantages and limitations, in terms of optical systems. This chapter also explains the need for alternative PPMs which is bandwidth utilization, and how each PPM differs from others. Different types of errors that arise in a PPM based communication system were explained. These error sources can introduce multiple errors in the system. Two novel schemes are proposed in this chapter called Priority Decoding for OPPM and Modified MPPM64, for reducing the errors and bandwidth expansion in OPPM and MPPM, respectively.

Chapter 4
VLC principle and experimental setup

4.1 Introduction

In recent years, LEDs have become more popular and their use in OWC has been growing. The growing interest and research on LEDs are due to lower power consumption, long life span, low cost and ability to transmit data. An OWC system that uses visible light for communication is referred as Visible Light Communication (VLC) system. With advancements in electronics, a VLC system can potentially transfer at higher data rates than RF systems [53, 81]. VLC offers the benefit of illumination with data transmission which means resources can be shared and the data cost can be reduced.

In the early years of 21st century, Japan was leading the way in the VLC system [53]. From handheld devices to vehicle-to-vehicle data transmissions were demonstrated along with the efficiency. Moreover, a VLC system using fluorescent and LEDs was developed and studied. A LED-based display was also presented by a company in Japan named Fuji Television, this display could transfer the data using the backlight. Japan Electronics and Information Technology Industries Association (JEITA) has proposed two standards for VLC systems encouraging researchers to investigate and implement VLC. Moreover, a project funded by European Union called OMEGA demonstrated that a VLC system can transfer data up to 100 Mbps. In the OMEGA project, a global standard for use of the VLC system was proposed. Another study was published by researchers from the University of Edinburgh on VLC, using the Orthogonal Frequency-Division Modulation (OFDM) technique with white LED and showed this system can transfer up to 124 Mbps. VLC was also used for Local Area Network (LAN), and JIETA proposed a new standard for visible light beacon system [82]. The use of optical fibre and free-space optical systems in local area network (LAN) is growing fast [83]. In 2013, a low-cost VLC system was developed and demonstrated using IEEE802.15.7 standard [84]. For underwater communications, radio communication is not desirable due to high noise, and the only tool for communication used by divers is ultrasonics which is affected by reflections from nearby objects or ground. A VLC system can be efficient and solve all the problems regarding underwater communications [85].

4.2 Applications of VLC

There can be many applications of the VLC system such as:

- Internet and Cellular connectivity within a confined space such as home or office.
- Connecting multiple computers within the same building or office to support LAN.
- Vehicle to vehicle communication to support Intelligent Transport System for reducing traffic accidents and to improve traffic flow.
- Smart cameras to capture more detailed images.
- Entertainment activities in themed parks using visible light.
- VLC can be used for underwater communication as the ultrasonic do not travel well underwater.

4.3 Advantages of VLC

VLC offers a wide range of advantages when compared with RF system, such as:

- Wide spectrum range from 400 THz to 780 THz.
- VLC experience zero electromagnetic interference, unlike RF systems.
- Implementation and cost are lower and can be implemented on existing illumination sources.
- Energy efficiency

A VLC system based on LED does not produce harmful RF waves. However, optical power must be limited to reduce the effect on the eyes.

4.4 VLC principle and system description

To achieve connectivity in a VLC system the intensity of light is varied i.e., turning the LED off and on. The switching speed is unable to detect with the naked eye, which is beneficial as the light produced by the system can also be used for illumination. On the receiver end, a photodiode (PD) is used to detect the change in light intensity. A PD is a type of sensor that produces current in the presence of photons or light, this current is proportional to the intensity of light or quantity of photons. In figure 4-1 shows the principle of the VLC system, this simple setup is used by researchers to fulfil illumination and data transmission needs. On the receiver side, a higher current at PD will represent 1, and lower currents will represent 0 [45].

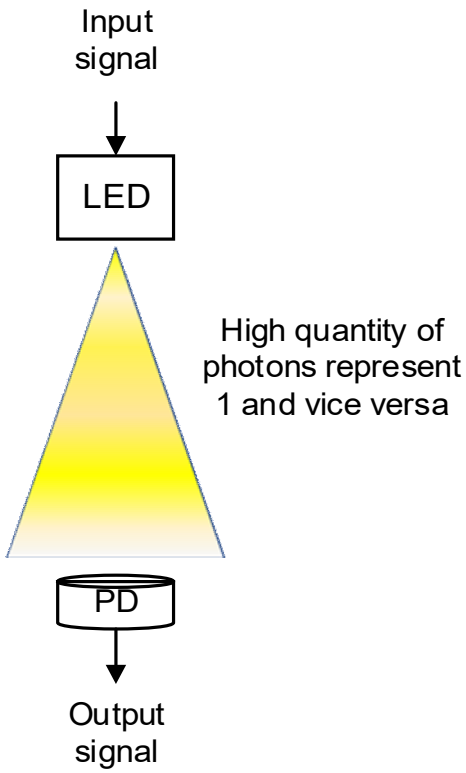


Figure 4-1 Visible Light Communications principle [45]

In figure 4-2, the block diagram of the VLC system is given. The diming control circuit is used to manipulate the light intensity at a significantly higher frequency up to nanoseconds. A fluorescent bulb in this setup is not desirable and difficult to control. On the receiver end, if PD and signal conditioning circuits can control higher frequencies a VLC system can potentially transmit data at GHz. It is easy and cost-effective to generate a low power signal for transmission [53].

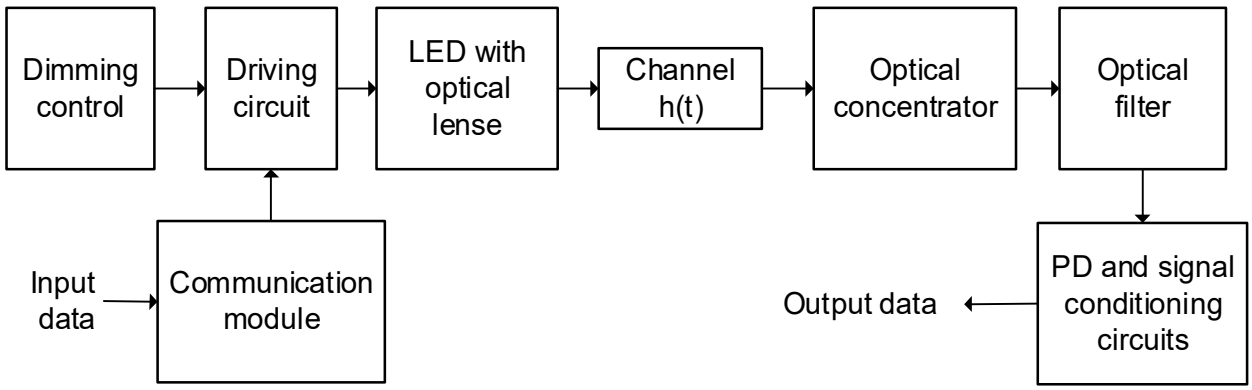


Figure 4-2 Block diagram of a VLC system [53]

4.5 Basic elements of VLC system

To achieve successful transmission in a VLC; the following three components play an important role [45].

4.5.1 Physical layer (modulation/demodulation)

Signal modulation and demodulation play a key role to achieve optimal performance in any communication system. The physical layer encodes the message signal using the desired modulation schemes such as PPM. After modulation, the signal is transferred to LED as an electrical signal.

4.5.2 LED

LED is used to convert electrical energy into optical energy. In the VLC system, LED is used as a transmitter and also provides significant light which can be used for illumination. A LED is a forward-biased p-n junction semiconductor, under suitable conditions it can emit radiations in different regions of the spectrum; including visible light, ultraviolet and infrared regions [86].

4.5.3 Photodiode

On the receiver end, the optical energy emitted by LED is captured at PD and converted into an electrical signal. This electrical signal is demodulated to extract the original message.

4.6 Experimental setup

The block diagram of the experimental link used is given in figure 4-3 and a more detailed setup is shown in appendix A. The fundamentals of the experimental setup were developed by Ahfayd [87]. An FPGA was used to transmit and receive the data and was programmed in VHDL. Furthermore, FPGA was also used for modulation of signal into PPM format. Firstly, the encoded signal in PPM format is transferred to Bias-T, the other input to Bias-T is the DC signal. A Bias-T combines both signals i.e., DC and AC signals, and sends the output signal to the LED. This output electrical signal is converted into an optical signal by the 20 W LED of temperature 6000 K. A convex lens was placed just above the PD to converge the optical signal on to PD. The variation in the intensity of the optical signal is determined by the frequency of the encoded signal. The output of PD is connected to signal conditioning circuits including Trans-Impedance amplifier (TIA) and RF amplifier. The

output of the signal condition circuit was connected to a comparator to digitize the analogue signal. The received signal is sent back to FPGA for decoding. The decoded signal is compared with the transmitted signal to calculate the Bit-error rate (BER) of the system. The variation in optical power is significantly lower and cannot be detected with the naked eye. The encoded signal is significantly lower in power and amplitude as compared to the DC signal. Hence, this VLC can also be used for illuminations. The LED is connected to an adjustable stand to adjust the transmission distance between 1.4 m to 3 m.

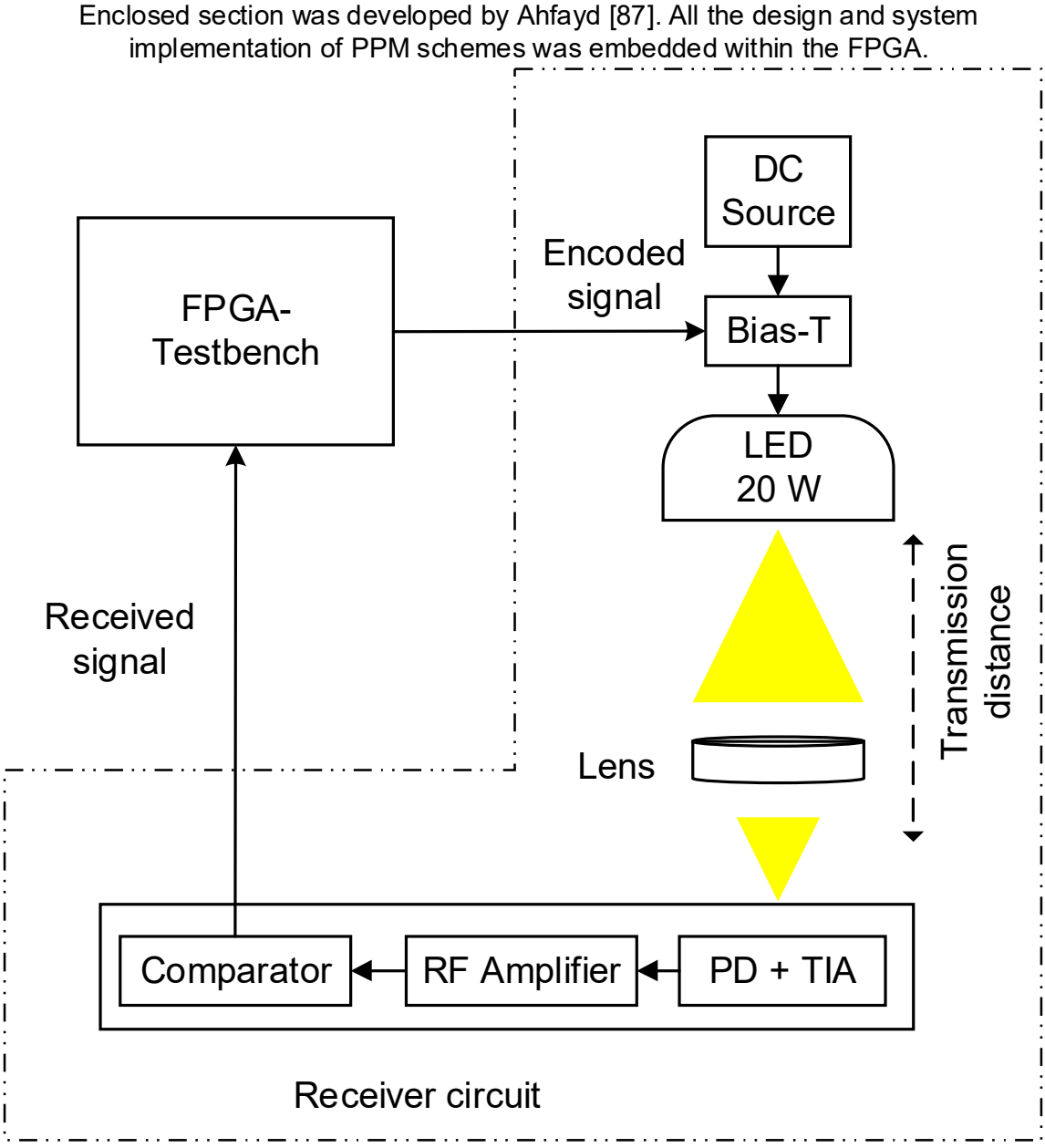


Figure 4-3 Experimental link used in this research

4.7 Receiver circuit design

The receiver circuitry used is given in figure 4-4, it consists of three stages, TIA, RF amplifier and a comparator. The PD and TIA were designed on a single PCB, the RF and comparator circuit were designed on a separate PCB.

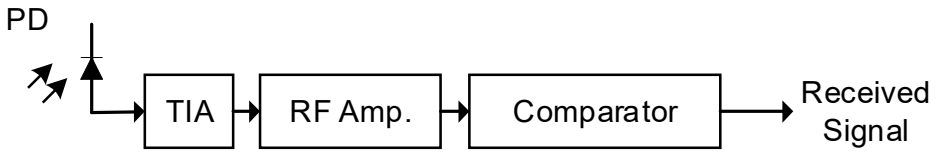


Figure 4-4 Receiver circuitry

4.7.1 Photodiode selection

A PD plays a key role in the VLC system, it converts optical energy to electrical energy. Selecting a PD for a VLC circuit requires a careful comparison between available diodes. Few things to be considered such as physical dimensions, active area, rise and fall times. For example, if the active area of the PD is smaller the focusing of the light on the PD becomes difficult. However, a smaller active area corresponds to having faster rise and fall times, as shown in table 4-I [88-90]. This means if high bandwidth is required a PD with a smaller active area must be used. However, having a smaller area means that less optical energy is reaching the PD. After considering rise/fall times and active areas from available PDs, OSD5-5T was used for experimentations in this research. Faster PD trade the active area to achieve higher rise and fall times, however, smaller areas capture a lesser amount of optical energy. The relation between area and rise and fall times is given in figure 4-5.

Table 4-I Different types of PD [88-90]

Parameters	**BPW34**	**OSD5-5T**	**FDS010**	**Units**
Wavelength range	400-1100	430-900	200-1100	nm
Rise/Fall time	20	9	1	ns
Active area	7.02	5	0.8	mm^2

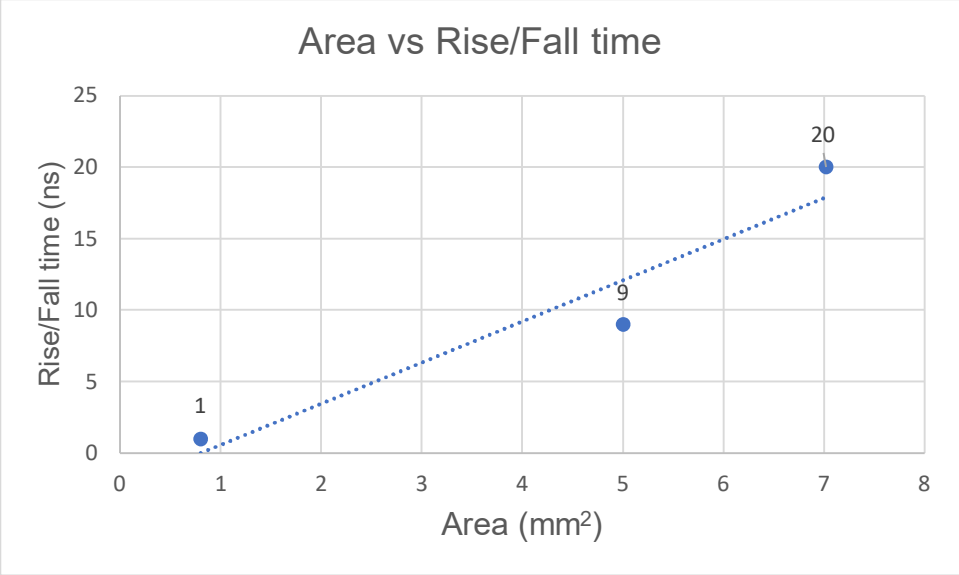


Figure 4-5 PD area vs rise and fall time comparison

4.7.2 Design of TIA

An emitter follower transimpedance amplifier was implemented and the circuit is given in figure 4-6. The transistors (BFR92A: Q1, Q2, Q3) used in the design have higher bandwidth with low noise. Q2 is set up as an amplifier with 20 dBm which is placed between two buffers.

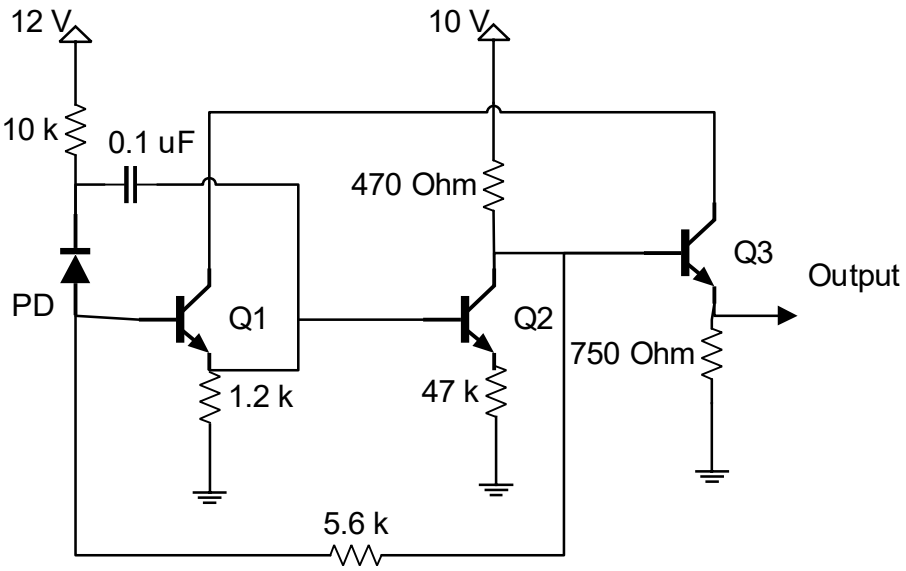


Figure 4-6 TIA circuit design

4.7.3 Compensator design

Compensators are used in many electronic circuits to regulate the undesired frequency response. There are three main types of compensators: phase lead, phase

lag and phase lead-lag [91]. The output of TIA is fed to the RC lead compensator in this experimental setup, as given in figure 4-7. TIA and compensator circuits were designed on PCB with an SMA connector (50 Ω). Compensator also plays an important role in extending the bandwidth of the system. The bandwidth extension can be seen in figure 4-8, approximately the system bandwidth was increased from 3 MHz to 40 MHz. However, the disadvantage of the compensator is that the amplitude of the signal is reduced.

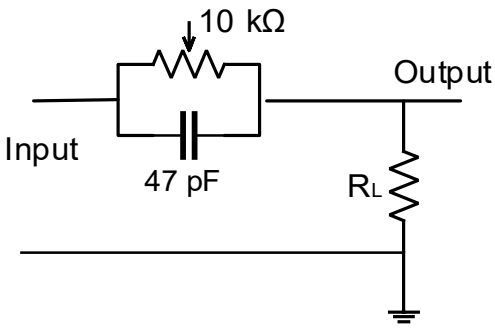


Figure 4-7 Lead compensator circuit design

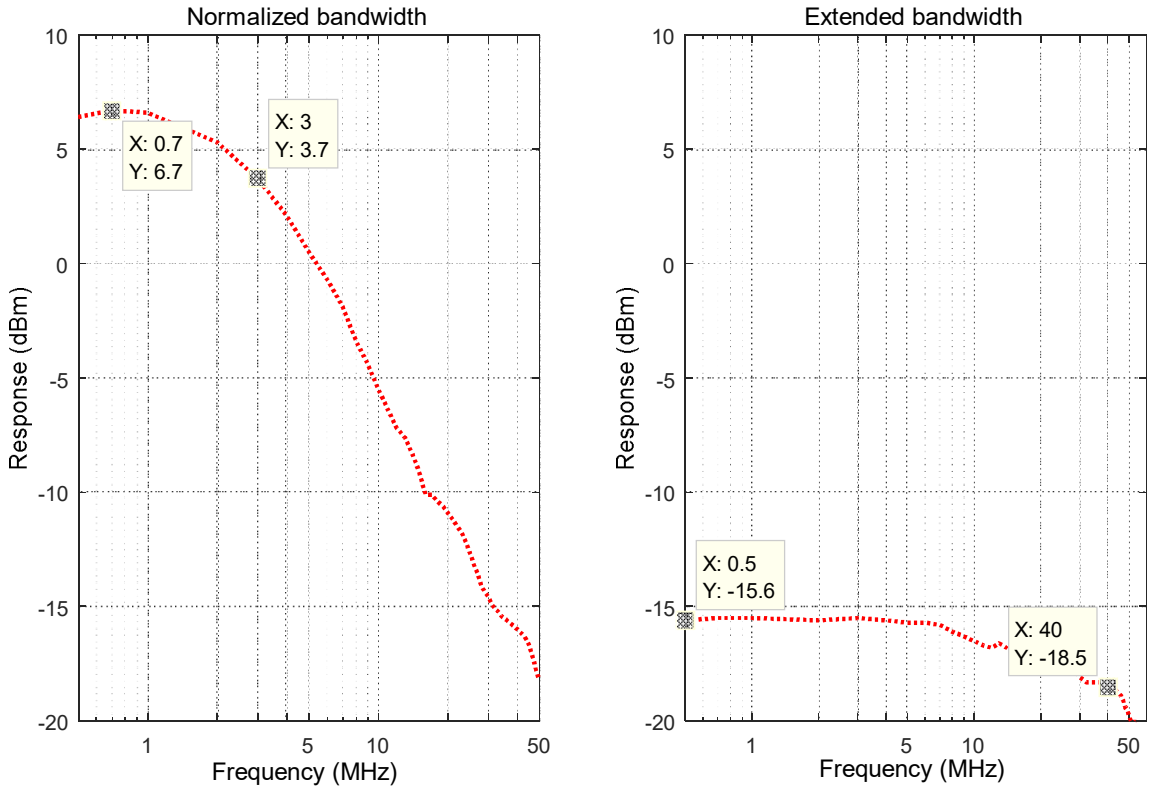


Figure 4-8 LED (20 W) response without (left plot) and with (right plot) compensator

4.7.4 Comparator circuit design

Comparator serves the purpose of digitizing the signal for FPGA. It compares the input signal with a reference signal and outputs a voltage between ground and Vdd. If the reference voltage is lower than the input voltage the comparator output is 0V and vice versa. Two comparators were available including LM393 and MAX942, the selection of comparator was carried out by comparing the rise and fall times [92, 93]. The rise and fall times of the comparators are given in table 4-II. MAX942 was used for experimentation as it offers faster rise and fall times, which enables the circuitry to achieve higher transfer rates. The comparator circuitry is given in figure 4-9.

Table 4-II Comparators specifications[92, 93]

Parameters	LM393	MAX942	Units
Current	400	350	uA
Rise/Fall time	1300	80	ns

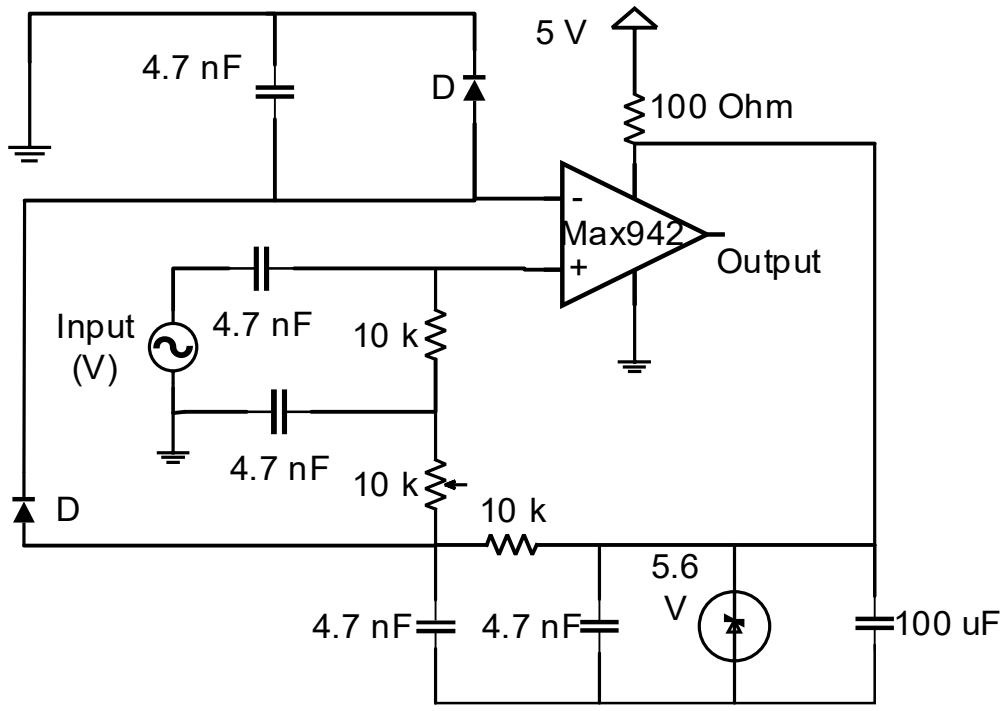


Figure 4-9 Comparator circuitry with Max942

4.7.5 Selection of LED

LED plays a vital role in a VLC system, as it acts as a transmitter. The LED used in this research was cool white 20 W 6000 K for reliable data transmission speeds of up to 20 MHz and to achieve a reasonable transmission distance of up to 3 m. Commercial LEDs are available in 10 W, 20 W and 50 W with different temperatures ranging from 3500 K to 7000 K. A detailed study was conducted to compare different commercially available LEDs for VLC applications [94]. It was shown that cool LEDs with a temperature range of 5800 K to 6000 K outperform warm LEDs with a temperature range of 2800 K to 3200 K, in terms of BER and have higher data transmission rates. In table 4-III, different LEDs are compared for BER and transmission distance. A warm 20 W LED achieved BER of 10^{-7} at a distance of only 1.35 m, however, 20 W cool LED achieved a higher distance of 1.9 m at 10 times lower BER of 10^{-8}. Due to the advantage of higher transmission rates and lower BER of cool LEDs, in this research cool LED was used for experimentations.

Table 4-III Comparison between different types of LEDs [94]

LED	BER at fixed 13 MHz (data rate)	Free space transmission distance (m)
20 W warm	$1.3x10^{-7}$	1.35
20 W cool	$8.2x10^{-8}$	1.9
30 W warm	$7.5x10^{-7}$	1.35
30 W cool	$8.2x10^{-8}$	1.9
50 W warm	$1.5x10^{-5}$	1.35
50 W cool	$8.2x10^{-8}$	1.35

4.8 FPGA testbench and HIL approach

Presented, for the first time, is an HIL approach to the design and evaluation of PPM modulation schemes. In the experimental setup, an FPGA was used to develop and implement a generic PPM testbench which enables the performance features of the different PPM schemes to be measured and evaluated. The testbench was

developed utilising an HIL approach, where the experimental parameters and tests configurations are controlled in order to enable direct performance comparisons of the PPM schemes within complex embedded systems. An additional benefit of the HIL approach is the facility to substitute elements of the experiment which are expensive and/or not available in the laboratory setting, with a controlled actuator based on the computer-simulated models [95]. The HIL experiment approachare is used to introduce known errors in the systems without the need for a physical link, as well as the facility to allow the user to measure the potential power consumption of a PPM scheme. HIL experiment structure was implemented on an FPGA which also contains a testbench model of multiple PPM schemes. The FPGA offers flexibility, reusability and reduced cost in terms of development for behavioural design of any embedded system.

A custom PCB was designed in this project to enable the connection between FPGA and the VLC circuitry, the details of the circuit are given in appendix B.

4.9 Summary

This chapter includes background information on VLC communications and how it has become an important area in the field of communications. Key components in VLC communications include LED, PD and receiver circuitry. The selection process of LED, PD is discussed since these parts play a vital role in reliable communication in a VLC system. The details of the experimental link used are included in this chapter. In conclusion, cool LEDs have better transmission capability and lead to faster data rates as compared with warm LEDs.

Chapter 5

Design and System Implementation of PPM schemes

5.1 Introduction

In this chapter, the designing and implementation of the test bench for different PPM schemes including DPPM, MPPM, OPPM, DiPPM and DuoPPM, is discussed. Additionally, to reduce errors CRC, parity check and MLSD are implemented for different schemes. This chapter also includes the synchronization and BER analysis of schemes implemented onto an FPGA, where a structured error induction process enables the performance of the PPM schemes to be evaluated without any physical channel or link required. A flow chart explaining the testbench development process is given in figure 5-1. The process was broken down into smaller tasks including designing encoder/decoder, synchronisation and BER techniques, designing of error correction and induction techniques and at last the full testbench development. Moreover, each scheme has its distinct features which means VHDL codes were not shared between schemes. Hence, designing and developing different PPM schemes requires careful consideration and separate designs. For example, OPPM has a dataword size of 3 bits, and MPPM has a dataword size of 4 bits. This means the designs used for OPPM cannot be used for MPPM, the design includes different modules such as encoder, decoder, synchronisation modules etc. Similarly, an error correction technique developed for one system cannot be shared with other schemes such as MLSD must be only used with DiPPM and DuoPPM.

5.2 A typical PPM scheme design

The FPGA development board modulation scheme implementation generates the encoded signal for transmission and receives the signal from the channel, a block diagram of the PPM design is given in figure 5-2. A PLL module is used in the design to generate the desired clock frequency, the output of PLL is connected to all the blocks. A Pseudo-Random Binary Sequence (PRBS) generator is used to generate appropriate random data, discussed later in section 5.2.1. The output of PRBS is fed to ‘PPM Encoder’, which converts the data into respective PPM format such as DPPM, MPPM formats etc. The output of the PPM encoder is connected to an XOR function, where the other input is received from the output of the ‘Error Inducing and BER calculation’ module, used to induce known error signals and also to determine the

output of the XOR gate. The output of the XOR function is referred as ‘Encoded signal’. This signal is used for transmission via Bias-T to LED which is part of the VLC system or the channel. The Received signal from the VLC system is fed back into the FPGA in the ‘PPM Decoder’ module. This module converts the signal into the original format or recovers the source PRBS signal. The output of the PPM decoder is transferred to the Synchronization module. Inside this module, an internal PRBS identical to source PRBS is available, the incoming signal is compared with internal PRBS to synchronise the signal. After synchronisation, BER is calculated by the Error module with the help of the synchronisation module. On other hand, a push-button is used with the Error Inducing module which allows the user to insert errors. The VHDL code and Quartus schematics for DPPM, MPPM74, OPPM, DiPPM and DuoPPM are given in appendix C, D, E, F and G respectively. The VHDL code for Modified MPPM64 is given in appendix H, it must be noted that the modified MPPM64 has an identical schematic to the original MPPM74.

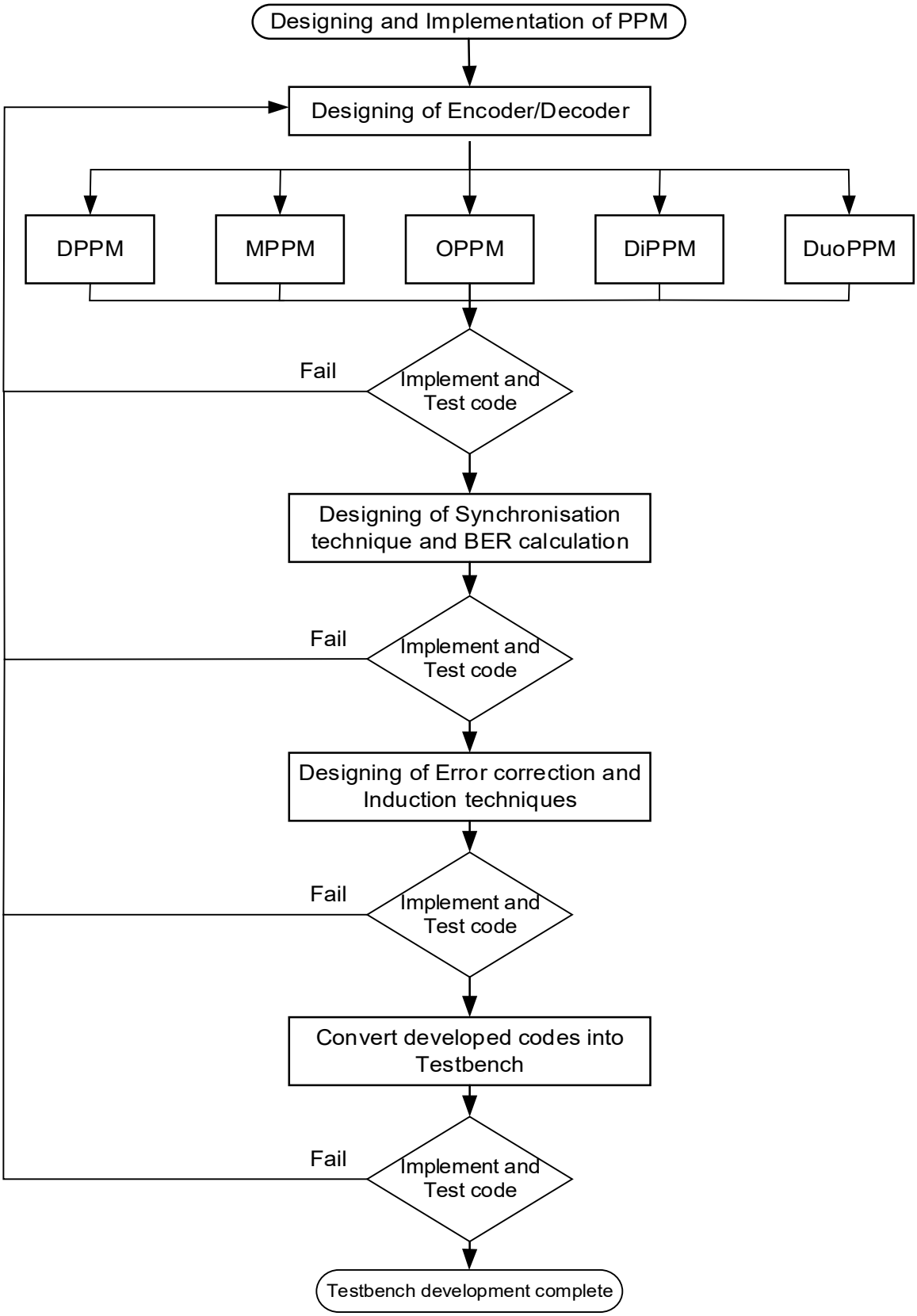


Figure 5-1 Flowchart of designing and system implementation of PPM schemes

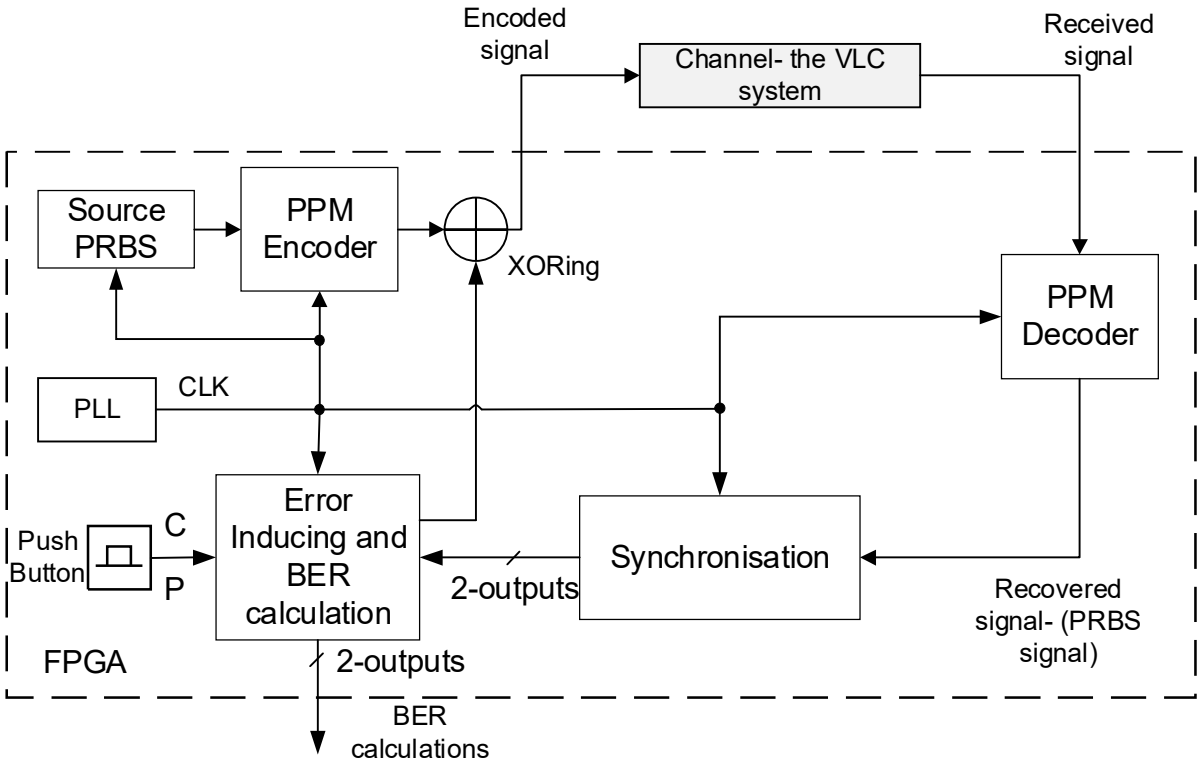


Figure 5-2 A typical PPM design for FPGA

5.2.1 PRBS module

The architectural block diagram of PRBS used is given in figure 5-3. The number n is reasonably large such as 15, 25, 50, to have larger repetitive sequences n must be kept as large as possible. In this design, the register is shifted right at every clock edge. The last two bits of the register are fed to an XOR function, and the output of the XOR function is fed back to the register resulting in appropriate random data. Initially, all the bits are set to 1 and after total sequences of 2^n, the register returns to its initial state and the cycle continues. The initial known state of the register enables synchronisation at the receiver to be achieved.

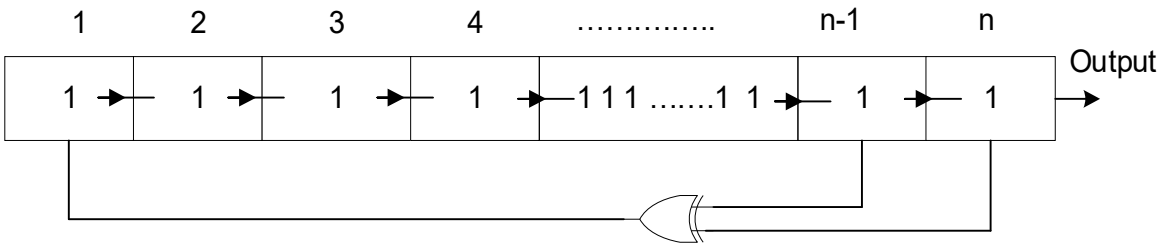


Figure 5-3 Block diagram of PRBS module

5.2.2 Synchronisation module

The synchronisation module has a control unit and two registers including an internal PRBS and an input register to store incoming data, as given in figure 5-4. The synchronisation module outputs two signals, the ‘Sync signal’ and the ‘Error_bits’. The Sync signal indicates that the synchronisation of incoming data from the decoder module has occurred and the ‘Error_bits’ indicates that a corrupted bit has been detected. Once the system is synchronised, as explained in section 5.2.3, the error module starts calculation of the system BER. It must be noted, the system requires initialisation, as synchronisation of the input signal must occur before identification of valid received signals, thus enabling the detection of any error or corrupted bits. These two signals ‘Error_bits’ and ‘Sync signal’ and are transferred to the Error Inducing and BER calculation module. The control unit continuously monitors both registers where the contents of both should be identical.

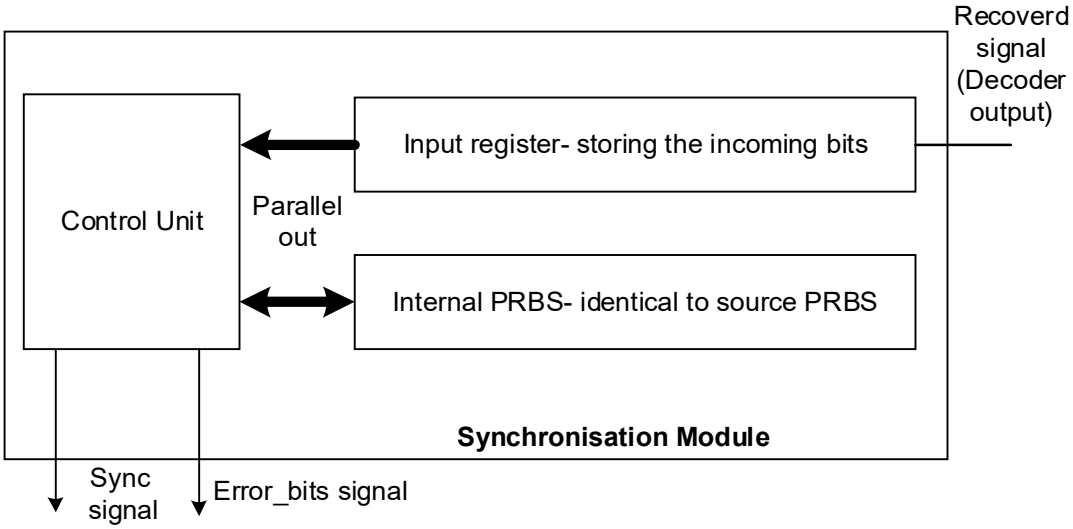


Figure 5-4 Block diagram of the synchronisation module

5.2.3 Error Inducing and BER calculation module

This module has two control units, as given in figure 5-5. Control unit 1 is used to calculate the number of corrupted bits via the counter 1 block. However, the control unit 2 is used to calculate the number of known errors induced in the system via the counter 2 block. Error inducing is controlled with a push-button, when the button is pushed, the input to XOR is set high to insert errors at an appropriate selected random sequence using the internal PRBS register. Counter 2 is used to record the total number of errors induced.

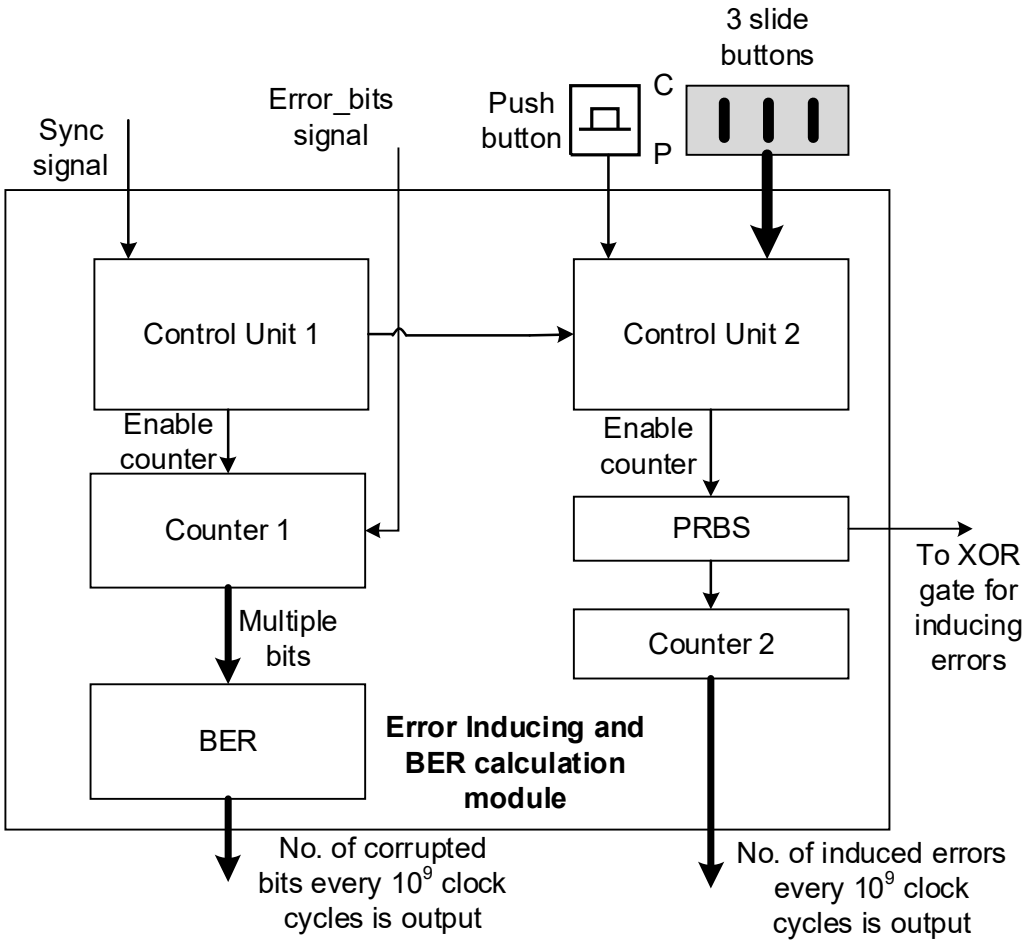


Figure 5-5 Block diagram of the Error Inducing and BER calculation module

5.3 Full Testbench Design and Implementation on FPGA

The block diagram of the full testbench design is given in figure 5-6. The testbench utilised multiple multiplexers to enable the correct set-up configuration for the selected PPM scheme. The given testbench configuration utilises only two PRBS sequence lengths; 15-bit and 16-bit register sizes, which are shared across different modules. The schemes with a dataword size of 3-bit use 15-bit PRBS and the remaining are connected with 16-bit PRBS. The PRBS sequence length can be adapted if required; provided the length of the PRBS is multiple of dataword size. Similarly, only two synchronisation modules are used with 15-bit and 16-bit internal PRBS registers. All schemes have dedicated Error Inducing and BER calculation modules due to different symbol sizes, except the DuoPPM and DiPPM which share identical features in modulation/demodulation of the signal.

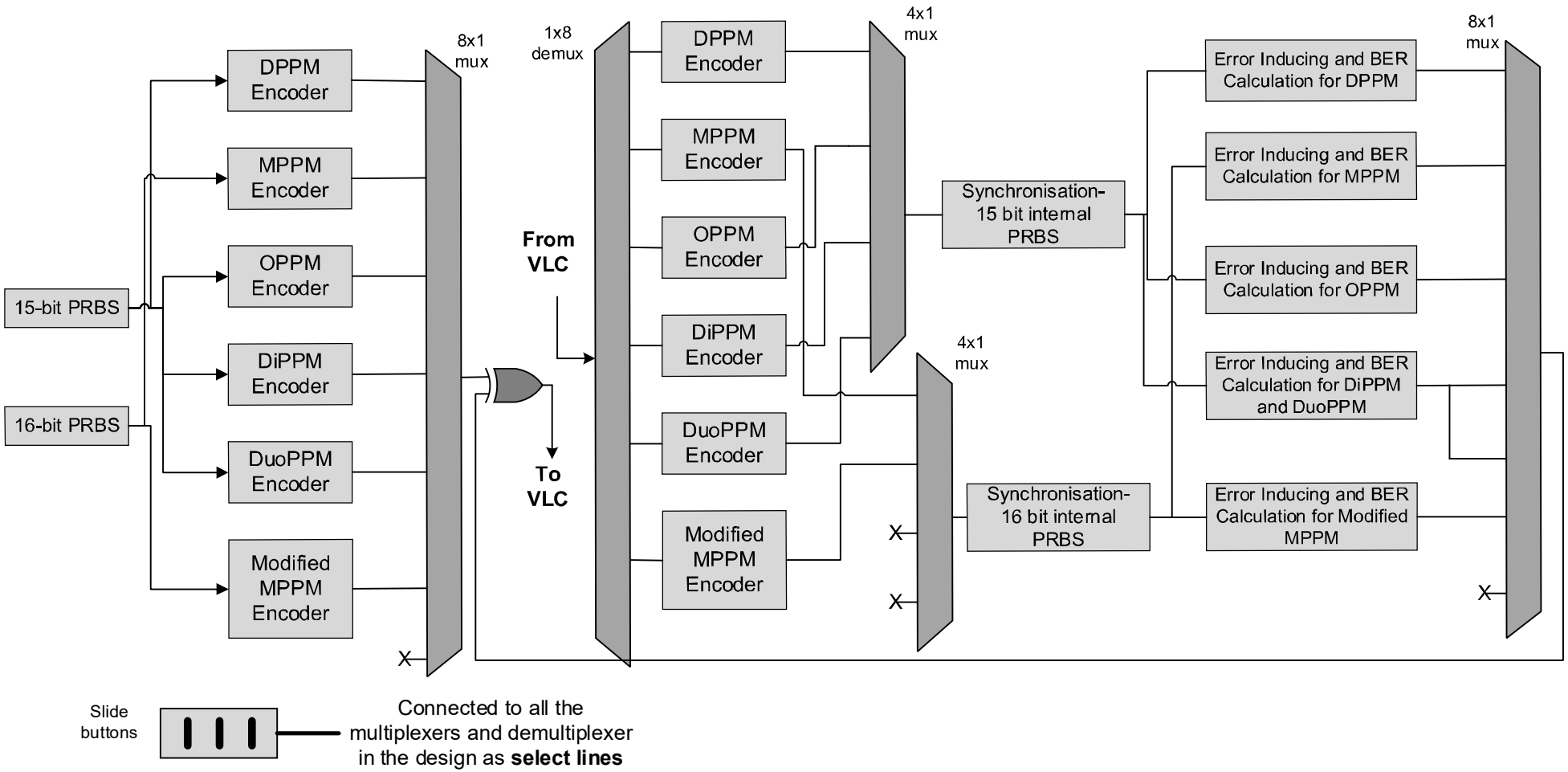


Figure 5-6 Block diagram of developed full Testbench

5.4 Design and Implementation of Priority Decoder for OPPM

A priority decoder method was proposed in this thesis to enhance the performance of OPPM by improving the BER of the system. Figure 5.7 shows the design of the priority decoder. A 4-bit input register is used to inspect the received data for any errors. If an invalid codeword is identified by the control unit, it instructs the combinational logic module to amend the register contents. After the inspection and amendments are completed, the data from the register is transferred to the original OPPM decoder. The original OPPM decoder recovers the message from the received inspected or amended bits. The output of the priority decoder is transferred to the synchronisation module for synchronisation of the system. By using simple logic gates and without any bandwidth reduction priority decoder can be implemented and the BER of the system can be improved. The VHDL code of the OPPM decoder is given in appendix I.

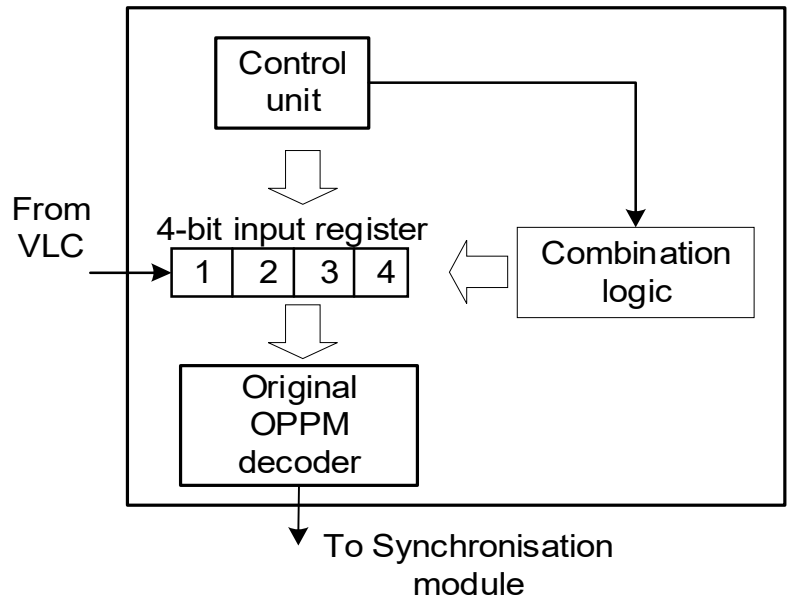


Figure 5-7 Design and Implementation of OPPM Priority Decoder

5.5 Design and Implementation of CRC and Parity check with Modified MPPM64 on FPGA

To correct errors Modified MPPM64 was integrated with CRC and parity check. The design and implementation are given in figure 5-8. The checksum calculation part was integrated into the encoder, the design takes 6 consecutive datawords from PRBS and calculates the checksum and places it at the end. Having 6 codewords and a checksum increases the bandwidth of MMPM64 to the equivalent of MPPM74. The

number of codewords can be revised to improve the bandwidth utilization of the system. On reception of the signal from VLC, a parity check is performed first and a flag is raised if odd parity was identified since the parity of the system will always be even. Moreover, the data passes through the MPPM decoder, the output of the decoder is transferred for checksum inspection. If a flag was raised for a certain codeword during parity check, that codeword is replaced with the checksum to correct the flagged codeword. The VHDL code of Modified MPPM integrated with CRC and Parity check is given in appendix J.

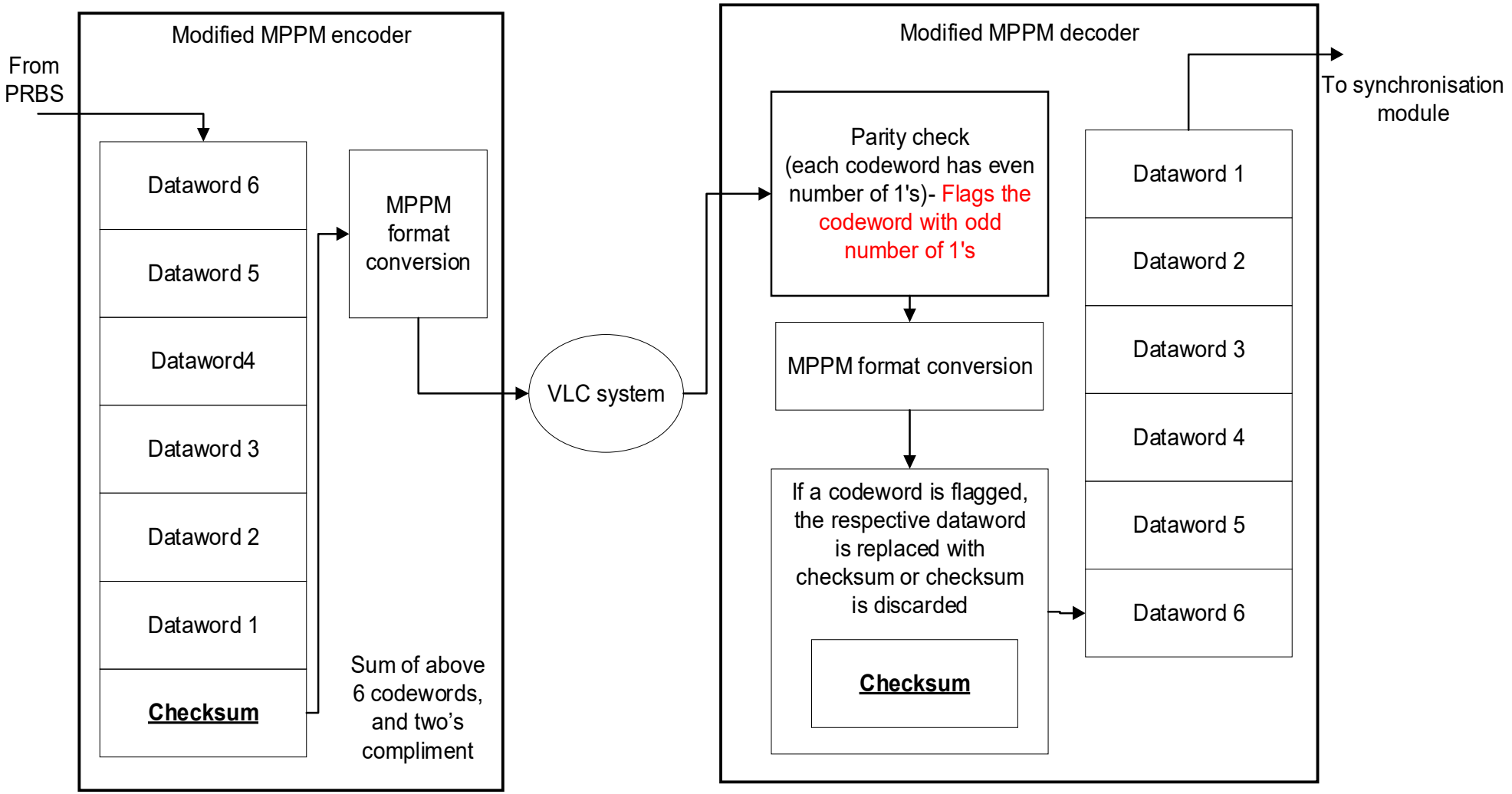


Figure 5-8 Design and Implementation of CRC and Parity check with Modified MPPM

5.6 Design and Implementation of MLSD decoder

In the MLSD decoder, the decision to alter the sequence for error correction is made depending upon the location of 1's and 0's or set (S) and reset (R) pulses, respectively. In DiPPM and DuoPPM, set and reset pulses are followed by each other. The received signal from the VLC is transferred into the control unit, as given in figure 5-9. The control unit fills both 30-bit registers depending on the received value i.e., S or R pulse. The size of the register must be greater than the size of PRBS to accommodate at least 3 pulses for correct analysis. Furthermore, while filling up the registers the control unit is continuously monitoring both registers. If the pattern of S and R pulses do not match the control unit inserts a respective bit or reverses a bit to correct the pattern, as shown in table 5-I. For example, if three consecutive S pulses are received pulses, the MLSD recognises the incorrect pattern received and will alter the middle pulse to R pulse. Similarly, if SRR is received as three consecutive pulses, a S pulse will be placed in the middle of two R pulses creating a new pattern of SRSR. The MLSD VHDL is given in appendix K.

Table 5-I MLSD control and correction technique

Scenario	MLSD correction	Error Detection & Correction
S**S**S	S**R**S	Detected + Correction
SSR	S**R**SR	Detected + Correction
SRS	SRS	No Error
S**RR**	SR**S**R	Detected + Correction
R**SS**	RS**R**S	Detected + Correction
RSR	RSR	No Error
RRS	R**S**RS	Detected + Correction
R**R**R	R**S**R	Detected + Correction

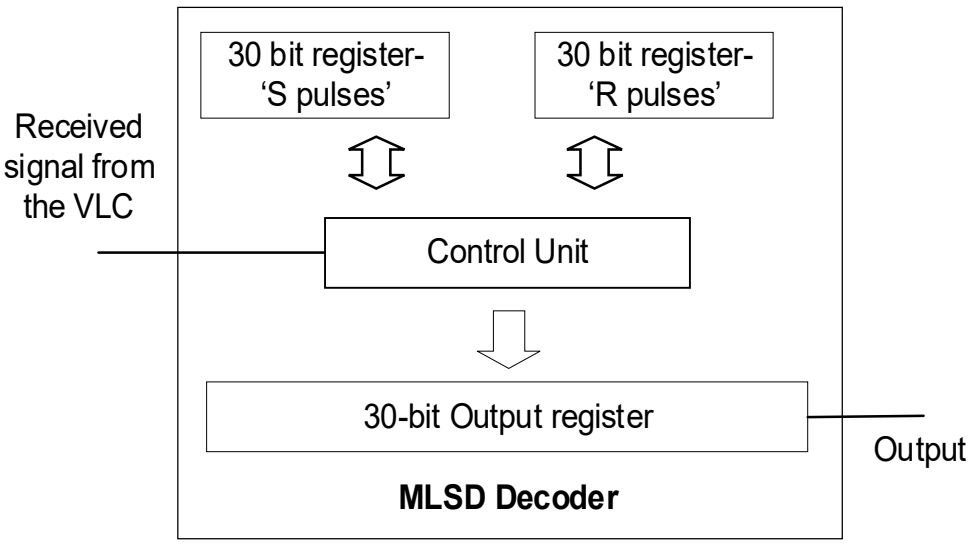


Figure 5-9 Design and Implementation of MLSD Decoder

5.7 Design and Implementation for power estimation of any given PPM

To measure the power consumption of any given PPM, a simple VHDL code was developed. Since power is proportional to voltage and current, using voltage or current relative power can be calculated. The higher number of 1's represent higher power consumption which requires higher optical power for transmission when compared with logic 0. Moreover, when logic '1' is passed to bias-T, it combines logic 1 with DC voltage level for LED as input. This means when logic 0 is passed to bias-T it will require lower power. A counter was designed to be placed before transferring the signal to LED for transmission. A higher number of 1's represents relatively higher power consumption. The design of the counter is shown in figure 5-10. This new module does not alter the signal being transmitted, therefore the signal characteristics remain unchanged. The VHDL code to estimate power is given in appendix L.

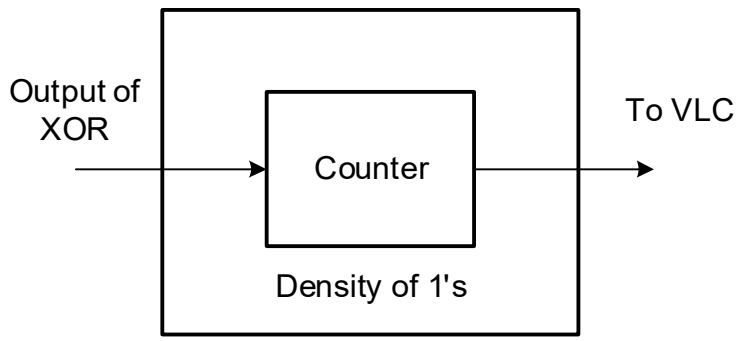


Figure 5-10 Design and Implementation of Power Estimation for given PPM

5.8 Development kit and Programming language

The Altera DE2 Development and Education [96] was used for the Implementation of PPM schemes. The Intel Quartus Prime software [97], earlier versions called Altera Quartus Prime, was used for designing the digital logic using VHDL. The specifications of the DE2 board are given in appendix M.

An FPGA, abbreviation of Field Programmable Gate Array, is a device or chip with programmable internal logic. Therefore, the user can change the internal logic after the chip has been manufactured. The programming language used to program FPGA is called Hardware description language (HDL) [98, 99]. The main purpose of using an FPGA is flexibility, reusability and parallelism i.e., sending and receiving the data simultaneously. This cannot be achieved by using a simple microprocessor as they are designed to run the program sequentially.

The error inducing and power estimation modules were designed for HIL experimentations, in order to induce know errors without the requirement for experimental links and also to estimate power consumption. In possible future development, this HIL approach can further be used to introduce desired BER, where a GUI can be used to assign values and set variables for the HIL experiments. A simple web application based GUI hosted on a System-on-Chip FPGA can be used for changing the parameters of the HIL experimentations. Furthermore, more features can be added to evaluate the configuration of LOS and non-line of site experimentation for VLC system.

5.9 Summary

This chapter explains the development and design of a typical PPM scheme. Using the typical PPM design the development of DPPM, OPPM, MPPM, DiPPM, DuoPPM and Modified MPPM64 schemes was carried out. Furthermore, using all the developed codes, a full testbench was developed which enables the user to select any schemes with the help of three slide buttons on the FPGA board. This testbench can be used to evaluate the performance features of the coding schemes. Along with performance features, user can also find the optimal scheme by comparing the given schemes for any given VLC system.

Chapter 6
Results and Discussions

6.1 Introduction

In this chapter, the experimentation and results of practical implementation of different PPM schemes on an FPGA and VLC system are discussed. Along with implementation, the analysis of the results was included. The comparison and detailed analysis were carried out to evaluate the performance of each modulation scheme against different criteria. Two novel PPM schemes have been presented to enhance the performance of MPPM and OPPM schemes. It was shown that MLSD can reduce the BER of the coding schemes. The experimental link limitations have been identified in this chapter which can further improve the performance of the system.

6.2 System testbench

The full system testbench experimental setup used 20 W cool LED with a temperature of 6000 K. To measure the BER the experiment was conducted in two ways. One experiment was conducted by fixing the line rate at 14 Mbps and varying the transmission distance between 1.5 m and 3 m, to calculate the BER. The second experiment was conducted by fixing the transmission distance at 1.5 m and varying the line rate between 13 Mbps and 18 Mbps, to calculate the BER. For indoor applications, the transmission distance of 1.5 m to 3 m offers reasonable height. Furthermore, the line rate of 13 Mbps to 18 Mbps offer reasonable speeds with the appropriate given setup. The cool LED was used with is often used for indoor lighting in offices and homes.

In figure 6-1, the line rate speed was fixed at 14 Mbps for all modulation schemes and transmission distance between LED and PD was varied to calculate BER. DPPM outperforms all in BER comparison with BER of less than 10^{-7} at 2.9 m. On other hand, DiPPM shows the least performance with a BER of less than 10^{-7} at a transmission distance of only 1.5 m. DuoPPM performs better than DiPPM by achieving a transmission distance of 0.3 m higher at BER of 10^{-7}. The OPPM performs intermediary when compared with all the other PPM schemes by achieving a BER of 10^{-8} at a distance of 2.3 m. The original MPPM (MPPM74) outperforms the modified MPPM (MPPM64) in terms of BER, however, it must be noted that MPPM64 has improved bandwidth utilization by 14.58% as compared to the original MPPM. In

simple terms, the MPPM64 is transferring higher data as compared to the modified MPPM. Table 6-1 shows the data rate for each PPM scheme for the experiment conducted in Figure 6-1.

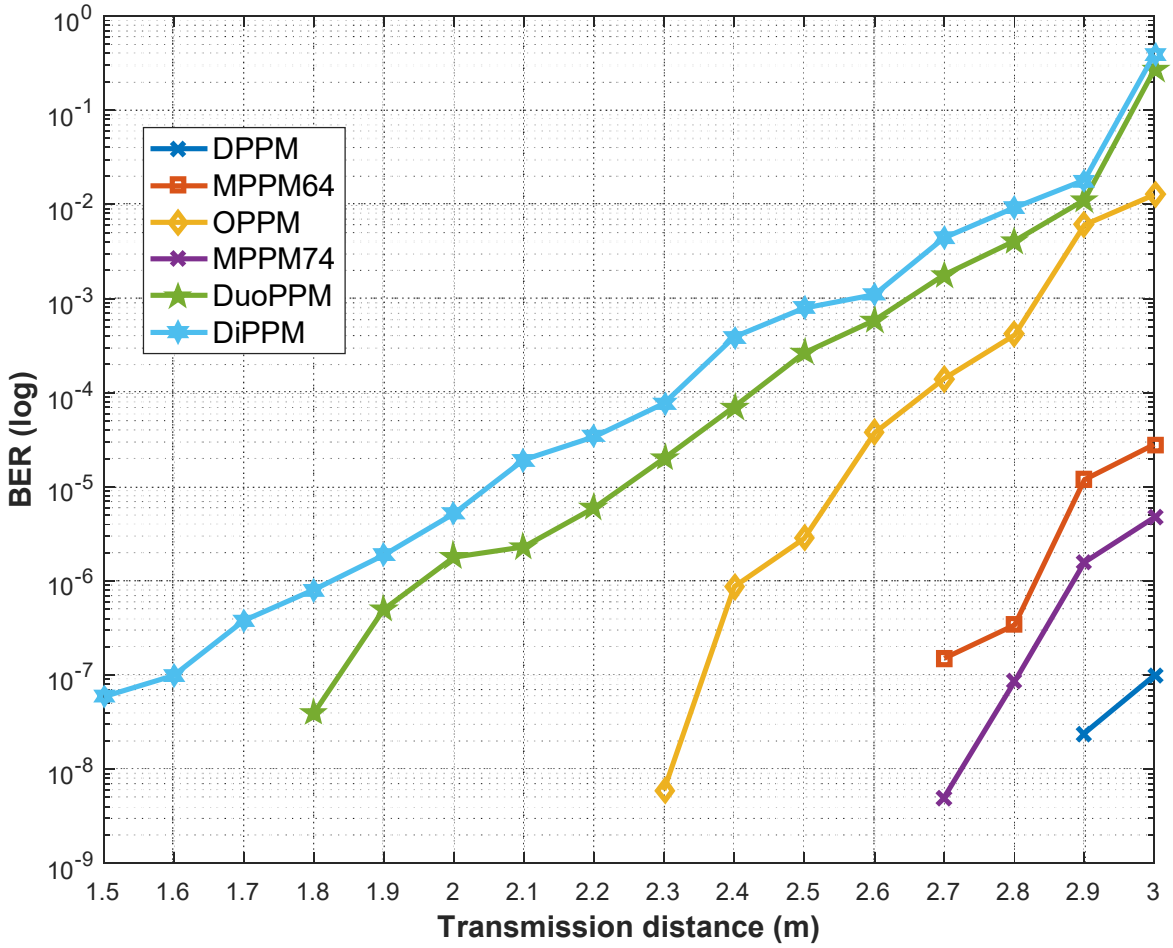


Figure 6-1 BER vs Distance measurements using Full testbench setup

Table 6-I PRBS rates for fixed line rate

Modulation scheme	Line rate (Mbps)	Data/PRBS rate (Mbps)
DPPM	14	5.25
MPPM74	14	7.99
MPPM64	14	9.33
OPPM	14	10.5
DiPPM	14	14
DuoPPM	14	14

Similarly, in figure 6-2, DPPM outperforms all by achieving BER of 10^{-6} at the speed of 17 Mbps and DiPPM performs worse at BER of less than 10^{-7} at the speed of 13 Mbps. DuoPPM shows relatively better performance by achieving a BER of 10^{-8} which is 10 times lower than DiPPM. MPPM64, MPPM74, OPPM and DPPM show similar BER of $10^{-}6$ at 17 Mbps, however, at 18 Mbps the BER of MPPM64 and OPPM drastically increases to 10^{-3} and 10^{-4}, respectively

Using table 6-II, DPPM shows the lowest power with the highest bandwidth expansion which means at the expanse of large bandwidth expansion DPPM is offering the lowest power. DPPM also shows the normalized bandwidth of 1.0 which is almost two times of OPPM bandwidth. DuoPPM and DiPPM show the second lowest power estimation of 0.8 and 0.79, respectively. However, both schemes offer the lowest bandwidth expansion with a normalized bandwidth expansion of 0.37. MPPM74 performs worse with normalized power estimation 0.9 when compared with DPPM, DiPPM and DuoPPM, however, it performs slightly better than MPPM64 and OPPM with both schemes offering normalized bandwidth of 1.0. MPPM64 offers lower bandwidth expansion by 14.28% when compared with MPPM74.

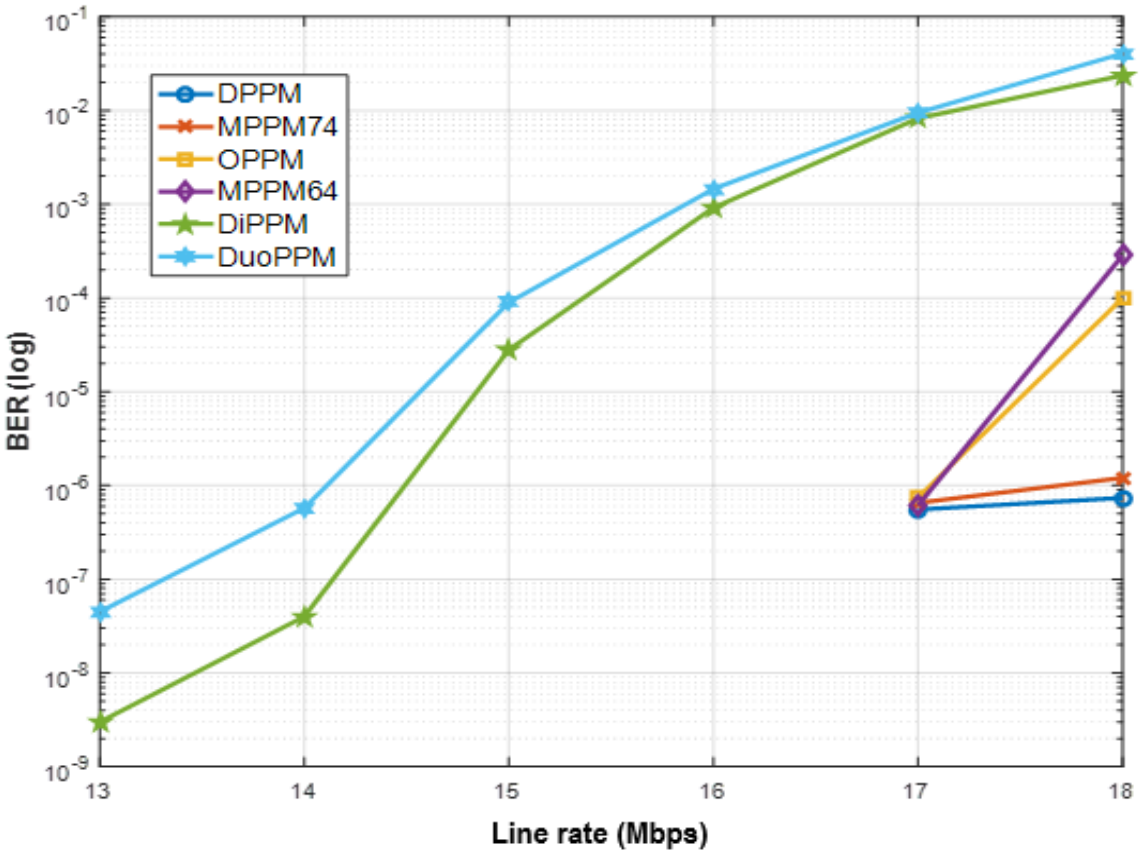


Figure 6-2 BER vs Data Rate measurements using full testbench setup

Table 6-II Power and bandwidth comparisons of different PPM schemes

Modulation scheme	Power estimation- Normalised Density of 1's	Normalised Bandwidth expansion
DPPM	0.4	1.0
MPPM74	0.9	0.65
MPPM64	1.0	0.56
OPPM	1.0	0.49
DiPPM	0.79	0.37
DuoPPM	0.8	0.37

6.3 Priority decoding OPPM

Figure 6-3 shows the BER comparison between original OPPM and Priority Decoding OPPM over the VLC system. Both modulation schemes were tested under identical circumstances with and without VLC i.e., identical transmission distance, LED and transmission speed [100]. A reasonable transmission speed of 14 Mbps was chosen to achieve reasonable transmission distance i.e., 1.4 m to 1.8 m. Higher transmission speed or longer distances reduce the BER of the system. At lower transmission distances the difference in BER of both PPM schemes is significant and as the transmission distance increases the difference in BER reduces. Priority decoding recorded almost 10 times fewer errors with a BER of 10^{-8}. At higher transmission distances less optical power reaches the PD and the system suffers from erasure errors. Priority decoding is ineffective in tackling erasures. If an erasure error occurs in a valid codeword, it becomes another valid codeword and the priority decoder is unable to detect an error in a valid code, which is a limitation of priority decoding.

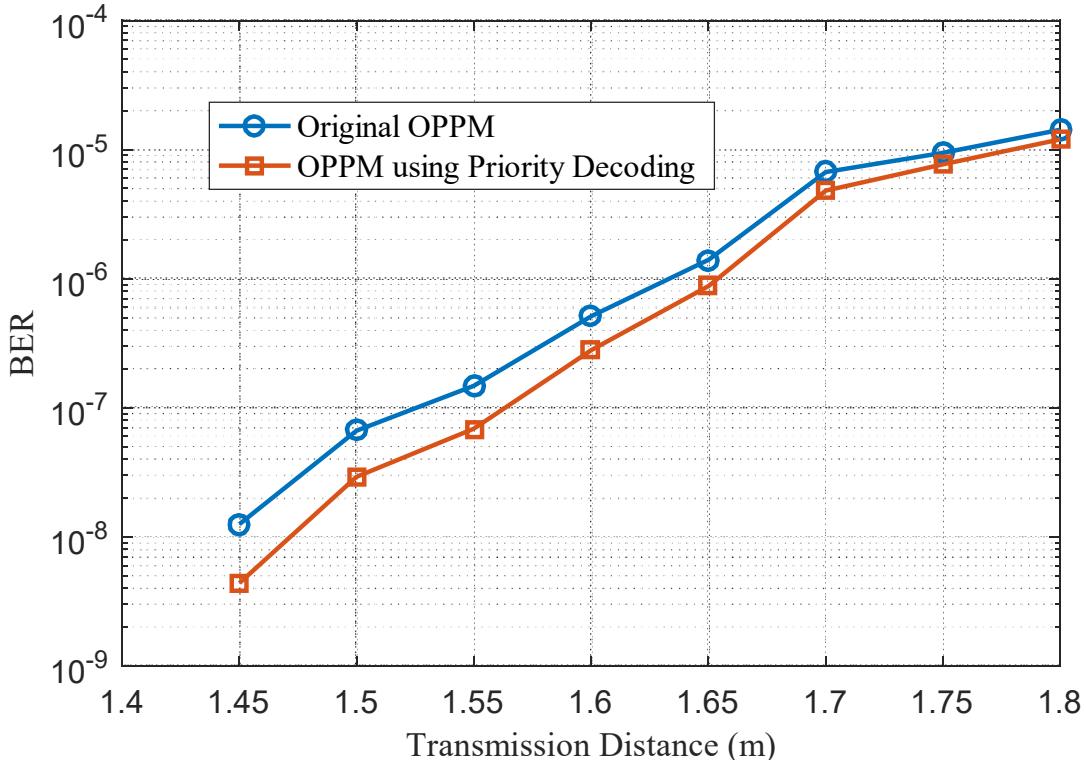


Figure 6-3 OPPM and OPPM Priority Decoding BER comparison [100]

The priority decoding was tested without the VLC circuitry, and the results are presented in the ratio of detected errors to induced errors. A single corrupted bit can result in multiple errors in some cases. Using error inducing module, known errors were induced in different locations of the codeword to check the performance of priority decoding. The location of corrupted bits determine the decoder output and hence the total number of errors. Using table 6-III, when an error occurs at MSB it does not affect the priority decoder output as it remains a valid codeword. However, when an error occurs at other bit locations .i.e., 2nd and 4th the output of the priority decoder will outperform the original OPPM. However, when an error occurs at bit-3, the priority decoding performs worse than the original OPPM. The overall probability of error in Priority decoding is 0.86475, on other hand, the original OPPM has a higher probability of 1.1875.

Figure 6-4 shows the working of OPPM with priority decoder. The clock signal is the main reference clock, the PRBS signal is used as data, the encoded signal is transferred to LED and the decoded signal is the incoming data from the receiver circuitry. As it can be seen the transmitted codeword was corrected to 0010 from 0011, resulting in BER improvement.

Table 6-III OPPM vs OPPM with Priority Decoding

	$Ratio = \frac{Detected\ error}{Induced\ error}$				Overall probability of error
	1st (MSB)	2nd	3rd	4th (LSB)	
Original OPPM	1	1.75	1	1	1.1875
OPPM with Priority decoding	1	0.71	1.25	0.499	0.86475

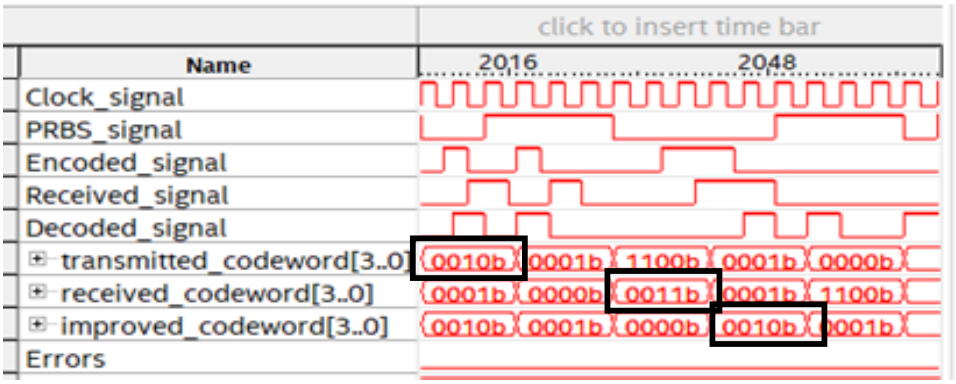


Figure 6-4 Priority decoding OPPM error corrections

6.4 Modified MPPM

The modified MPPM64 offers the benefit of reduced bandwidth expansion. In figure 6-5, the BER comparison between original and modified MPPM has been presented. Both schemes were tested under identical conditions for accurate comparison and analysis. The data rate was fixed at 9 Mbps for both schemes to demonstrate the bandwidth expansion and how it affects the overall system performance. As illustrated in figure 6.1 the MPPM64 BER is higher than that achieved by MPPM74 for a fixed line rate of 14 MHz. However, it must be noted that MPPM64 offers a 14.28% improvement in bandwidth utilisation. Furthermore, the modulation selection method, as discussed in section 7.1, shows that MPPM64 outperforms when the weighing criteria are higher for bandwidth utilisation. Modified MPPM64 offers the same data rate at a lower line rate which means to transmit an equal amount of data higher line rate is required in the original MPPM74. In table 6-IV, the original MPPM74 requires a line rate of 15.75 Mbps and modified MPPM64 requires a line rate of 13.5 Mbps. The modified MPPM reduces the line rate of the system by 14.28% since it has a shorter codeword. Both schemes offer BER of less than 10^{-7} at a data rate of 9 Mbps,

however, at different transmission distances. Modified MPPM offers a 0.4 m transmission distance advantage over original MPPM at BER of less than 10^{-7}.

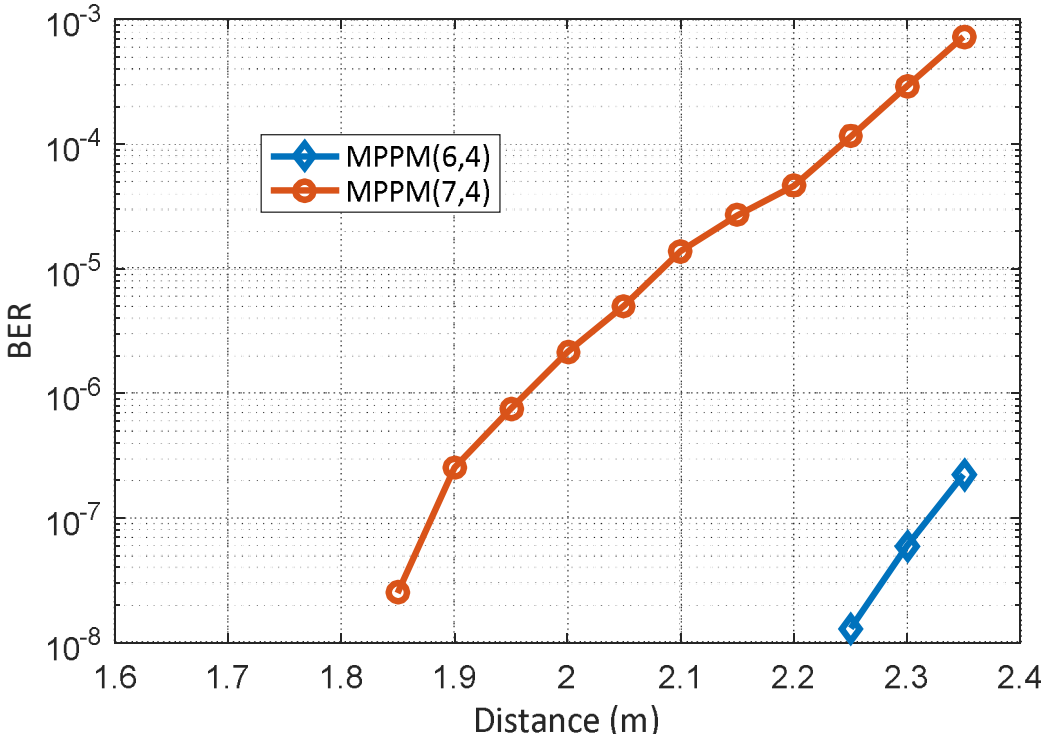


Figure 6-5 MPPM vs Modified MPPM BER comparison

Table 6-IV Line rates of original and modified MPPM

Coding Scheme	Data Rate (Mbps)	Line Rate (Mbps)
Modified MPPM (6,4)	9	13.5
Original MPPM (7,4)	9	15.75

Another test was conducted by artificially inducing errors in each bit location of both schemes, the results are given in table 6-V. As it can be seen that ratio of detected errors to induced errors is higher in the original MPPM74. The main reason behind the modified MPPM64 improved ratio is half the number of possible codeword combinations when compared with MPPM74. MPPM64 has 64 possible codeword combination when compared with the original MPPM74 which has 128 possible combinations. The overall probability of error in the original MPPM74 is 1.7142 as compared to 1.40145 in modified MPPM64, an improvement of 18.24 %.

Table 6-V Original and Modified MPPM comparison

	$ratio = \frac{detected\ errors}{induced\ errors}$								Overall probability of error
	Bit7 (MSB)	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0 (LSB)**	
MPPM 74	1.5018	1.498	1.5	1.5003	1.5	1.5	1.5003	1.499	1.7142
MPPM 64	N/A	1.2007	1.204	1.206	1.2001	1.2001	1.1988	1.199	1.40145

6.4.1 Modified MPPM with CRC and Parity check

Modified MPPM was integrated with a simple CRC and parity check and the BER is given in figure 6-6. Since the checksum is appended with codewords transmitted, this means increased bandwidth expansion in MPPM64. CRC and parity check implementation results in a higher line rate to transmit an equal amount of data. Both schemes achieved BER of less than 10^{-7}, at the identical line and data rates. However, MPPM64 with error correction outperformed the original MPPM74 by achieving a higher transmission distance of 0.2 m, an improvement of 17.78 %. In simple terms, MPPM64 with error correction achieved a transmission distance of 2.25 m with BER of 10^{-7}, however, original MPPM74 achieved a similar BER at 1.85 m. From table 6-VI, the line and data rates of both schemes are identical and represents that transmitting an equal amount of data. CRC and parity check can offer advantages of improved BER or transmission distance. As the transmission distance increase, the BER of both schemes also increases due to light dispersion and low optical power being received at the receiver. The working of CRC and parity check is shown in figure 6-7, the highlighted box shows three different codewords with 0111b being the CRC check. On the receiver side, the CRC check is ignored and data is kept constant for another cycle. This experiment proves that simple error correction techniques can significantly improve the BER of PPM schemes at identical speeds.

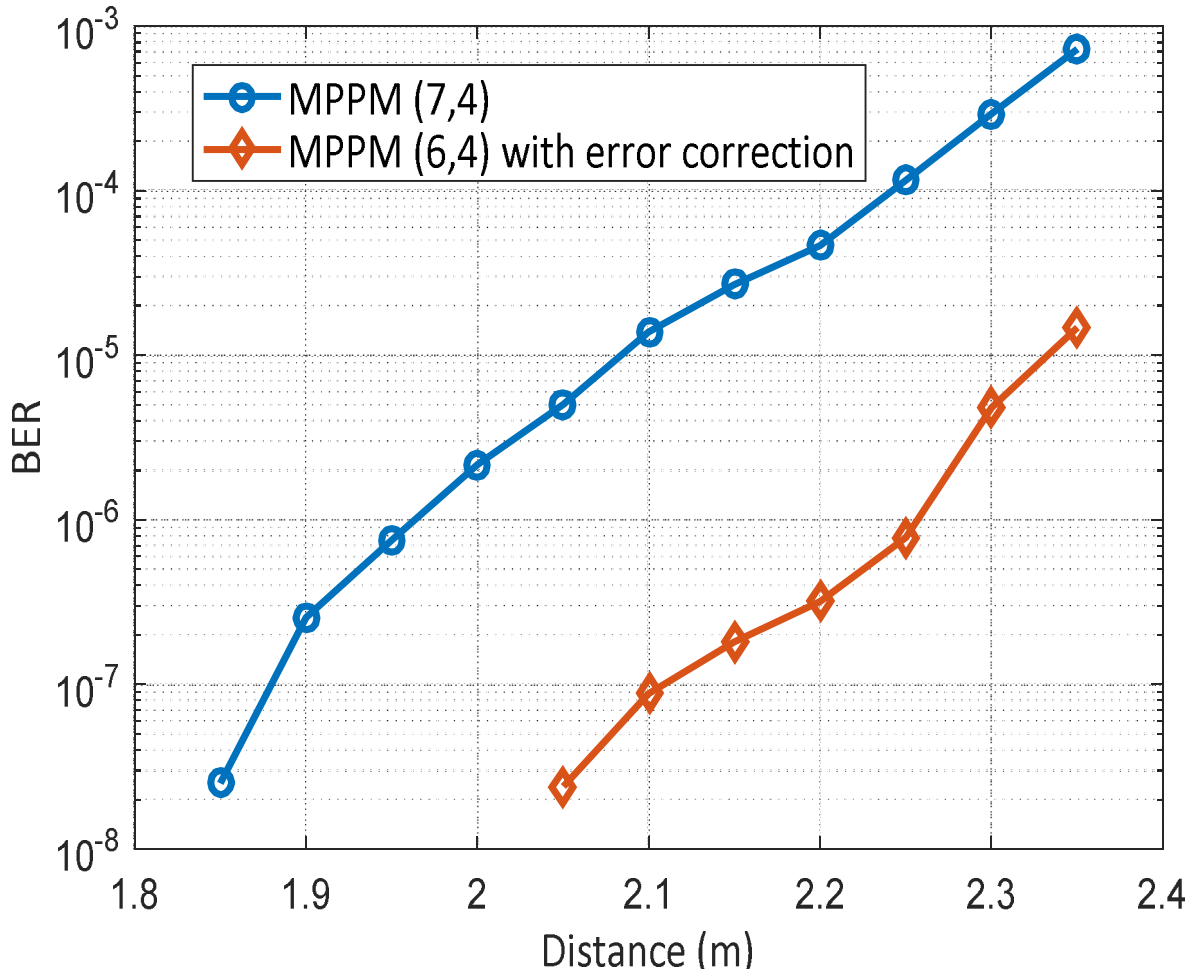


Figure 6-6 Modified MPPM with CRC and Parity check

Table 6-VI Modified MPPM with CRC and parity check line and data rates

Coding Scheme	Data Rate (Mbps)	Line Rate (Mbps)
MPPM74	9	15.75
MPPM64 with checksum and parity check	9	15.75

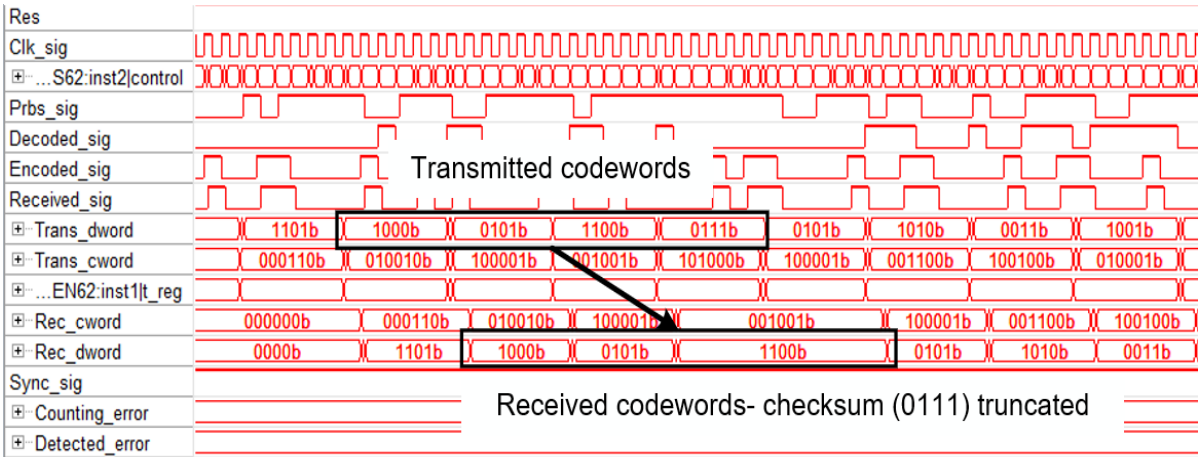


Figure 6-7 Signal simulations of MPPM with CRC and parity check

6.5 MLSD

Two experiments were conducted to prove the eligibility of MLSD codes. One experiment was conducted on the VLC link and the other was conducted without the VLC link. Figures 6-8 and 6-9 show the BER comparisons of DiPPM and DuoPPM with MLSD, respectively. The line rate was fixed at 14 Mbps with a transmission distance of 1.5 m to 3.0 m. In both cases, the MLSD outperformed original schemes at lower distances. DiPPM and DuoPPM achieved BER of around 10^{-7} at a distance of 1.5 m and MLSD achieved BER of 10^{-8}, offering almost 10 times lower BER. As the transmission distance increases above 2.5 m, the original schemes achieve similar or lower BER than the MLSD. This is due to excessive errors and less optical power reaching PD, where MLSD struggles to correct errors thus performing worse than the original. From table 6-VII, theoretical results show MLSD schemes outperform the original schemes by correcting up to 40 % and 37.4 % errors in DiPPM and DuoPPM, respectively. The errors were inserted artificially using error inducing module and the ratio of detected error to induced errors was calculated.

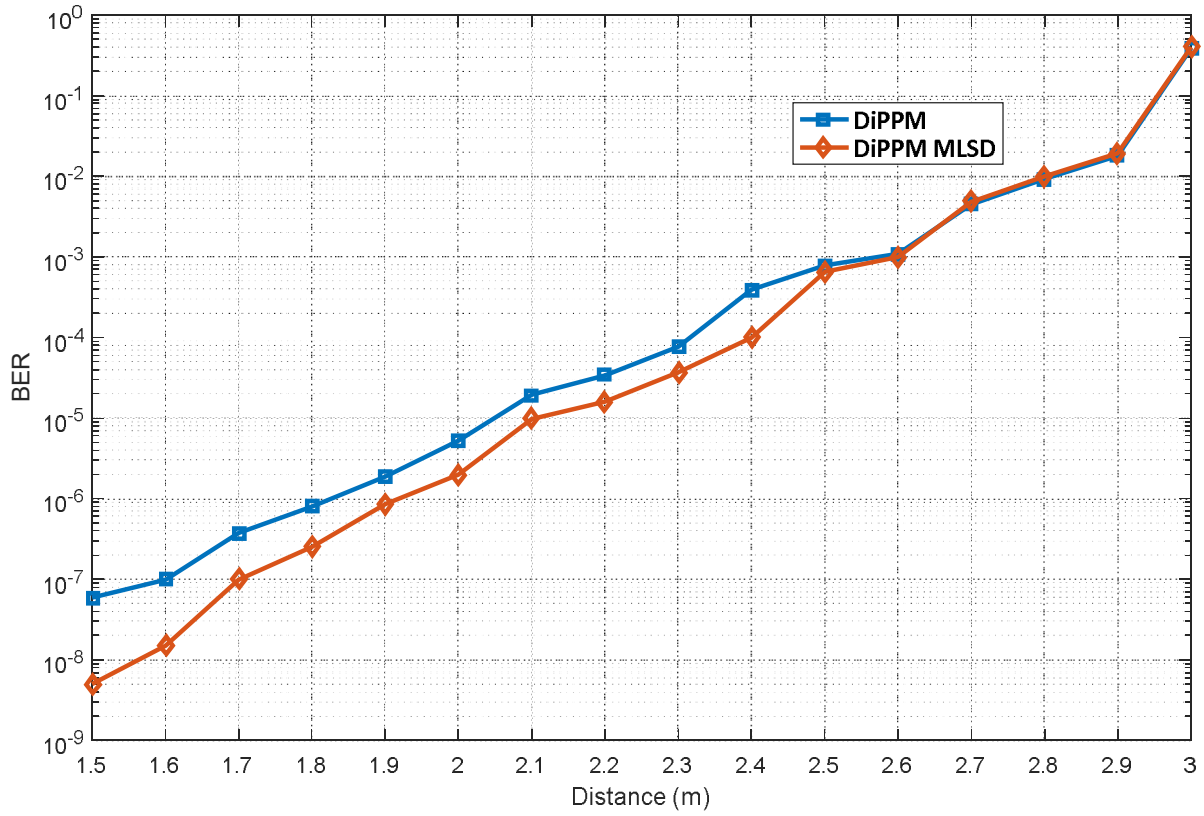


Figure 6-8 DiPPM and DiPPM MLSD BER comparison

Table 6-VII DiPPM and DuoPPM MLSD comparison

Coding scheme	DiPPM			DuoPPM		
	Without MLSD	With MLSD	Improvement (%)	Without MLSD	With MLSD	Improvement (%)
Ratio= Detected errors/ Induced errors	0.5	0.3	40	0.499	0.312	37.4

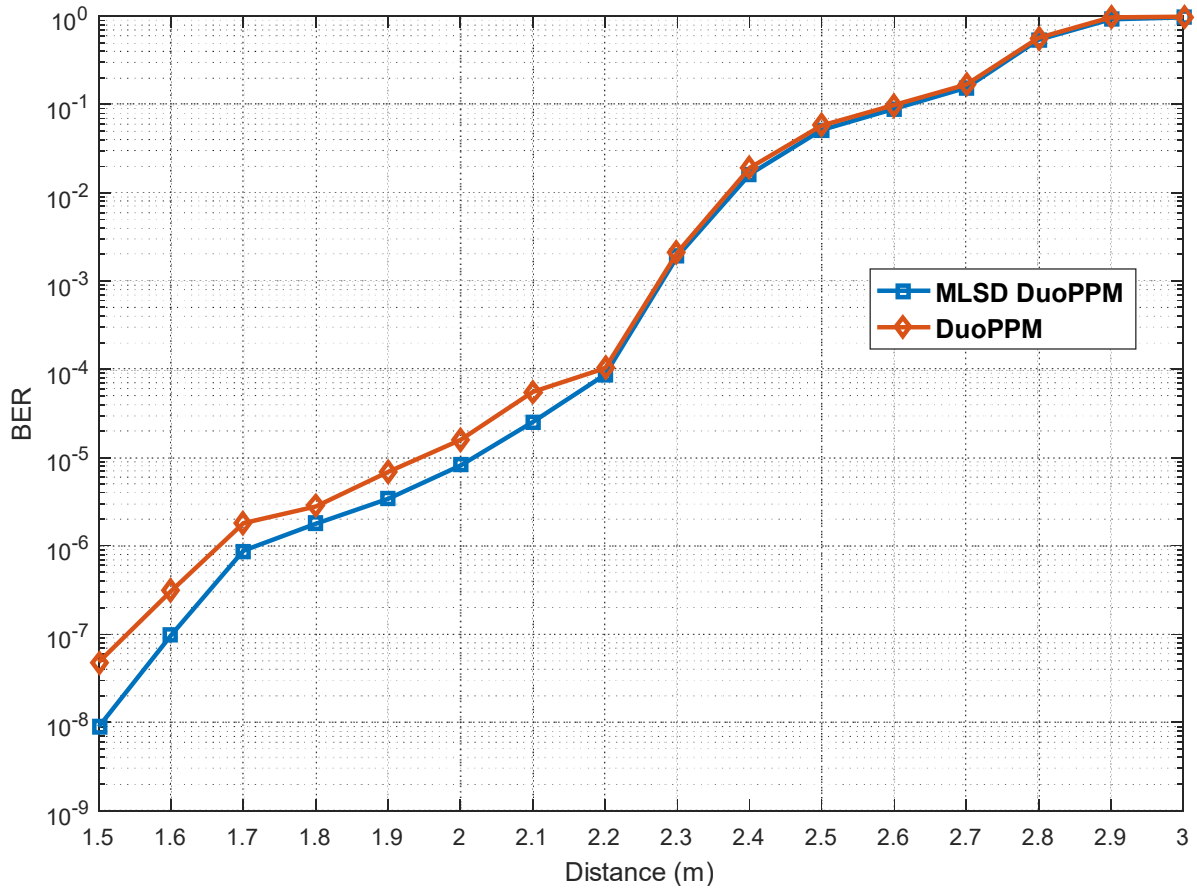


Figure 6-9 MLSD DiPPM signal simulations

6.6 Other tests

6.6.1 Bias-Tee

Figure 6-10 shows the input and output signals of the bias-T at 14 Mbps. It can be seen that the output of bias-T slightly differs from the input signal. Deteriorated pulses can be seen at the output of bias-T signal, resulting in transmitting an altered signal to the LED. This altered signal on the receiver side can cause multiple errors. If the spikes are above the threshold at the FPGA input pin these can be detected as logic 1, hence, altering the actual signal. Hence, Bias-T also contributes to ISI occurrence in the system.

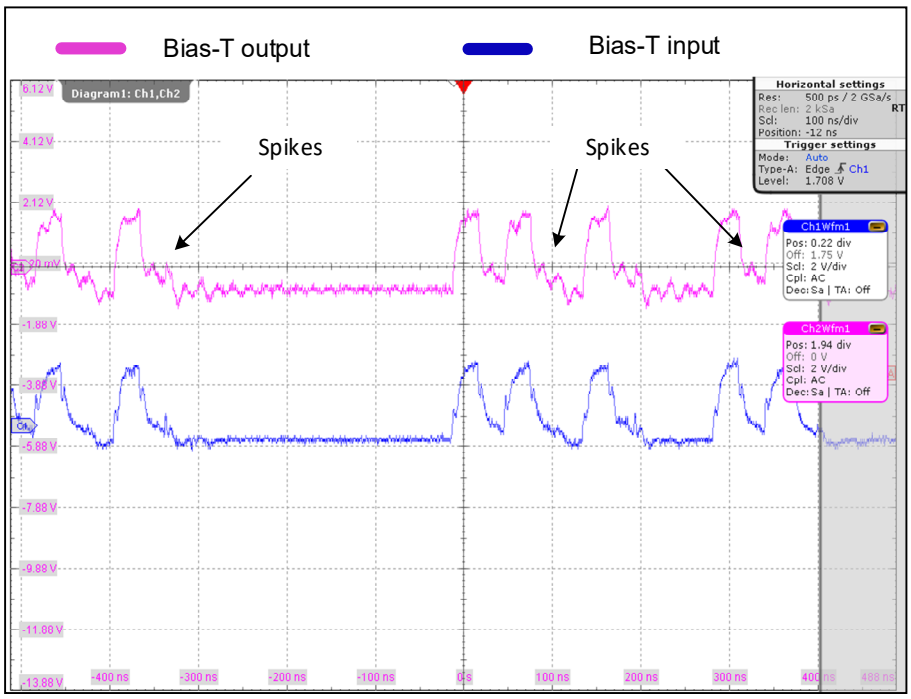


Figure 6-10 Bias-T output and input signals

6.6.2 Comparator

The received signal from VLC circuitry is fed back into the FPGA via the comparator. This comparator digitizes the output of the receiver circuitry for FPGA. From figure 6-11, the comparator rise time is slower and the output does not reach the full-scale value. This behaviour will cause some pulses to remain undetected at the FPGA input pin and pulses will be detected as 0 or no pulse. This behaviour of the comparator can significantly increase the BER of the system. Despite using a relatively faster comparator (Max942) the system continues to suffer from ISI.

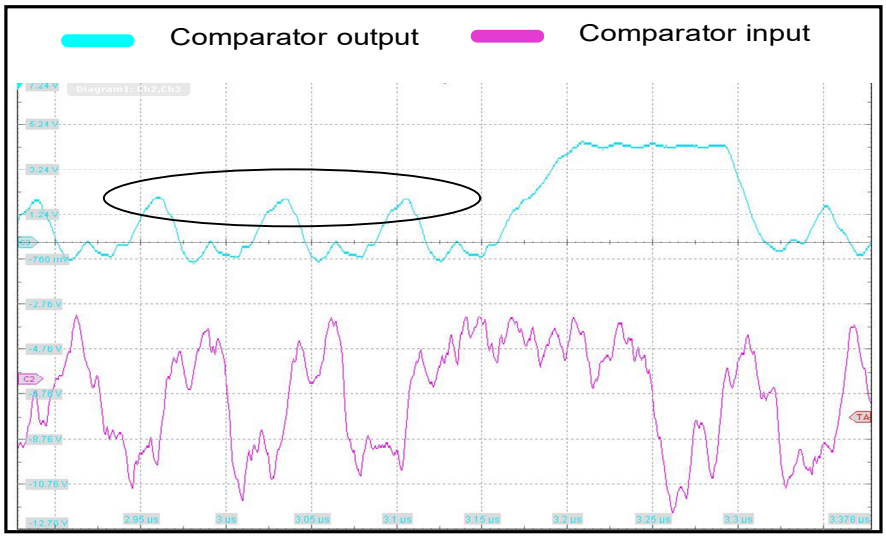


Figure 6-11 Comparator input and output signal waveforms

6.6.3 FPGA PLL signal

The DE2 board with Quartus provides a PLL solution and can generate desired clock frequency using the available PLL module. From figure 6-12, the clock frequency of 12.5 MHz shows a 50% duty cycle. However, the clock speed of 15 MHz does not produce a 50% duty cycle. The other observations include the clock frequencies with multiples of 2 show a 50% duty cycle. This means when the clock frequency is not a multiple of 2, the errors can be introduced by the clock signal. For example, if the distance between consecutive rising edges of the clock is not consistent some bits will be undetected, the FPGA will not be able to detect the correct sequence of incoming pulses.

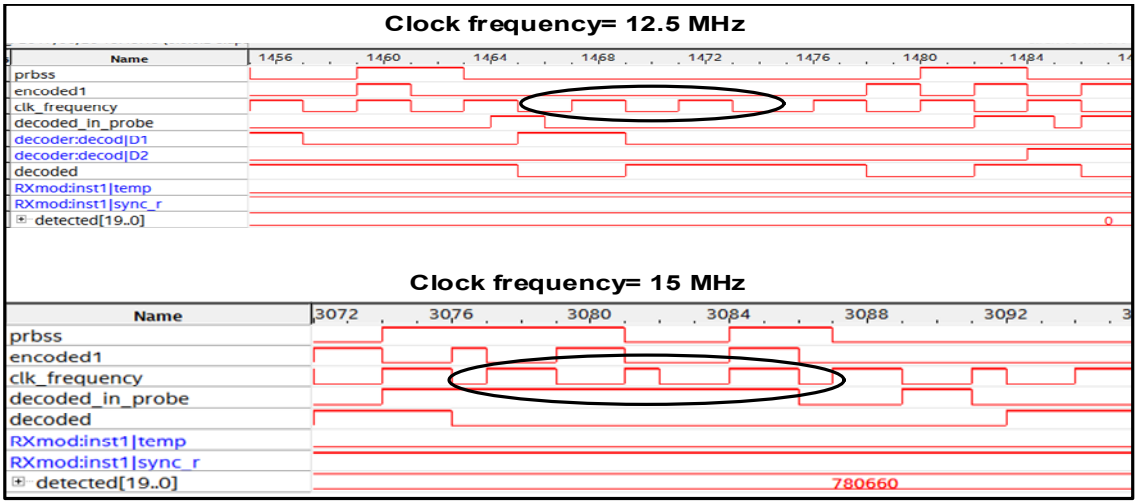


Figure 6-12 PLL clock signal at 12.5 MHz and 15 MHz

6.6.4 Types of errors detected

Mainly two types of errors were detected in the VLC system, erasure and ISI. Figures 6-13, shows the ISI effect occurrence at the received signal from the VLC circuitry. It can be seen the overlapping of pulses with consecutive pulses result in ISI errors. In the figure, 6-14 erasure error is detected at the received signal. A pulse has disappeared at the received signal showing an erasure error. Wrong slot and false alarm are very less likely to occur due to relatively faster electronic circuitry and PD.

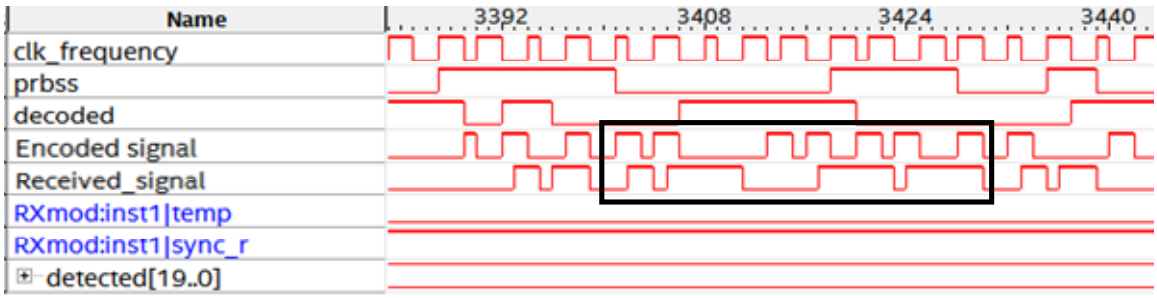


Figure 6-13 ISI detected in the received signal

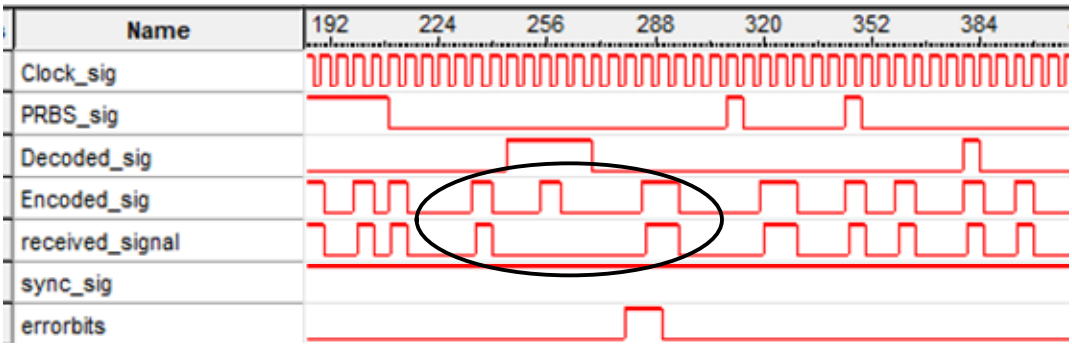


Figure 6-14 Erasure error detected at the received signal

6.7 Summary

This chapter presented, for the first time, the results and analysis for practical implementation and comparison of different PPM schemes. DPPM, MPPM74, OPPM, DiPPM, DuoPPM and MPPM64 were tested under identical conditions and the performance was analysed. A novel modified form of MPPM called Modified MPPM or MPPM64 offers 14.28% lower bandwidth expansion. A novel priority decoding method for OPPM was very effective against ISI and false alarms errors, and also improves the BER without the expense of bandwidth reduction. A simple error correction

technique like CRC can improve the BER of the system which can be implemented with simple combinational logic gates as compared to complex RS codes, convolution etc. MLSD can reduce improve the BER of the system by up to 10 times. The VLC system limitations have been identified; for example, the slower response of Bias-T, slower comparator response and threshold issue, the PLL signal duty cycle issue and the main types of errors detected include ISI and erasure.

Chapter 7
Conclusion and Future Work

7.1 Conclusions

The existing schemes must be identified and studied thoroughly before proposing new and revised schemes. This study sheds light on existing modulation techniques and focuses on practicality and implementation, which is lacking in existing studies. Mainly the existing studies focus only on theoretical models and calculation. Contrary to existing studies this study shows the practical implementation and comparison at the scale of up to 6 different schemes on a physical system. A detailed practical comparison was carried out under identical conditions over a VLC system.

This study shows that all selected PPM schemes (DPPM, MPPM, DiPPM, OPPM and DuoPPM) demonstrate reasonable performance on a VLC system, by achieving a minimum BER of 10^{-7} at a transmission distance of 1.5 m with a line rate of 14 Mbps. The answer to the optimal scheme for a VLC system or any other communication is not straight forward. For applications requiring high data rates with BER levels also in a relatively high range, then the most appropriate schemes will be DiPPM and DuoPPM, as these schemes provide no bandwidth expansion and the data rate is equal to the line rate of the system. DiPPM and DuoPPM achieved BER of 10^{-7} with the transmission distance of 1.5 m and 1.8 m, respectively, at the line rate of 14 Mbps. It is worth noting that DuoPPM achieved a higher distance of 0.3 m as compared to DiPPM. If an application requires very reliable data transmissions at higher transmissions distances then the answer will be the DPPM scheme, at the expanse of large bandwidth expansion or lower data rates. DPPM achieved a distance of 2.9 m with a BER of only 10^{-7} at the line rate of 14 Mbps. If an application requires moderate data transmission, data rate and transmission distance then the answer will be MPPM64, MPPM74 or OPPM which show moderate performance. Where OPPM achieved a transmission distance of 2.3 m with a BER of 10^{-8} at the line rate of 14 Mbps. On other hand, MPPM64 and MPPM74 achieved a distance of 2.7 m at the BER of 10^{-8} and 10^{-6}, respectively. Although the MPPM64 achieves higher BER when compared with the original MPPM74, it must be noted that MPPM64 requires less bandwidth expansion due to smaller codeword length. Thus, MPPM64 transmits more information at an identical line rate. Theoretical results show that MPPM64 has an

18.24 % less probability of error occurrence. To conclude the PPM selection, the Weighted sum approach can be utilised to simplify the process, depending on the weighting of different parameters defined by the user application performance requirements. For example, in table 7-I, more weight is given to bandwidth expansion and speed as compared to the power and transmission distance, where it can be seen that OPPM performs best followed by MPPM64 and MPPM74.

Table 7-I Weighted sum approach for PPM selection

	Normalised power estimation	**Normalised bandwidth expansion**	**Distance (m) -** *from figure 6-1*	**Speed (Mbps)-** *from figure 6-2*	**Score**
Weighting	**0.1**	**0.4**	**0.1**	**0.4**	**--**
DPPM	0.4	1.0	1.0	1.0	0.94
MPPM74	0.9	0.65	1.07	1.0	0.85
MPPM64	1.0	0.56	1.07	1.0	0.831
OPPM	1.0	0.49	0.79	1.0	0.775
DiPPM	0.79	0.37	1.61	1.307	0.91
DuoPPM	0.8	0.37	1.93	1.307	0.944

The BER of OPPM was improved with a novel Priority decoding method proposed in this study. When the experiment was conducted under identical circumstances the priority decoding and OPPM achieved the BER of 10^{-8} and 10^{-7}, respectively, at the transmission distance of 1.45 m at a line rate of 14 Mbps. This error correction scheme can be implemented utilising simple digital logic and does not require complex equations and circuits. This novel error correction was very effective against wrong slot and ISI errors occurring during transmission.

A modified form of MPPM called modified MPPM64 was presented in this study, which requires less bandwidth expansion due to the smaller codeword size. The advantage of transmission distance of 0.4 m was achieved at similar BERs of around 10^{-7} when the modulation schemes were tested under identical conditions i.e., identical LED, transmission speed and VLC circuitry. MPPM64 is capable of transmitting more information as compared to MPPM74 at identical speeds, and the results show

improved and reliable data transmission. It is worth mentioning that MPPM64 has half the number of total possible codewords when compared with MPPM74 codewords i.e., 64 codeword combinations.

This study presented, for the first time, practical implementation and comparison of MLSD for DiPPM and DuoPPM schemes. The DiPPM and DuoPPM are prone to error, a single bit can result in multiple errors. The system will introduce errors until the correct pulse or sequence is received. For example, if a S pulse is corrupted and becomes R pulse, the system will decode the wrong sequence until a correct pulse is received, resulting in a stream of errors. Theoretical tests show that MLSD has the potential to correct errors up to 40% in DiPPM and 37.4% in DuoPPM. However, the practical results have shown 10 times fewer errors were recorded using MLSD i.e., MLSD achieved BER of 10^{-8} while DiPPM and DuoPPM achieved 10^{-7}.

The artificial error inducing and BER calculation modules allow testing of any given scheme without the experimental link. Therefore, the performance measurements of PPM schemes can be measured without the need for an experimental link.

This testbench development and implementation also improved the performance of the VLC system; for example, the ISI and slower electronic circuitry response was identified during experimental studies and published by Farhat [80]. After addressing these issues and using a faster PD the VLC system performance was increased and a 0.5 m transmission distance advantage was achieved in DuoPPM, at BER of 10^{-9}.

The developed testbench offers flexibility and adaptability for incorporating more schemes in the future. The use of multiplexers in the testbench offer scalability. The modular setup of each scheme offers the addition of custom modules if and when required in the future. The ease of use and scalability offers the designer the flexibility to add more PPM schemes to the system test bench for more detailed and comprehensive analysis.

To conclude; the novel contributions of this research are:

- First-time presented, practical implementation and comparison of DiPPM, DuoPPM and OPPM schemes.

- First-time presented, practical implementation and comparison of different PPM schemes under identical conditions, enabling direct performance comparisons to be made.
- Testbench development using the HIL approach can be used to evaluate the performance of different PPM schemes without the need for an experimental set-up.
- Two novel modulation schemes i.e., Priority decoding and MPPM64 schemes.
- Identified the circuitry performance limitation issues within the existing VLC system.

7.2 Future Work and recommendations

The successful designing and implementation of different PPM schemes over an experimental link showed the validity of the schemes. This testbench was used for the detailed analysis of VLC circuitry and identified the flaws in the VLC system. These flaws must be addressed to further improve the VLC performance. The results have shown that even after improving the electronic circuitry i.e., faster PD and comparator IC, the ISI still exists showing the slower response of circuitry. The limitations of slower Bias-T, comparator and compensator circuits needs to be addressed. It is known that compensator circuits can add a damping effect on the signal thus slowing the rise and fall times of the circuit. A further investigation into electronic circuitry is required to achieve even higher data rates and transmission distances. An industry-standard PCB of multiple layers (4 or more) and surface mount ICs must be used to reduce the signal to noise ratio. The RF, TIA, PD and comparator circuits must be built on a single PCB.

The digital design of the testbench requires improvement to optimise the power and performance of the testbench model. This can be achieved using select signals with clock signal for each module to deactivate the unused modules, reducing dynamic power of the system. HIL experimentations can be further expanded to allow the user to achieve desired BER such as achieving a BER of 10^{-5}.

More schemes such as DPIM, VPPM, DH-PIM, Overlapping PPM can be integrated for future works. The research has shown that simple error correction techniques such as CRC can have significant improvement with regard to BER in the system. Error correction techniques such as hamming codes, RS codes, convolution can be considered and the results of complex and simple error corrections can be compared. Error inducing of different types such as erasure, wrong slot and false

alarm can be explored, to determine the performance of a given scheme against different types of error sources.

A full VLC based application can be designed to show the capability of the VLC using high power LEDs in the real world. Measurements of BER at different angles can be explored for real-world applications.

References

1. Fitz, M., *Fundamentals of communications systems*. 2007: Tata McGraw-Hill Education.
2. Tse, D. and P. Viswanath, *Fundamentals of wireless communication*. 2005: Cambridge university press.
3. Wood, A.W. and K. Karipidis, *Non-ionizing Radiation Protection: Summary of Research and Policy Options*. 2017: John Wiley & Sons.
4. Union, I.T. [cited 2020 21 March]; Available from: https://www.itu.int/en/ITU-R/Pages/default.aspx.
5. Ofcom. [cited 2020 21 March]; Available from: https://www.ofcom.org.uk/spectrum/information/uk-fat.
6. Armon, R.H. and O. Hänninen, *Environmental indicators*. 2015: Springer.
7. Society, A.C. [cited 2020 1 March]; Available from: https://www.cancer.org/cancer/cancer-causes/radiation-exposure/radiofrequency-radiation.html.
8. Commission, F.C. [cited 2020 1 March]; Available from: https://www.fcc.gov/engineering-technology/electromagnetic-compatibility-division/radio-frequency-safety/faq/rf-safety.
9. Saadi, M., et al., *Visible light communication: opportunities, challenges and channel models.* International Journal of Electronics & Informatics, 2013. **2**(1): p. 1-11.
10. Wang, Z., et al., *Visible light communications: modulation and signal processing*. 2017: John Wiley & Sons.
11. Ghassemlooy, Z., et al., *Digital pulse interval modulation for optical communications.* IEEE Communications Magazine, 1998. **36**(12): p. 95-99.
12. Mahdiraji, G.A. and E. Zahedi. *Comparison of selected digital modulation schemes (OOK, PPM and DPIM) for wireless optical communications*. in *4th Student Conference on Research and Development, 2006, (SCOReD), Selangor, Malaysia.* 2006. IEEE.
13. Zhang, T., et al. *A novel modified digital pulse interval modulation for optical wireless communications*. in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing. Dalian, China.* 2008. IEEE.
14. Abshire, J., *Performance of OOK and low-order PPM modulations in optical communications when using APD-based receivers.* IEEE transactions on communications, 1984. **32**(10): p. 1140-1143.
15. Sibley, M., *Performance analysis of a dicode PPM system, operating over plastic optical fibre, using maximum likelihood sequence detection.* IEE Proceedings-Optoelectronics, 2005. **152**(6): p. 337-343.
16. Quan, Z., M. Zhu, and S. Cui. *A fast receiver sensitivity identification method for wireless systems*. in *2016 IEEE International Conference on Communications (ICC)*. 2016. IEEE.

17. Garrett, I., *Pulse-position modulation for transmission over optical fibers with direct or heterodyne detection.* IEEE transactions on communications, 1983. **31**(4): p. 518-527.
18. Cryan, R. *Space communications employing optical position modulation*. in *3rd European Conference on Satellite Communications-ECSC-3, 1993.* . 1993. IET.
19. Leeson, M.S., *Pulse position modulation for spectrum-sliced transmission.* IEEE Photonics Technology Letters, 2004. **16**(4): p. 1191-1193.
20. Shiu, D.-s. and J.M. Kahn, *Differential pulse-position modulation for power-efficient optical communication.* IEEE transactions on communications, 1999. **47**(8): p. 1201-1210.
21. Xu, T., et al. *Design of a wireless laser communication system based on PPM technique*. in *2011 International Conference on Electrical and Control Engineering (ICECE). Yichang, China.* 2011. IEEE.
22. Sugiyama, H. and K. Nosu, *MPPM: A method for improving the band-utilization efficiency in optical PPM.* Journal of Lightwave Technology, 1989. **7**(3): p. 465-472.
23. Aldibbiat, N., Z. Ghassemlooy, and R. McLaughlin, *Dual header pulse interval modulation for dispersive indoor optical wireless communication systems.* IEE Proceedings-Circuits, Devices and Systems, 2002. **149**(3): p. 187-192.
24. Sibley, M.J., *Analysis of offset pulse position modulation–a novel reduced bandwidth coding scheme.* IET optoelectronics, 2011. **5**(4): p. 144-150.
25. Sibley, M.J., *Dicode pulse-position modulation: a novel coding scheme for optical-fibre communications.* IEE Proceedings-Optoelectronics, 2003. **150**(2): p. 125-131.
26. Mostafa, K., M.J. Sibley, and P.J. Mather, *Duobinary pulse position modulation–a novel coding scheme for the dispersive optical channel.* IET Optoelectronics, 2014. **8**(6): p. 264-269.
27. Lee, K. and H. Park, *Modulations for visible light communications with dimming control.* IEEE photonics technology letters, 2011. **23**(16): p. 1136-1138.
28. Shalaby, H.M., *Capacity and cutoff rate for optical overlapping pulse-position modulation channels.* IEEE transactions on communications, 1995. **43**(2/3/4): p. 1284-1288.
29. Selmy, H., H.M. Shalaby, and Z.-I. Kawasaki, *Proposal and performance evaluation of a hybrid BPSK-modified MPPM technique for optical fiber communications systems.* Journal of Lightwave Technology, 2013. **31**(22): p. 3535-3545.
30. Miao, M., et al., *Hybrid Modulation Scheme Combining PPM with Differential Chaos Shift Keying Modulation.* IEEE Wireless Communications Letters, 2018.

31. Ohtsuki, T., I. Sasase, and S. Mori. *Differential overlapping pulse position modulation in optical direct-detection channel*. in *IEEE ICC/SUPERCOMM'94 - 1994 IEEE International Conference on Communications, New Orleans, L.A., USA*. 1994. IEEE.
32. Sahoo, P.K., Y.K. Prajapati, and R. Tripathi, *PPM-and GMSK-based hybrid modulation technique for optical wireless communication cellular backhaul channel.* IET Communications, 2018. **12**(17): p. 2158-2163.
33. Seidman, L., *The performance of a PPM/PM hybrid modulation system.* IEEE Transactions on Communication Technology, 1970. **18**(5): p. 697-698.
34. Sombra, A. and J. Da Silva. *Pulse position modulation (PPM) of ultrashort pulse trains in optical fibres*. in *1997 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference. 'Linking to the Next Century'. Natal, Brazil.* 1997. IEEE.
35. Liu, C., et al. *Analysis of direct detection and coherent detection in wireless optical communication with polarization shift keying*. in *Conference on Lasers and Electro-Optics/Pacific Rim, Shanghai, China.* 2009. Optical Society of America.
36. Kim, H., *Wireless communications systems design*. 2015: John Wiley & Sons.
37. Bassett, R.K., *To the digital age: Research labs, start-up companies, and the rise of MOS technology*. 2002: JHU Press.
38. Glazebrook, R., *James Clerk Maxwell and modern physics*. 1896: MacMillan.
39. Nahin, P.J., *The Logician and the Engineer: how George Boole and Claude Shannon created the information age*. 2017: Princeton University Press.
40. Bhagyaveni, M., R. Kalidoss, and K. Vishvaksenan, *Introduction to analog and digital communication*. Vol. 46. 2016: River Publishers.
41. Ramirez-Iniguez, R., S.M. Idrus, and Z. Sun, *Optical wireless communications: IR for wireless connectivity*. 2008: CRC press.
42. Sklar, B., *Digital Communications: Pearson New International Edition UPDF EBook : Fundamentals and Applications*. 2013, Harlow, UNITED KINGDOM: Pearson Education, Limited.
43. Dimitrov, S. and H. Haas, *Principles of LED light communications: towards networked Li-Fi*. 2015: Cambridge University Press.
44. Ghassemlooy, Z., et al., *Visible light communications: theory and applications*. 2017: CRC press.
45. Arnon, S., et al., *Advanced optical wireless communication systems*. 2012: Cambridge university press.
46. Chan, V., *Space coherent optical communication systems--An introduction.* Journal of lightwave technology, 1987. **5**(4): p. 633-637.

47. Gregory, M., et al. *Inter-satellite and satellite-ground laser communication links based on homodyne BPSK*. in *Free-Space Laser Communication Technologies XXII*. 2010. International Society for Optics and Photonics.
48. Schiff, M., *Introduction to Communication systems simulation*. 2006: Artech House.
49. Lathi, B.P., *Modern digital and analog communications systems*. 1998: Oxford university press.
50. Du, K.-L. and M.N. Swamy, *Wireless communication systems: from RF subsystems to 4G enabling technologies*. 2010: Cambridge University Press.
51. Steendam, H., et al., *Synchronization in wireless communications*. 2009, SpringerOpen.
52. Wilson, B. and Z. Ghassemlooy, *Pulse time modulation techniques for optical communications: a review.* IEE Proceedings J (Optoelectronics), 1993. **140**(6): p. 346-358.
53. Ghassemlooy, Z., W. Popoola, and S. Rajbhandari, *Optical wireless communications: system and channel modelling with Matlab®*. 2019: CRC press.
54. Parween, S. and A. Tripathy. *Free space optic communication using optical AM, OOK-NRZ and OOK-RZ modulation techniques*. in *2019 3rd International conference on electronics, materials engineering & nano-technology (IEMENTech)*. 2019. IEEE.
55. Kumar, P. *Comparative analysis of BER performance for direct detection and coherent detection FSO communication systems*. in *2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT), Gwalior, India.* 2015. IEEE.
56. Amin, T.F. and S. Majumder. *Performance limitation of M-ary pulse position modulation and on-off keying in a direct detection optical wireless communication system*. in *10th (ICCIT 2007) International conference on Computer and information technology, 2007. Dhaka, Bangladesh.* 2007. IEEE.
57. Atkin, G. and K. Fung, *Performance analysis of coded optical PPM system using direct and coherent detection.* IEE Proceedings I-Communications, Speech and Vision, 1990. **137**(4): p. 226-232.
58. Lee, G. and G. Schroeder, *Optical pulse position modulation with multiple positions per pulsewidth.* IEEE Transactions on Communications, 1977. **25**(3): p. 360-364.
59. Gol'Dshteyn, Y.A. and B. Frezinskiy, *Investigation of the Transmission of a Multi-Position PPM Optical Signal Through a Communications Line Containing Repeaters.* RADIO ENG. & ELECTRON. PHYS., 1979. **24**(7): p. 65-71.
60. KONONOV, A., *INCREASING THE DATA TRANSMISSION SPEED IN AN OPTICAL DIGITAL COMMUNICATIONS CHANNEL.* 1978.

61. Ray, I., M.J. Sibley, and P.J. Mather, *Performance analysis of offset pulse-position modulation over an optical channel.* Journal of lightwave technology, 2011. **30**(3): p. 325-330.
62. Chaaban, A. and M.S. Alouini. *Optical intensity modulation direct detection versus heterodyne detection: A high-SNR capacity comparison*. in *2015 5th International Conference on Communications and Networking (2015 COMNET). Nov. 2015. Tunis, Tunisia.* 2015. IEEE.
63. Georghiades, C.N., *Modulation and coding for throughput-efficient optical systems.* IEEE Transactions on Information Theory, 1994. **40**(5): p. 1313-1326.
64. Fan, Y. and R.J. Green, *Comparison of pulse position modulation and pulse width modulation for application in optical communications.* Optical Engineering, 2007. **46**(6): p. 065001.
65. Jiang, Y., et al., *Packet error rate analysis of OOK, DPIM, and PPM modulation schemes for ground-to-satellite laser uplink communications.* Applied optics, 2014. **53**(7): p. 1268-1273.
66. Peppas, K.P., A.C. Boucouvalas, and Z. Ghassemloy, *Performance of underwater optical wireless communication with multi-pulse pulse-position modulation receivers and spatial diversity.* IET Optoelectronics, 2017. **11**(5): p. 180-185.
67. Issa, A.A.-K.M., *Investigation of the pulse time modulation techniques for transmission of wideband signals*. 1997, Sheffield Hallam University.
68. Xiong, F., *Digital Modulation Techniques, (Artech House Telecommunications Library)*. 2006: Artech House, Inc.
69. Garrett, I. *Digital pulse-position modulation over dispersive optical fibre channels*. in *Int. Workshop on Digital Communications, Tirrenia, Italy*. 1983.
70. Garrett, I. *Digital pulse-position modulation over slightly dispersive optical fibre channels*. in *International symposium on Information Theory, St. Jovite*. 1983.
71. Cryan, R. and M.J. Sibley, *Minimising intersymbol interference in optical-fibre dicode PPM systems.* IEE Proceedings-Optoelectronics, 2006. **153**(3): p. 93-100.
72. Spolitis, S. and G. Ivanovs, *Realization of combined chromatic dispersion compensation methods in high speed WDM optical transmission systems.* Elektronika ir Elektrotechnika, 2011. **113**(7): p. 101-106.
73. McEliece, R., *Practical codes for photon communication.* IEEE Transactions on Information Theory, 1981. **27**(4): p. 393-398.
74. Barry, J.R. *Sequence detection and equalization for pulse-position modulation*. in *ICC/SUPERCOMM'94 - 1994 International Conference on Communications on August 2002. New Orleans, L.A., USA.* 1994. IEEE.

75. Carlson, A.B., *Communication systems. An introduction to signals and noise in electrical engineering (3rd ed.)*. 1986, London; New York; McGraw-Hill.
76. Lee, K. and H. Park, *Modulations for visible light communications with dimming control.* IEEE photonics technology letters, 2011. **23**(13): p. 1136.
77. Charitopoulos, R., M.J. Sibley, and P. Mather, *Maximum likelihood sequence detector for dicode pulse position modulation.* IET optoelectronics, 2011. **5**(6): p. 261-264.
78. Al-Nedawe, B.M., et al. *Improving error performance of dicode pulse position modulation system using forward error correction codes*. in *2013 21st Telecommunications Forum Telfor (TELFOR)*. 2013. IEEE.
79. Ahfayd, M.H., et al. *Visible light communication based system using high power LED and dicode pulse position modulation technique*. in *2017 25th IEEE Telecommunication Forum (TELFOR)*. 2017. Belgrade, Serbia IEEE.
80. Farhat, Z.A., et al. *Practical implementation of duobinary pulse position modulation using FPGA and visible light communication*. in *2017 IEEE 15th Student Conference on Research and Development (SCOReD)*. 2017. Putrajaya, Malaysia: IEEE.
81. Arnon, S., *Visible light communication*. 2015: Cambridge University Press.
82. Haigh, P., et al. *Development of a visible light communications system for optical wireless local area networks*. in *2012 Computing, Communications and Applications Conference*. 2012. IEEE.
83. Pepeljugoski, P., et al., *Development of system specification for laser-optimized 50-μm multimode fiber for multigigabit short-wavelength LANs.* Journal of Lightwave Technology, 2003. **21**(5): p. 1256.
84. Bellè, A., et al. *Development of IEEE802. 15.7 based ITS services using low cost embedded systems*. in *2013 13th International Conference on ITS Telecommunications (ITST)*. 2013. IEEE.
85. Uema, H., et al., *Research and development on underwater visible light communication systems.* Electronics and Communications in Japan, 2015. **98**(3): p. 9-13.
86. Sze, S.M. and K.K. Ng, *Physics of semiconductor devices*. 2006: John wiley & sons.
87. Ahfayd, M.H., *Visible Light Communication Based on High-Power LED*. 2019, University of Huddersfield.
88. Vishay. *Silicon PIN Photodiode BPW34*. [Datasheet] 2011 [cited 2021 01 April]; Available from: http://www.farnell.com/datasheets/2046123.pdf.
89. Centronic. *Silicon Photodetector Series 5T.* [cited 2021 01 April]; Available from: https://docs.rs-online.com/514b/0900766b8151e3d2.pdf.

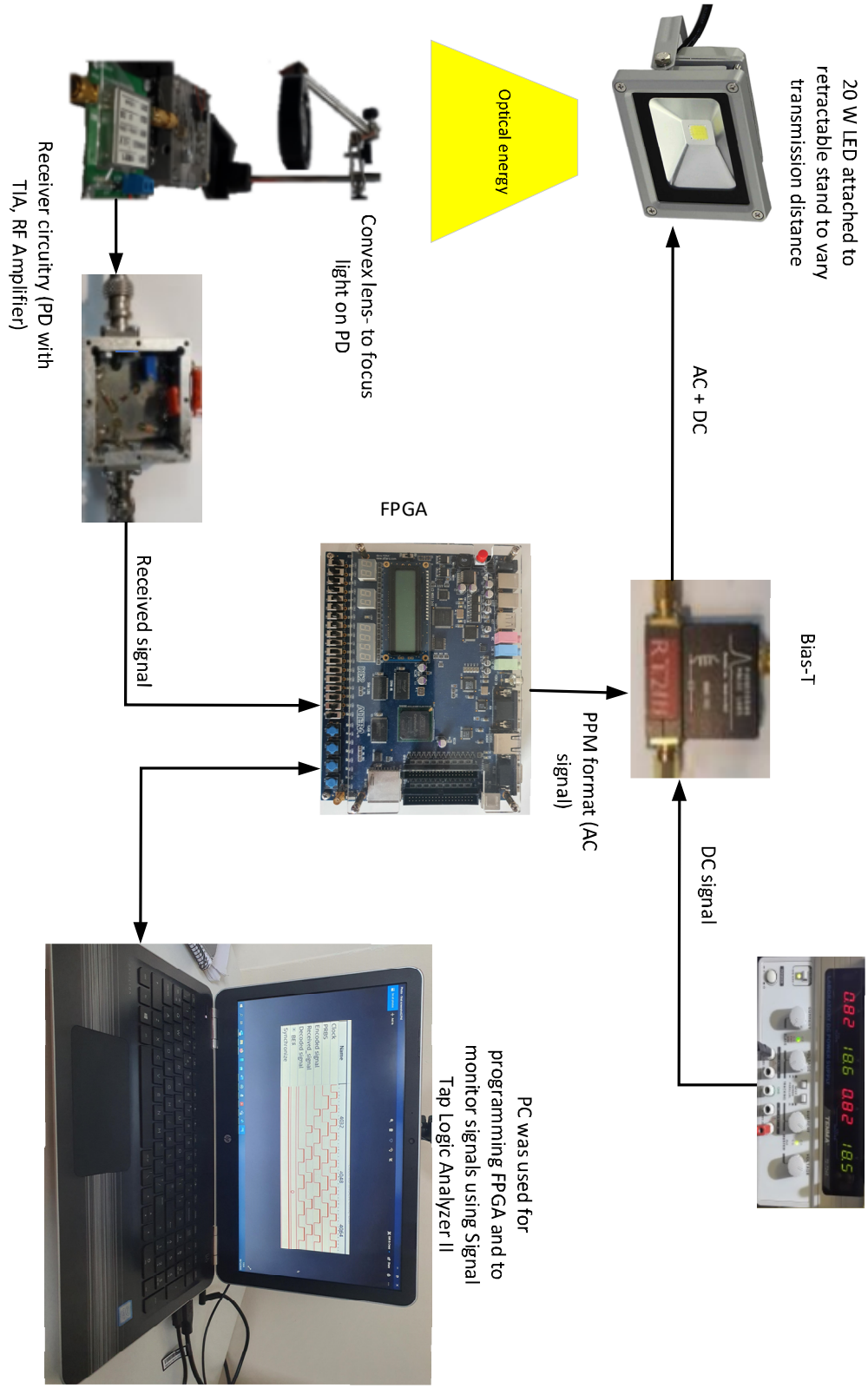
90. THORLABS. *Si Photodiode*. 2014 [cited 2021 01 April]; Available from: https://docs.rs-online.com/514b/0900766b8151e3d2.pdf.
91. Leven, A., *Telecommunication circuits and technology*. 2000: Elsevier.
92. Maxim. *High-Speed, Low-Power, 3V/5V, Rail-to-Rail,*

Single-Supply Comparators. 2014 [cited 2021 01 April]; Available from: https://datasheets.maximintegrated.com/en/ds/MAX941-MAX944.pdf.
93. TI. *LMx93-N, LM2903-N Low-Power, Low-Offset Voltage, Dual Comparators*. 2018 [cited 2021 01 April]; Available from: https://www.ti.com/lit/gpn/lm393-n.
94. Ahfayd, M.H., et al. *Selection of high power LEDs for Li-Fi applications*. in *2018 25th International Conference on Telecommunications (ICT)*. 2018. IEEE.
95. Veraszto, Z. and G. Stepan, *Nonlinear dynamics of hardware-in-the-loop experiments on stick–slip phenomena.* International Journal of Non-Linear Mechanics, 2017. **94**: p. 380-391.
96. Terasic. *Altera DE2 Developement and Expansion board*. [cited 2020 30 Dec]; Available from: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=53&No=30&PartNo=2.
97. Intel. *Intel Quartus Prime: FPGA Design Software*. 2020 [cited 2020 31 December]; Available from: https://www.intel.co.uk/content/www/uk/en/software/programmable/quartus-prime/overview.html.
98. Bezerra, E.A. and D.V. Lettnin, *Synthesizable VHDL design for FPGAs*. 2014: Springer.
99. Ashenden, P.J., *The designer's guide to VHDL*. 2010: Morgan Kaufmann.
100. Farhat, Z.A., et al. *Improved BER for offset pulse position modulation using priority decoding over VLC system*. in *2019 Wireless Days (WD)*. 2019. IEEE.

Appendices

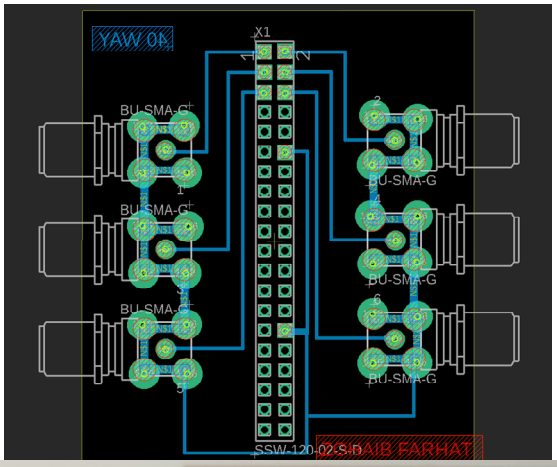
Appendix A

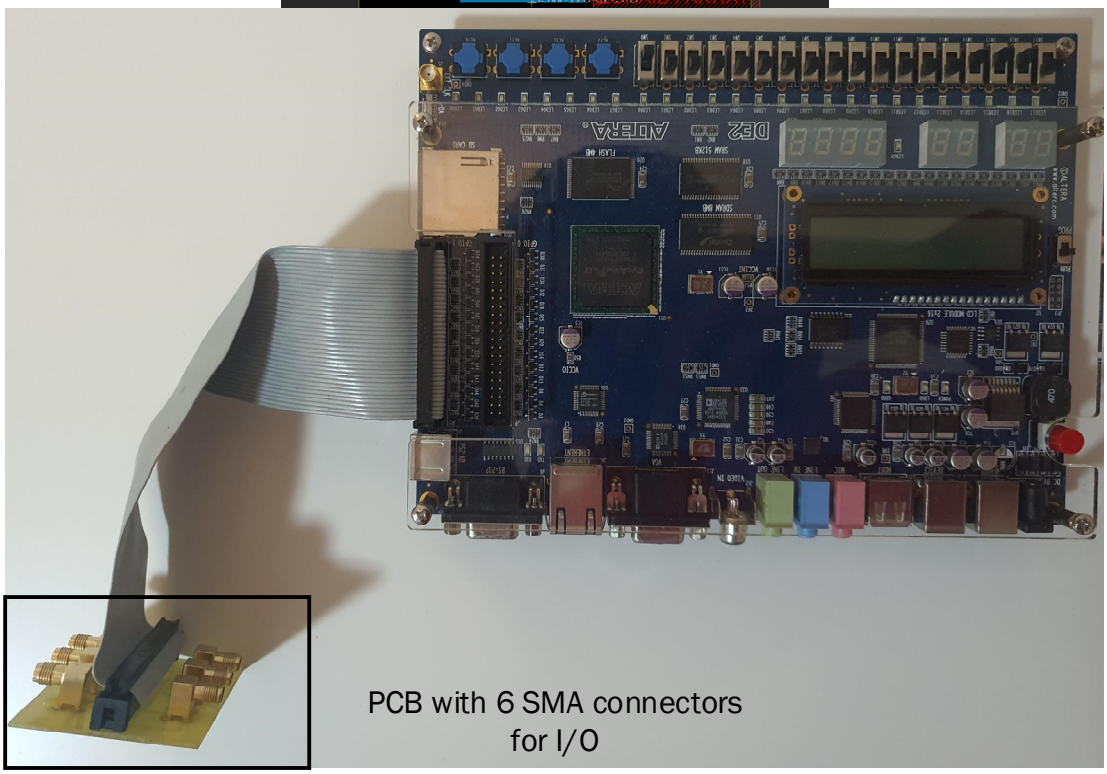
Block diagram of the experimental link used is given below:



Appendix B

The Altera DE2 Development and Education Board [96] was used for designing and implementing different PPM schemes. This board provides two expansion headers for simple I/O (2 x 40 pins) interfaces. A custom PCB was designed with 6 SMA connectors for reliable I/O operation. The schematic and the picture of the custom PCB used to connect FPGA to the experimental link is given below:



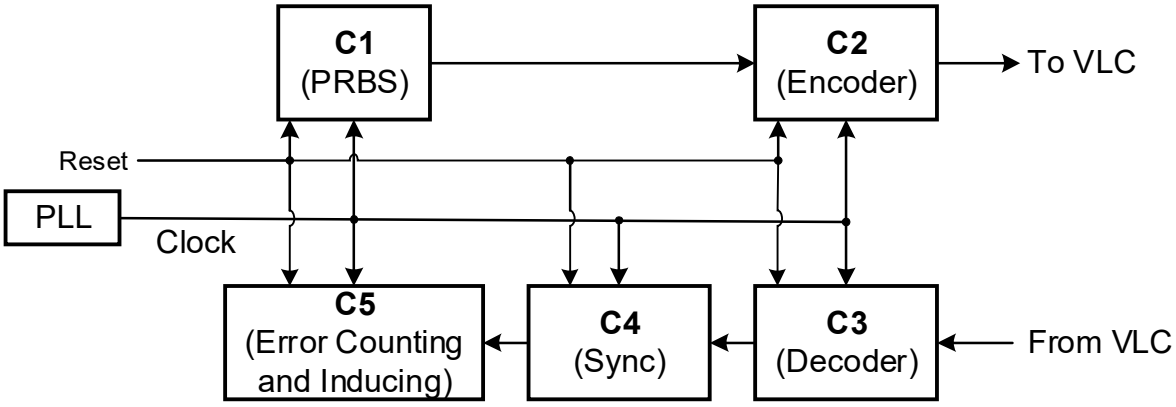


PCB with 6 SMA connectors for I/O

Appendix C

Simplified Quartus schematic and VHDL code for DPPM is given below:

Schematic:



Code C1 (PRBS)

```
--DPPM 15-bit PRBS using one clock
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;

Entity PRBS is
    PORT(
        clk :   IN STD_LOGIC;
        res :   IN STD_LOGIC;
        outp    :   OUT STD_LOGIC
        );
End PRBS;

Architecture bhv of PRBS is
Signal  control :STD_LOGIC_VECTOR(7 downto 0):="11111000";
Signal  reg     :STD_LOGIC_VECTOR(14 downto 0):=(others=>'1'); --15bit
Begin
Process(res,clk)
Begin
    IF(res='0')then
        control <=  "11111000";
        reg     <=  (others=>'1');
    ELSIF(clk'event and clk='1')then
        control <=  control(0) & control(7 downto 1);
        IF(control(4)='0')then-works at 4
            reg <=  reg(13 downto 0) & (reg(14) XOR reg(13));
        END IF;
    END IF;
End process;
outp    <=  reg(14);
End bhv;
```

Code C2 (Encoder)

```
--OFFSET PPM encoder using one clock
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
                        --++                    ENTITY                    ++-
-
ENTITY encoder38 IS
PORT    (   clk :   IN STD_LOGIC;
            res :   IN STD_LOGIC;
            inp :   IN STD_LOGIC;
            transmitted_cw  :   OUT STD_LOGIC_VECTOR(7 downto 0);
            transmitted_dw  :   OUT STD_LOGIC_VECTOR(2 downto 0);
            outp:   OUT STD_LOGIC);
END encoder38;
                        --++                ARCHITECTURE                ++-
-
ARCHITECTURE behaviour OF encoder38 IS
SIGNAL control  :   STD_LOGIC_VECTOR(7 downto 0):="11111100";  --control reg
SIGNAL reg8 :STD_LOGIC_VECTOR(7 downto 0):=(others => '0');--4bit output reg
SIGNAL sipo STD_LOGIC_VECTOR(2 downto 0):=(others => '0'); --7bit input reg
SIGNAL reg_cd   :   STD_LOGIC_VECTOR(7 downto 0):=(others   =>'0');
SIGNAL reg3     :   STD_LOGIC_VECTOR(2 downto 0):=(others => '0');
SIGNAL out_reg  :   STD_LOGIC:='0';
BEGIN
PROCESS(res,clk)
BEGIN
    IF(res='0') THEN
        control <=  "11111100";
        sipo        <=  (others =>  '0');
        reg3        <=  (others=>   '0');
        reg8        <=  (others =>  '0');
        reg_cd  <=  (others=>'0');
        out_reg <=  '0';

    ELSIF(clk'EVENT AND clk='1') THEN
        sipo    <=  inp & sipo(2 downto 1);     --input -> serial in
        control<= control(0) & control(7 downto 1);  --control reg
        IF(control="11111100") THEN         --control regs synced
            transmitted_dw  <= sipo;
            CASE sipo IS                                --encoding to 4bits
                when "000" =>   reg8<="00000001"; reg_cd<= "00000001";
                when "001" =>   reg8<="00000010"; reg_cd<= "00000010";
                when "010" =>   reg8<="00000100"; reg_cd<= "00000100";
                when "011" =>   reg8<="00001000"; reg_cd<= "00001000";
                when "100" =>   reg8<="00010000"; reg_cd<= "00010000";
                when "101" =>   reg8<="00100000"; reg_cd<= "00100000";
                when "110" =>   reg8<="01000000"; reg_cd<= "01000000";
                when "111" =>   reg8<="10000000"; reg_cd<= "10000000";
            end case;
        ELSE
            reg8<= '0' & reg8(7 downto 1);      --shift zeros
        END IF;
    END IF;
    out_reg <=  reg8(0);
END PROCESS;
transmitted_cw  <=  reg_cd;
outp <= out_reg;
END behaviour;
```

Code C3 (Decoder)

```
--  DPPM PPM Decoder
-- Decodes 8bits to 3bits
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
                                --++            ENTITY          ++--
ENTITY decoder83 IS
PORT (  inp :       IN STD_LOGIC;
            clk :       IN STD_LOGIC;
            res :       IN STD_LOGIC;
            rec_cw:     OUT STD_LOGIC_VECTOR(7 downto 0);
            rec_dw:     OUT STD_LOGIC_VECTOR(2 downto 0);
            outp:           OUT STD_LOGIC
        );
END decoder83;
                                --++        ARCHITECTURE        ++--
ARCHITECTURE behaviour OF decoder83 IS
SIGNAL  control :   STD_LOGIC_VECTOR(7 downto 0):="11100111";       --control
register
SIGNAL   reg3          :    STD_LOGIC_VECTOR(2 downto 0):=(others    =>   '0');
--output register
SIGNAL   sipo          :    STD_LOGIC_VECTOR(7 downto 0):=(others    =>   '0');
--7bit input reg- serial in
Signal   reg_dw   :    STD_LOGIC_VECTOR(2 downto 0):=(others    =>   '0');
signal   reg8          :    STD_LOGIC_VECTOR(7 downto 0):=(others    =>   '0');
SIGNAL   reg_rec :    STD_LOGIC_VECTOR(7 downto 0):=(others=> '0');
signal   out_reg :    std_logic;
signal   in_reg   :    std_logic;

BEGIN
--In this process, last 4bits of sipo reg are used to decode the message and
serially outputted
PROCESS(res,clk)
BEGIN
    IF(res='0')THEN
        sipo        <=(others    =>   '0');
        control <="11100111";
        reg3        <=(others    =>   '0');
        reg_dw   <=(others    =>   '0');
        reg8        <=(others    =>   '0');
        reg_rec <=(others    =>   '0');
        out_reg <=   '0';
    ELSIF (clk'EVENT AND clk='1') THEN
        control <= control(0) & control(7 downto 1);       --cotrol register\
        in_reg   <=   inp;
        sipo        <= in_reg & sipo(7 downto 1);        --serial input
        IF (control="11111001") THEN
            reg_rec <=   sipo(7 downto 0);    --original codeword received
            --reg8       <=   sipo(7 downto 0);
            case sipo(7 downto 0) is                        --decoding
                when "00000001" =>   reg3 <= "000"; reg_dw <= "000";
                when "00000010" =>   reg3 <= "001";   reg_dw <= "001";
                when "00000100" =>   reg3 <= "010";   reg_dw <= "010";
                when "00001000" =>   reg3 <= "011";   reg_dw <= "011";
                when "00010000" =>   reg3 <= "100"; reg_dw <= "100";
                when "00100000" =>   reg3 <= "101"; reg_dw <= "101";
                when "01000000" =>   reg3 <= "110"; reg_dw <= "110";
                when "10000000" =>   reg3 <= "111"; reg_dw <= "111";
```

```
                when others => reg3 <= "111";
            end case;
        ELSIF(control(0)='0')THEN
           reg3    <=  '0' & reg3(2 downto 1); --shifting with zeros->serial
output
        END IF;
    END IF;
END PROCESS;
rec_cw  <=  reg_rec;
rec_dw  <=  reg_dw;
outp    <=  reg3(0);
END behaviour;
```

Code C4 (Sync)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY syncmod IS
    PORT        (   clk3                :   IN STD_LOGIC;
                    res             :   IN STD_LOGIC;
                    sin             :   IN STD_LOGIC;
                    synced          :   OUT STD_LOGIC;
                    error_det       :   OUT STD_LOGIC
                    );
END syncmod;

ARCHITECTURE behaviour OF syncmod IS

SIGNAL  reg15           :   STD_LOGIC_VECTOR(14 downto 0);  --input register
SIGNAL  prbs            :   STD_LOGIC_VECTOR(14 downto 0);  -- PRBS register
SIGNAl  control     :   STD_LOGIC_VECTOR(7 downto 0):="11111000";
SIGNAL  sync_r      :   STD_LOGIC:='0';
signal  in_reg      :   std_logic:='0';
SIGNAL  temp            :   STD_LOGIC:='0';
signal  err_reg     :  std_logic:='0';
signal  sync_reg        :   std_logic:='0';

BEGIN
in_reg  <=  sin;
PROCESS(res,clk3)
BEGIN
    IF (res='0') THEN
        reg15       <=  (others=>'0');
        prbs        <=  (others=>'0');
        control <=  "11111000";
        sync_r  <=  '0';
        temp        <=  '0';

    ELSIF (clk3'EVENT AND clk3='1') THEN
    control<= control(0) & control(7 downto 1);
    if(control(5)='0')then
        reg15   <=  reg15(13 downto 0) & sin;   --input register serial input
        prbs      <= prbs(13  downto  0)  &  (prbs(13)  XOR  prbs(14));--PRBS
expression

        IF(reg15="11111111111111" AND sync_r='0') THEN --Wait for sync
            prbs<="1111111111111" & sin;--To match it with input reg(reg15)
            sync_r<=    '1';
        ELSIF(sync_r='1') THEN
            temp    <=  reg15(0) XOR prbs(0);           --new input at zero
place(reg15) & new seq for prbs at zero place.
        ELSE
            temp    <= '0';
        END IF;
    end if;
    END IF;
    err_reg <=  temp;
    sync_reg    <=  sync_r;
END PROCESS;
error_det   <=err_reg;
synced      <=sync_reg;       -- to indicate if the system has been synced
END behaviour;
```

Code C5 (Error Counting and Inducing)

```
--      Error inducing and error counting system        --
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
--                                                              ENTITY
--
Entity ERRmod Is
    PORT    (   clk         :   IN STD_LOGIC;
                res         :   IN STD_LOGIC;
                sync            :   IN STD_LOGIC;
                enable      :   IN STD_LOGIC;
                error           :   IN STD_LOGIC;
                bit_sel     :   IN  STD_LOGIC_VECTOR(2 downto 0);
                insert      :   OUT STD_LOGIC;
                count_error :   OUT STD_LOGIC_VECTOR(29 DOWNTO 0);
                err_ind     :   OUT STD_LOGIC_VECTOR(29 downto 0);
                err_det     :   OUT STD_LOGIC_VECTOR(29 downto 0)
            );

END ERRmod;
--        ARCHITECTURE                      --
ARCHITECTURE behaviour of ERRmod Is

SIGNAL  control             :STD_LOGIC_VECTOR(7 downto 0):=”10000000”;
SIGNAL  induced     :STD_LOGIC_VECTOR(29 downto 0):=(others =>  ‘0’);
SIGNAL  detected        :STD_LOGIC_VECTOR(29 downto 0):=(others =>  ‘0’);
SIGNAL  temp1           :STD_LOGIC_VECTOR(29 DOWNTO 0):=(others =>  ‘0’);
SIGNAL  pulse_count :STD_LOGIC_VECTOR(29 downto 0):=(others =>  ‘0’);
SIGNAL  prbs            :STD_LOGIC_VECTOR(6 downto 0):=(others  =>  ‘1’);
SIGNAL  err_bit     :STD_LOGIC;
SIGNAL  insert_bit  :STD_LOGIC;
SIGNAL  reg_error   :STD_LOGIC;
SIGNAL  reg_sync        :STD_LOGIC;

BEGIN
reg_sync    <=  sync;
--This process operates at clk4, and induces the error by inverting the
respective bit
PROCESS(res,clk)
BEGIN
    IF(res=’0’)THEN
        control             <=  “10000000”;
        err_bit     <=  ‘0’;
        insert_bit  <=  ‘0’;
        prbs            <=(others   =>  ‘1’);
        induced     <=(others   =>  ‘0’);
        detected        <=(others   =>  ‘0’);
        temp1           <=(others   =>  ‘0’);
        pulse_count <=(others   =>  ‘0’);
    ELSIF(clk’EVENT AND clk=’1’)THEN
        control     <=  control(0) & control(7 downto 1);
        prbs    <=  prbs(5 downto 0) & (prbs(6) XOR prbs(5));

        IF(control=”10000000”)THEN
            insert_bit  <=  prbs(5) AND prbs(6) AND prbs(4) AND prbs(3) AND
prbs(2) AND prbs(1) AND prbs(0)AND (not enable) AND reg_sync;   --wait for
sync & enable
        END IF;
```

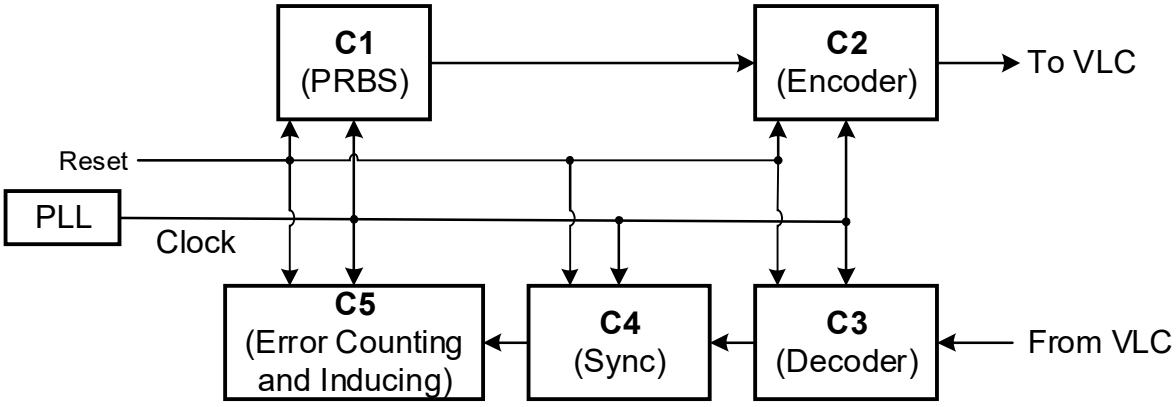
```
--select where to insert error0
        case    bit_sel is
            when    "000"   =>  reg_error   <= control(7) AND insert_bit;
            when    "001"   =>  reg_error   <= control(6) AND insert_bit;
            when    "010"   =>  reg_error   <= control(5) AND insert_bit;
            when    "011"   =>  reg_error   <= control(4) AND insert_bit;
            when    "100"   =>  reg_error   <= control(3) AND insert_bit;
            when    "101"   =>  reg_error   <= control(2) AND insert_bit;
            when    "110"   =>  reg_error   <= control(1) AND insert_bit;
            when    "111"   =>  reg_error   <= control(0) AND insert_bit;
        end case;

        IF(pulse_count=1000000000) THEN -1giga pulses
            temp1   <=  detected;
            pulse_count <=  (others =>  '0');
            detected        <=  (others =>  '0');
            induced     <=  (others =>  '0');
        else
                pulse_count <=  pulse_count +1;
        END IF;

            detected    <=  detected + error;
            induced <=  induced + reg_error;
    --EnD IF;
    END IF;
END PROCESS;
count_error <=  detected;
err_ind     <=  induced;
err_det     <=  temp1;
insert      <=  reg_error;
END behaviour;
```

Appendix D

Quartus software schematic and VHDL code for MPPM is given below:



Code C1 (PRBS)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;

Entity PRBS is
    PORT(
        clk :   IN STD_LOGIC;
        res :   IN STD_LOGIC;
        outp    :   OUT STD_LOGIC
        );
End PRBS;

Architecture bhv of PRBS is
Signal  control :STD_LOGIC_VECTOR(6 downto 0):="1110000";
Signal  reg     :STD_LOGIC_VECTOR(15 downto 0):=(others=>'1');
Signal  temp        :STD_LOGIC:='0';
Begin
Process(res,clk)
Begin
    IF(res='0')then
        temp        <='0';
        control <=  "1110000";
        reg     <=  (others=>'1');
    ELSIF(clk'event and clk='1')then
        control <=  control(0) & control(6 downto 1);
        IF(control(5)='0')then
            reg <=  reg(14 downto 0) & (reg(14) XOR reg(15));
        END IF;
    END IF;
outp    <=  reg(14);
End process;
End bhv;
```

Code C2 (Encoder)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;
Entity mppmEN is
    PORT(
        clk :   IN STD_LOGIC;
        dataword    :   OUT STD_LOGIC_VECTOR(3 downto 0);
        codeword    :   OUT STD_LOGIC_VECTOR(6 downto 0);
        inp :   IN STD_LOGIC;
        res :   IN STD_LOGIC;
        outp    :   OUT STD_LOGIC
        );
End mppmEN;

Architecture bhv of mppmEN is
Signal  inreg           :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  dreg            :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  outreg      :   STD_LOGIC_VECTOR(6 downto 0):=(others=>'0');
Signal  creg            :   STD_LOGIC_VECTOR(6 downto 0):=(others=>'0');
Signal  control     :   STD_LOGIC_VECTOR(6 downto 0):="1110000";
Signal  temp            :   STD_LOGIC:='0';
Begin
Process(res,clk)
Begin
    IF(res='0')then
        inreg       <=  (others=>'0');
        outreg  <=  (others=>'0');
        dreg        <=  (others=>'0');
        creg        <=  (others=>'0');
        control <=  "1110000";
        temp        <='0';
    ELSIF(clk'event and clk='1')then
        control <=  control(0) & control (6 downto 1);
        inreg       <=  inp & inreg(3 downto 1);
        IF(control="1110000")then
            dreg    <=  inreg(3 downto 0);
            case inreg(3 downto 0) is
                when "0000" => outreg   <=  "1100000";  creg  <= "1100000";
                when "0001" => outreg   <= "1010000";   creg  <= "1010000";
                when "0010" => outreg   <= "1001000";   creg  <= "1001000";
                when "0011" => outreg   <= "1000100";   creg  <= "1000100";
                when "0100" => outreg   <= "1000010";   creg  <= "1000010";
                when "0101" => outreg   <= "1000001";   creg  <= "1000001";
                when "0110" => outreg   <= "0110000";   creg  <= "0110000";
                when "0111" => outreg   <= "0101000";   creg  <= "0101000";
                when "1000" => outreg   <= "0100100";   creg  <= "0100100";
                when "1001" => outreg   <= "0100010";   creg  <= "0100010";
                when "1010" => outreg   <= "0100001";   creg  <= "0100001";
                when "1011" => outreg   <= "0011000";   creg  <= "0011000";
                when "1100" => outreg   <= "0010100";   creg  <= "0010100";
                when "1101" => outreg   <= "0010010";   creg  <= "0010010";
                when "1110" => outreg   <= "0010001";   creg  <= "0010001";
                when "1111" => outreg   <= "0001100";   creg  <= "0001100";
            end case;
        ELSE
            outreg  <= '0' & outreg(6 downto 1);
        END IF;
```

```
    END IF;
temp    <=  outreg(0);
End process;
dataword<=  dreg;
codeword<=  creg;
outp    <=  temp;
End bhv;
```

Code C3 (Decoder)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;

Entity mppmDE is
    PORT(
        clk :   IN STD_LOGIC;
        inp :   IN STD_LOGIC;
        dataword    :   OUT STD_LOGIC_VECTOR(3 downto 0);
        codeword    :   OUT STD_LOGIC_VECTOR(6 downto 0);
        res :   IN STD_LOGIC;
        outp    :   OUT STD_LOGIC
        );
End mppmDE;

Architecture bhv of mppmDE is
Signal  inreg           :   STD_LOGIC_VECTOR(6 downto 0):=(others=>'0');
Signal  outreg      :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  dreg            :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  creg            :   STD_LOGIC_VECTOR(6 downto 0):=(others=>'0');
Signal  control     :   STD_LOGIC_VECTOR(6 downto 0):="1100011";
Signal  temp            :   STD_LOGIC:='0';

Begin
Process(res,clk)
Begin
    IF(res='0')then
        inreg       <=  (others=>'0');
        outreg  <=  (others=>'0');
        dreg        <=  (others=>'0');
        creg        <=  (others=>'0');
        control <=  "1100011";
        temp        <='0';
    ELSIF(clk'event and clk='1')then
        control <=  control(0) & control (6 downto 1);
        inreg       <=  inp & inreg(6 downto 1);
        IF(control="1110001")then
            creg    <=  inreg;
            case inreg is
                when "1100000" => outreg    <=  "0000"; dreg    <=  "0000";
                when "1010000" => outreg    <= "0001";  dreg    <= "0001";
                when "1001000" => outreg    <= "0010";  dreg    <= "0010";
                when "1000100" => outreg    <= "0011";  dreg    <= "0011";
                when "1000010" => outreg    <= "0100";  dreg    <= "0100";
                when "1000001" => outreg    <= "0101";  dreg    <= "0101";
                when "0110000" => outreg    <= "0110";  dreg    <= "0110";
                when "0101000" => outreg    <= "0111";  dreg    <= "0111";
                when "0100100" => outreg    <= "1000";  dreg    <= "1000";
                when "0100010" => outreg    <= "1001";  dreg    <= "1001";
                when "0100001" => outreg    <= "1010";  dreg    <= "1010";
                when "0011000" => outreg    <= "1011";  dreg    <= "1011";
                when "0010100" => outreg    <= "1100";  dreg    <= "1100";
                when "0010010" => outreg    <= "1101";  dreg    <= "1101";
                when "0010001" => outreg    <= "1110";  dreg    <= "1110";
                when "0001100" => outreg    <= "1111";  dreg    <= "1111";
                when others     => outreg   <=  "0000"; dreg    <=  "0000";
            end case;
```

```
        ELSIF(control(0)='0')THEN
            outreg  <= '0' & outreg(3 downto 1);
        END IF;
        temp    <=  outreg(0);
    END IF;
End process;
dataword    <=  dreg;
codeword    <=  creg;
outp    <=  temp;
End bhv;
```

Code C4 (Sync)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY syncmod IS
    PORT        (
                    clk             :   IN STD_LOGIC;
                    res             :   IN STD_LOGIC;
                    sin             :   IN STD_LOGIC;
                    synced          :   OUT STD_LOGIC;
                    error_det       :   OUT STD_LOGIC
                    );
END syncmod;
ARCHITECTURE behaviour OF syncmod IS

SIGNAL reg15            :    STD_LOGIC_VECTOR(15 downto 0):=(others=>'0');
--input register
SIGNAL prbs             :    STD_LOGIC_VECTOR(15 downto 0):=(others=>'0');
-- PRBS register
SIGNAL control      :   STD_LOGIC_VECTOR(6 downto 0):="1110000";
SIGNAL sync_r       :   STD_LOGIC:='0';
SIGNAL temp         :   STD_LOGIC:='0';
BEGIN
PROCESS(res,clk)
BEGIN
    IF (res='0') THEN
        reg15       <=  (others=>'0');
        prbs        <=  (others=>'0');
        control <=  "1110000";
        sync_r  <=  '0';
        temp        <=  '0';

    ELSIF (clk'EVENT AND clk='1') THEN
        control<=   control(0) & control(6 downto 1);

        IF(control(4) = '0')THEN
            reg15 <=  reg15(14 downto 0) & sin;--input register serial input
            prbs   <= prbs(14 downto 0) & (prbs(15) XOR prbs(14));--PRBS
expression
            IF(reg15="111111111111111" AND sync_r='0') THEN --Wait for sync
                prbs    <=  "111111111111111" & sin;                    -
-To match it with input reg i.e. reg15
                sync_r<=    '1';
            ELSIF(sync_r='1') THEN
                temp       <=   reg15(0) XOR prbs(0);--new input at zero
place(reg15) & new seq for prbs at zero place.
            ELSE
                temp    <= '0';
            END IF;
         END IF;
    END IF;
END PROCESS;
error_det   <=temp;
synced      <=sync_r;   -- to indicate if the system has been synced
END behaviour;
```

Code C5 (Error counting and Inducing)

```
--      Error inducing and error counting system        --
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

--                                  ENTITY                              --

Entity ERRmod Is
    PORT    (   --clk3      :   IN STD_LOGIC;
                clk     :   IN STD_LOGIC;
                res     :   IN STD_LOGIC;
                sync        :   IN STD_LOGIC;
                enable  :   IN STD_LOGIC;
                error       :   IN STD_LOGIC;
                bit_sel :   IN  STD_LOGIC_VECTOR(2 downto 0);
                insert  :   OUT STD_LOGIC;
                count_error:    OUT STD_LOGIC_VECTOR(29 DOWNTO 0);
                err_ind :   OUT STD_LOGIC_VECTOR(29 downto 0);
                err_det :   OUT STD_LOGIC_VECTOR(29 downto 0)
            );

END ERRmod;

--                  ARCHITECTURE                    --

ARCHITECTURE behaviour of ERRmod Is

SIGNAL  control             :STD_LOGIC_VECTOR(6 downto 0):="1110000";
SIGNAL  ce                  :STD_LOGIC_VECTOR(6 downto 0):="1000000";
SIGNAL  induced     :STD_LOGIC_VECTOR(29 downto 0):=(others => '0');
SIGNAL  detected        :STD_LOGIC_VECTOR(29 downto 0):=(others => '0');
SIGNAL  temp1           :   STD_LOGIC_VECTOR(29 DOWNTO 0):=(others => '0');
SIGNAL  pulse_count :STD_LOGIC_VECTOR(29 downto 0):=(others => '0');
SIGNAL  prbs            :STD_LOGIC_VECTOR(6 downto 0):=(others => '1');
SIGNAL  err_bit     :STD_LOGIC;
SIGNAL  insert_bit  :STD_LOGIC;
SIGNAL  reg_error   :STD_LOGIC;
SIGNAL  reg_sync        :STD_LOGIC;

BEGIN
reg_sync    <=  sync;
reg_error<= error;
--This process operates at clk4, and induces the error by inverting the
respective bit
PROCESS(res,clk)
BEGIN
    IF(res='0')THEN
        ce              <=  "1000000";
        control     <=  "1110000";
        err_bit     <=  '0';
        insert_bit  <=  '0';
        prbs            <=(others   =>  '1');
        induced     <=(others   =>  '0');
        detected        <=(others   =>  '0');
        temp1           <=(others   =>  '0');
        pulse_count <=(others   =>  '0');
    ELSIF(clk'EVENT AND clk='1')THEN
```

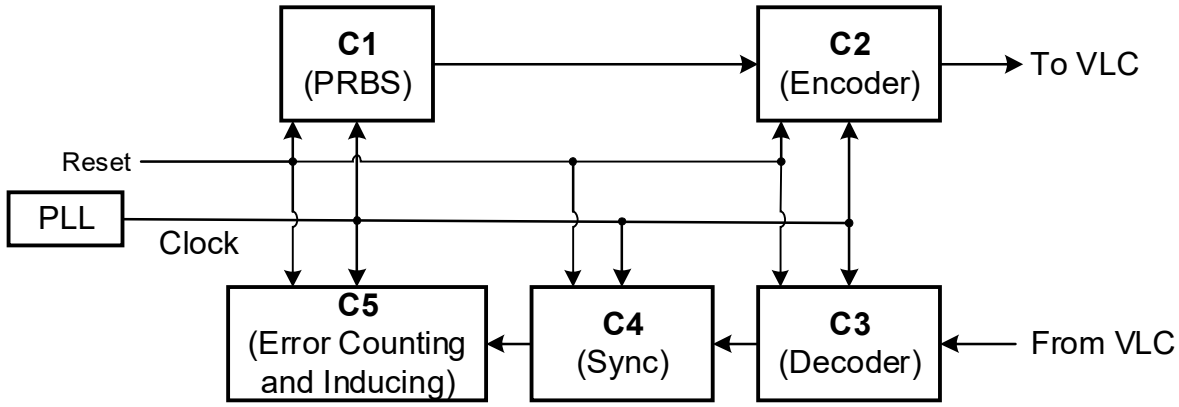
```
        control     <=  control(0) & control(6 downto 1);
        ce      <=  ce(0) & ce(6 downto 1);
        prbs    <=  prbs(5 downto 0) & (prbs(6) XOR prbs(5));

        IF(control="1110000")THEN
            err_bit <=  prbs(5) AND prbs(6) AND prbs(4) AND prbs(3) AND
prbs(2) AND prbs(1) AND prbs(0)AND (not enable) AND reg_sync;   --wait for
sync & enable
        END IF;
        --select where to insert error0
        case    bit_sel is
            when    "000"   =>  insert_bit  <= ce(6) AND err_bit;
            when    "001"   =>  insert_bit  <= ce(5) AND err_bit;
            when    "010"   =>  insert_bit  <= ce(4) AND err_bit;
            when    "011"   =>  insert_bit  <= ce(3) AND err_bit;
            when    "100"   =>  insert_bit  <= ce(2) AND err_bit;
            when    "101"   =>  insert_bit  <= ce(1) AND err_bit;
            when    "110"   =>  insert_bit  <= ce(0) AND err_bit;
            when    others  =>  insert_bit  <= ce(3) AND err_bit;
        end case;
        induced <=  induced + insert_bit;
        detected    <=  detected + reg_error;   -- +1
            IF(pulse_count=1000000000) THEN -1giga pulses
                temp1   <=  detected;
                pulse_count <=  (others =>  '0');
                detected        <=  (others =>  '0');
                induced     <=  (others =>  '0');
            else
                pulse_count <=  pulse_count +1;
            END IF;
    END IF;

END PROCESS;
count_error <=  detected;
err_ind     <=  induced;
err_det     <=  temp1;
insert      <=  insert_bit;
END behaviour;
```

Appendix E

Quartus software schematic and VHDL code for OPPM with design and implementation are given below:



Code C1 (PRBS)

```
--Offset PRBS using one clock
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;

Entity PRBS is
    PORT(
        clk :   IN STD_LOGIC;
        res :   IN STD_LOGIC;
        outp    :   OUT STD_LOGIC
        );
End PRBS;
Architecture bhv of PRBS is
Signal  control :STD_LOGIC_VECTOR(3 downto 0):="0001";--delayed
Signal  reg     :STD_LOGIC_VECTOR(14 downto 0):=(others=>'1'); --15bit
Signal  temp        :STD_LOGIC:='0';
Begin
Process(res,clk)
Begin
    IF(res='0')then
        temp        <='0';
        control <=  "0001";
        reg     <=  (others=>'1');
    ELSIF(clk'event and clk='1')then
        control <=  control(0) & control(3 downto 1);
        IF(control(0)='0')then
            reg <=  reg(13 downto 0) & (reg(14) XOR reg(13));
        END IF;
    END IF;
outp    <=  reg(14);
End process;
End bhv;
```

Code C2 (Encoder)

```
--OFFSET PPM encoder using one clock
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
                --++          ENTITY          ++--
ENTITY encoder34 IS
PORT    (   clk :   IN STD_LOGIC;
            res :   IN STD_LOGIC;
            inp :   IN STD_LOGIC;
            transmitted_cd  :   OUT STD_LOGIC_VECTOR(3 downto 0);
            transmitted_dw: OUT STD_LOGIC_VECTOR(2 downto 0);
            outp:   OUT STD_LOGIC);
END encoder34;
                --++         ARCHITECTURE         ++--
ARCHITECTURE behaviour OF encoder34 IS
SIGNAL control  : STD_LOGIC_VECTOR(3 downto 0):="1000"; --control reg
SIGNAL reg4 :STD_LOGIC_VECTOR(3 downto 0):=(others => '0');--4bit output reg
SIGNAL sipo:STD_LOGIC_VECTOR(2 downto 0):=(others => '0');  --7bit input reg
SIGNAL reg3     :   STD_LOGIC_VECTOR(2 downto 0):=(others => '0');
SIGNAL reg_cd   :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
SIGNAL out_reg  :   STD_LOGIC:='0';
SIGNAL in_reg   :   STD_LOGIC:='0';
BEGIN
PROCESS(res,clk)
BEGIN
    IF(res='0') THEN
        control     <=  "1100";
        sipo    <=  (others =>  '0');
        reg4    <=  (others =>  '0');
        reg3    <=  (others =>  '0');
        reg_cd  <=  (others=>'0');
        out_reg <=  '0';
    ELSIF(clk'EVENT AND clk='1') THEN
    sipo    <=  inp & sipo(2 downto 1);     --input -> serial in
            control<= control(0) & control(3 downto 1);             --control
reg
            IF(control="1100") THEN         --control regs synced
                reg3    <= sipo;
                CASE sipo IS                                    --encoding to 4bits
                    when "000" =>   reg4<="0000"; reg_cd<= "0000";
                    when "001" =>   reg4<="0001"; reg_cd<= "0001";
                    when "010" =>   reg4<="0010"; reg_cd<= "0010";
                    when "011" =>   reg4<="0100"; reg_cd<= "0100";
                    when "100" =>   reg4<="1000"; reg_cd<= "1000";
                    when "101" =>   reg4<="1001"; reg_cd<= "1001";
                    when "110" =>   reg4<="1010"; reg_cd<= "1010";
                    when "111" =>   reg4<="1100"; reg_cd<= "1100";
                end case;
            ELSE
                reg4<= '0' & reg4(3 downto 1);      --shift zeros
            END IF;
    END IF;
END PROCESS;
transmitted_cd  <=  reg_cd;
transmitted_dw  <= reg3;
outp <= reg4(0);
END behaviour;
```

Code C3 (Decoder)

```
--  OFFSET PPM Decoder
-- Decodes 4bits to 3bits
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
                                --++            ENTITY          ++--
ENTITY decoder43 IS
PORT (  inp :       IN STD_LOGIC;
            clk :       IN STD_LOGIC;
            res :       IN STD_LOGIC;
            rec_cd:     OUT STD_LOGIC_VECTOR(3 downto 0);
            rec_dw:     OUT STD_LOGIC_VECTOR(2 downto 0);
            outp:       OUT STD_LOGIC
        );
END decoder43;
                                --++        ARCHITECTURE        ++--
ARCHITECTURE behaviour OF decoder43 IS
SIGNAL  control     :   STD_LOGIC_VECTOR(3 downto 0):="1100";       --control
register
SIGNAL  reg3    :   STD_LOGIC_VECTOR(2 downto 0):=(others   =>  '0');       -
-output register
SIGNAL  sipo    :   STD_LOGIC_VECTOR(5 downto 0):=(others   =>  '0');       -
-7bit input reg- serial in
Signal  reg4    :   STD_LOGIC_VECTOR(2 downto 0):=(others    =>  '0');
signal  reg5    :   STD_LOGIC_VECTOR(3 downto 0):=(others    =>  '0');
SIGNAL  reg_rec:    STD_LOGIC_VECTOR(3 downto 0):=(others    =>  '0');
signal  out_reg:    std_logic;

BEGIN
--In this process, last 4bits of sipo reg are used to decode the message and
serially outputted
PROCESS(res,clk)
BEGIN
    IF(res='0')THEN
        sipo    <=(others   =>  '0');
        control         <="1100";
        reg3        <=(others   =>  '0');
        reg4        <=(others   =>  '0');
        reg5        <=(others   =>  '0');
        reg_rec <=(others   =>  '0');
    ELSIF (clk'EVENT AND clk='1') THEN
        control <= control(0) & control(3 downto 1);                    --
cotrol register\
    sipo    <= inp & sipo(5 downto 1);      --serial input
        IF (control="1001") THEN
            reg_rec <=  sipo(3 downto 0);   --original codeword received
            reg5    <=  sipo(3 downto 0);
            case reg5 is                    --decoding
                when "0000" =>  reg3 <= "000"; reg4 <= "000";
                when "0001" =>  reg3 <= "001";  reg4    <= "001";
                when "0010" =>  reg3 <= "010";  reg4    <= "010";
                when "0100" =>  reg3 <= "011";  reg4    <= "011";
                when "1000" =>  reg3 <= "100";  reg4    <= "100";
                when "1001" =>  reg3 <= "101";  reg4    <= "101";
                when "1010" =>  reg3 <= "110";  reg4    <= "110";
                when "1100" =>  reg3 <= "111";  reg4    <= "111";
                when others => reg3 <= "111";   reg4    <= "111";
            end case;
```

```
        ELSIF(control(3)='0')THEN
                reg3    <=  '0' & reg3(2 downto 1);          --shifting with
zeros->serial output
        END IF;
    END IF;
END PROCESS;
rec_cd  <=  reg_rec;
rec_dw  <=  reg4;
outp    <=  reg3(0);
END behaviour;
```

Code C4 (Sync)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY syncmod IS
    PORT        (   clk3                :   IN STD_LOGIC;
                    res             :   IN STD_LOGIC;
                    sin             :   IN STD_LOGIC;
                    synced          :   OUT STD_LOGIC;
                    error_det       :   OUT STD_LOGIC
                    );
END syncmod;
ARCHITECTURE behaviour OF syncmod IS
SIGNAL  reg15           :   STD_LOGIC_VECTOR(14 downto 0);          --input
register
SIGNAL  prbs            :   STD_LOGIC_VECTOR(14 downto 0);          -- PRBS
register
SIGNAl  control     :   STD_LOGIC_VECTOR(3 downto 0):="1000";
SIGNAL  sync_r      :   STD_LOGIC:='0';
signal  in_reg      :   std_logic:='0';
SIGNAL  temp            :   STD_LOGIC:='0';
signal  err_reg     : std_logic:='0';
signal  sync_reg        :   std_logic:='0';
BEGIN
in_reg  <=  sin;
PROCESS(res,clk3)
BEGIN
    IF (res='0') THEN
        reg15       <=  (others=>'0');
        prbs        <=  (others=>'0');
        control <=  "1000";
        sync_r  <=  '0';
        temp        <=  '0';
    ELSIF (clk3'EVENT AND clk3='1') THEN
    if(control(3)='0')then
        reg15<=  reg15(13 downto 0) & in_reg;--input register serial input
        prbs<= prbs(13 downto 0) & (prbs(13) XOR prbs(14)); --PRBS expression
        IF(reg15="111111111111111" AND sync_r='0') THEN --Wait for sync
            prbs    <=  (others=>'1');--To match it with input reg i.e. reg15
            sync_r<=    '1';
        ELSIF(sync_r='1') THEN
            temp    <=  reg15(0) XOR prbs(0);           --new input at zero
place(reg15) & new seq for prbs at zero place.
        ELSE
            temp    <= '0';
        END IF;
    end if;
    END IF;
    err_reg <=  temp;
    sync_reg    <=  sync_r;
END PROCESS;
error_det   <=err_reg;
synced      <=sync_reg;   -- to indicate if the system has been synced
END behaviour;
```

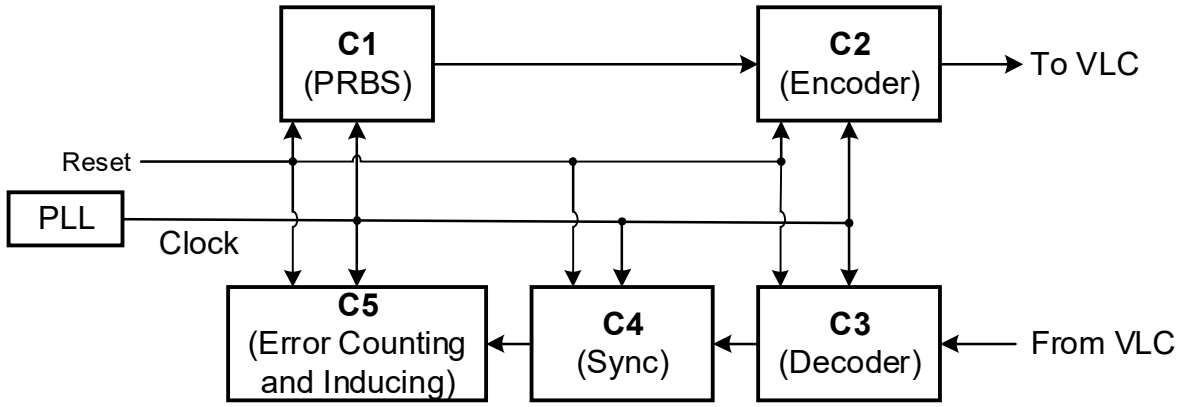
Code C5 (Error counting and inducing)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY syncmod IS
    PORT        (   clk3                :   IN STD_LOGIC;
                    res             :   IN STD_LOGIC;
                    sin             :   IN STD_LOGIC;
                    synced          :   OUT STD_LOGIC;
                    error_det       :   OUT STD_LOGIC
                    );
END syncmod;
ARCHITECTURE behaviour OF syncmod IS
SIGNAL  reg15           :   STD_LOGIC_VECTOR(14 downto 0); --input register
SIGNAL  prbs            :   STD_LOGIC_VECTOR(14 downto 0); -- PRBS register
SIGNAl  control     :   STD_LOGIC_VECTOR(3 downto 0):="1000";
SIGNAL  sync_r      :   STD_LOGIC:='0';
signal  in_reg      :   std_logic:='0';
SIGNAL  temp            :   STD_LOGIC:='0';
signal  err_reg     : std_logic:='0';
signal  sync_reg        :   std_logic:='0';
BEGIN
in_reg  <=  sin;
```

```
PROCESS(res,clk3)
BEGIN
    IF (res='0') THEN
        reg15       <=  (others=>'0');
        prbs        <=  (others=>'0');
        control <=  "1000";
        sync_r  <=  '0';
        temp        <=  '0';
    ELSIF (clk3'EVENT AND clk3='1') THEN
    if(control(3)='0')then
        reg15  <=  reg15(13 downto 0) & in_reg;--input register serial input
        prbs<= prbs(13 downto 0) & (prbs(13) XOR prbs(14)); --PRBS expression
        IF(reg15="111111111111111" AND sync_r='0') THEN      --Wait for sync
            prbs    <=  (others=>'1');                              --To match
it with input reg i.e. reg15
            sync_r<=    '1';
        ELSIF(sync_r='1') THEN
            temp    <=  reg15(0) XOR prbs(0);           --new input at zero
place(reg15) & new seq for prbs at zero place.
        ELSE
            temp    <= '0';
        END IF;
    end if;
    END IF;
    err_reg <=  temp;
    sync_reg    <=  sync_r;
END PROCESS;
error_det   <=err_reg;
synced      <=sync_reg;    -- to indicate if the system has been synced
END behaviour;
```

Appendix F

Simplified Quartus schematic and VHDL code for DiPPM design and implementation is given below:



Code C1 (PRBS)

```
--15bit PRBS generator
--Active low reset
Library IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
--                                                                    *ENTITY*
--
```

```
ENTITY PRBS IS
PORT    (
            clk     :   IN STD_LOGIC;
            res     :   IN STD_LOGIC;
            output  :   OUT STD_LOGIC);
END PRBS;
--                                                                *ARCHITECTURE*
--
```

```
ARCHITECTURE behaviour OF PRBS IS
SIGNAL prbs_reg :   STD_LOGIC_VECTOR(14 downto 0):=(others=>'1');
BEGIN
    PROCESS(clk,res)
    BEGIN
        IF(res='0') THEN
            prbs_reg    <=  (others=>   '1');
        ELSIF(clk'EVENT AND clk='1') THEN
        prbs_reg        <=    prbs_reg(13   downto   0)   &   (prbs_reg(14)   XOR
prbs_reg(13));
        END IF;
    END PROCESS;
output  <= prbs_reg(14);
END behaviour;
```

Code C2 (Encoder)

```
--DiPPM Encoder
--Active low reset
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;
--                          Entity                                      --
---------------------------------------------------------------------------
ENTITY encoder IS
        PORT    (   clk     :IN STD_LOGIC;
                    res     :IN STD_LOGIC;
                    input       :IN STD_LOGIC;
                    o_out       :OUT STD_LOGIC;
                    output  :OUT STD_LOGIC
                );
END encoder;
--                      Architecture                                    --
---------------------------------------------------------------------------
ARCHITECTURE behaviour OF encoder IS
SIGNAL  reset_reg:  STD_LOGIC_VECTOR(1 downto 0):="00";
SIGNAL  set_reg :   STD_LOGIC_VECTOR(1 downto 0):="00";
SIGNAL  D1          :   STD_LOGIC;
SIGNAL  D2          :   STD_LOGIC;
SIGNAL  out_reg :   STD_LOGIC:='0';
SIGNAL  count   :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
SIGNAL  prbs        :STD_LOGIC_VECTOR(6 downto 0):=(others=>'1');
BEGIN
PROCESS(clk,res)
BEGIN
    IF(res='0') THEN
        reset_reg   <=  "00";
        prbs            <="1111111";
    ELSIF(clk'EVENT AND clk='1')THEN
        reset_reg<= reset_reg(0) & input;
        prbs    <=  prbs(5 downto 0)& (prbs(6) xor (prbs(5)));
    END IF;

    D2  <=  (NOT(clk) AND (reset_reg(1)) AND NOT(reset_reg(0)));
END PROCESS;
---------------------------------------------------------------------------
PROCESS(res,clk)
BEGIN
    IF(res='0') THEN
        set_reg <=  "00";

    ELSIF(clk'EVENT AND clk='0') THEN
            set_reg<= set_reg(0) & input;
    END IF;

    D1  <=  (clk AND (NOT(set_reg(1))) AND set_reg(0));
END PROCESS;
PROCESS(D1,D2)
BEGIN
    o_out   <=  D1 or D2;
    out_reg <=  D1 OR D2;
END PROCESS;
output  <=  out_reg;
END behaviour;
```

Code C3 (Decoder)

```
--DiPPM Encoder
--Reset Active low
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
--                          *   ENTITY  *                --
ENTITY decoder IS
        PORT    (
                    clk     :   IN STD_LOGIC;
                    res     :   IN STD_LOGIC;
                    input       :   IN STD_LOGIC;
                    output  :   OUT STD_LOGIC
                );
END decoder;
--              *   ARCHITECTURE *                   --
ARCHITECTURE behaviour OF decoder IS
signal  temp:   STD_LOGIC:='0';
signal  t1  :   STD_LOGIC:='0';
signal  out_reg :   STD_LOGIC:='0';
Signal  reg_S:      STD_LOGIC_VECTOR(29 downto 0):=(others=>'0');
Signal  reg_R:      STD_LOGIC_VECTOR(29 downto 0):=(others=>'0');
signal  case_reg    :   STD_LOGIC_VECTOR(1 downto 0):="00";
begin
PROCESS(clk,res)
BEGIN
    IF(res='0') THEN
        temp    <=  '0';
        --reg_S     <=  (others=>'0');
    ElSIF(clk'EVENT AND clk='0') THEN
        temp    <=  input;      --set pulses
    END IF;
    t1<=temp;
END PROCESS;
PROCESS(clk,res)
BEGIN--dectecting pulses to apply MLSD
    IF (res='0')THEN
        case_reg    <=  "00";
        reg_R       <=  (others=>'0');
        reg_S       <=  (others=>'0');
    ELSIF(clk'EVENT AND clk='1') THEN

        reg_S   <=  (t1 & reg_S(29 downto 1));
        reg_R   <=  (input & reg_R(29 downto 1));

        case_reg    <=  reg_S(0) & reg_R(0);

        case case_reg is
            when "00" => out_reg <= out_reg;
            when "01" => out_reg    <=  '0';
            when "10" => out_reg    <=  '1';
            when "11" => out_reg    <=  '0';
        end case;
    end if;
end process;
output  <=  out_reg;
END behaviour;
```

Code C4 (Sync)

```
--Rxmod receives input from decoder
--Outputs synced and error_det
--To check if system is synced and errors in the output
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
--                      ***          ENTITY       ***                          --
ENTITY Rxmod IS
    PORT         (    clk              :    IN STD_LOGIC;
                      res              :    IN STD_LOGIC;
                      sin              :    IN STD_LOGIC;
                      synced           :    OUT STD_LOGIC;
                      error_det        :    OUT STD_LOGIC
                      );
END Rxmod;
--                 *** ARCHITECTURE     ***                                  --
ARCHITECTURE behaviour OF Rxmod IS

SIGNAL reg15         :    STD_LOGIC_VECTOR(14 downto 0); --input register
SIGNAL prbs          :    STD_LOGIC_VECTOR(14 downto 0); -- PRBS register
SIGNAL sync_r        :    STD_LOGIC:='0';
SIGNAL temp          :    STD_LOGIC:='0';
BEGIN
PROCESS(res,clk)
BEGIN
    IF (res='0') THEN
        reg15    <=  (others =>  '0');
        prbs     <=  (others =>  '0');
        sync_r<=     '0';
        temp     <=  '0';
    ELSIF (clk'EVENT AND clk='1') THEN
        reg15 <=  reg15(13 downto 0) & sin;  --input register serial input
        prbs <= prbs(13 downto 0) & (prbs(13) XOR prbs(14));--PRBS expression
        IF(reg15="111111111111111" AND sync_r='0') THEN --Wait for sync
            Prbs <="111111111111111"& sin;     --To match it with input reg
i.e. reg15
            sync_r<=     '1';
        ELSIF(sync_r='1') THEN
            --temp  <=  sin XOR prbs(0);
            temp       <=    reg15(0)  XOR  prbs(0);     --new  input  at  zero
place(reg15) & new seq for prbs at zero place.
        ELSE
            temp     <= '0';
        END IF;

    END IF;

END PROCESS;
error_det    <=temp;
synced       <=  sync_r;
END behaviour;
```

Code C5 (Error counting and inducing)

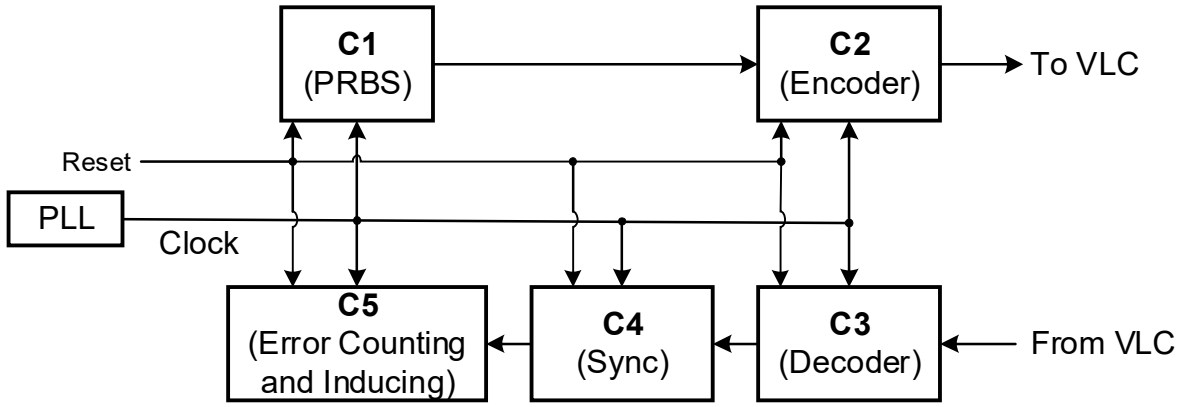
```
--DiPPM Error module
--Active low reset & enable error controls
--Counts the total number of errors induced and detected
--Induces error in the system
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
--                              ***      ENTITY      ***                    --
ENTITY ERRmod IS
    PORT    (   detect_error        :   IN STD_LOGIC;
                synced              :   IN STD_LOGIC;
                clk                 :   IN STD_LOGIC;
                res                 :   IN STD_LOGIC;
                enable              :   IN STD_LOGIC;
                count_error         :   OUT STD_LOGIC_VECTOR(29 DOWNTO 0);
                error_detected      :   OUT STD_LOGIC_VECTOR(29 DOWNTO 0);
                error_induced       :   OUT STD_LOGIC_VECTOR(29 DOWNTO 0);
                insert              :   OUT STD_LOGIC
            );
END ERRmod;
--                              *** ARCHITECTURE    ***                 --
ARCHITECTURE behaviour OF ERRmod IS
SIGNAL  reg_pulses      :   STD_LOGIC_VECTOR(29 DOWNTO 0);
SIGNAL  reg_detected    :   STD_LOGIC_VECTOR(29 DOWNTO 0);
SIGNAL  reg_induced     :   STD_LOGIC_VECTOR(29 DOWNTO 0);
SIGNAL  temp1               :   STD_LOGIC_VECTOR(29 DOWNTO 0);
SIGNAL  PRBS                :   STD_LOGIC_VECTOR(6 DOWNTO 0);
SIGNAL  reg_insert      :   STD_LOGIC;
SIGNAL  temp                :   STD_LOGIC;
BEGIN
PROCESS(res,clk)
BEGIN
    IF(res='0') THEN
        reg_induced <=      (others => '0');
        reg_detected<=      (others => '0');
        PRBS            <=      (others =>  '1');
        temp1           <=      (others =>  '0');
    ELSIF(clk'EVENT AND clk='1') THEN
        --pulse_reg <=  pulse_reg(0)    & reg_insert;
        PRBS            <= PRBS(5 DOWNTO 0) & (PRBS(6) XOR PRBS(5));
        reg_insert  <= PRBS(6) AND PRBS(4) AND  PRBS(1) AND (not PRBS(3))
AND ( PRBS(5)) AND PRBS(2) AND synced AND (NOT enable);
        reg_pulses  <=  reg_pulses  +1;
        IF(reg_pulses=1000000000) THEN
            reg_pulses      <= (others  => '0');
            temp1               <=  reg_detected;
            reg_detected    <=  (others => '0');
            reg_induced     <=  (others => '0');
        END IF;
        IF(detect_error='1') THEN
            reg_detected    <= reg_detected +1;
        END IF;
        reg_induced <=  reg_induced + reg_insert;
    END IF;
    temp    <=      (( clk) AND reg_insert);--reset pulse
END PROCESS;

count_error     <=  reg_detected;
```

```
error_induced   <=  reg_induced;
error_detected  <=  temp1;
insert  <= temp;
END behaviour;
```

Appendix G

Simplified Quartus schematic and VHDL code for DuoPPM design and implementation is given below:



Code C1 (PRBS)

It uses the same code as given in Appendix F.

Code C2 (Encoder)

```
--DuoBinary Encoder
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
--                              **ENTITY**                              -
-
ENTITY duoencoder IS
        PORT    (
                    clk     :   IN      STD_LOGIC;
                    res     :   IN  STD_LOGIC;
                    sin     :   IN      STD_LOGIC;
                    output  :   OUT STD_LOGIC
                );
END duoencoder;
--                          **ARCHITECTURE**                            -
-
ARCHITECTURE behaviour OF duoencoder IS
SIGNAL  reg1        :STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL  out_reg :STD_LOGIC;
SIGNAL  temp        :STD_LOGIC;

BEGIN

PROCESS(res,clk)
```

```
BEGIN
    IF(res='0')THEN
        reg1        <="00";
    ElSIF(clk'EVENT AND clk='1')THEN
        reg1    <=  reg1(0) & sin;
    END IF;

END PROCESS;
PROCESS(clk)
BEGIN
        case reg1 is
            when "11" => out_reg    <=  clk;
            when "00" => out_reg    <= not clk;
            when others => out_reg <= '0';
        end case;
        temp    <=  out_reg;
END PROCESS;
output  <=  out_reg;
END behaviour;
```

Code C3 (Decoder)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY duodecoder IS
        PORT    (
                    input       :IN STD_LOGIC;
                    clk     :IN     STD_LOGIC;
                    res     :IN STD_LOGIC;
                    output  :OUT    STD_LOGIC
                );
END duodecoder;
ARCHITECTURE behaviour OF duodecoder IS
SIGNAL reg1     :STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL D1       :STD_LOGIC;
SIGNAL D2       :STD_LOGIC;
Signal out_reg  :std_logic;
BEGIN
PROCESS(clk,res)
BEGIN
    IF(res='0')THEN
        D2      <='0';
        reg1    <=  (others => '0');
    ELSIF(clk'EVENT AND clk='1')THEN
        reg1    <= D1 & input;  -- swap places
        case reg1 is
            when "00"   =>  out_reg <=  not out_reg;
            when "01"   => out_reg  <= '0';
            when "10"   =>  out_reg <=  '1';
            when "11"   =>  out_reg <=  out_reg;
        end case;
    END IF;
END PROCESS;
```

```
PROCESS(clk,res)
BEGIN
    IF(res='0')THEN
        D1      <='0';
    ELSIF(clk'EVENT AND clk='0')THEN
        D1  <= input;
    END IF;
END PROCESS;
```

```
output  <=  out_reg;
END behaviour;
```

Code C4 (Sync)

It uses the same code as given in Appendix F.

Code C5 (Error counting and inducing)

It uses the same code as given in Appendix F.

Appendix H

VHDL code for Modified MPPM design and implementation is given below:

Schematic:

Identical schematic to the original MPPM. See Appendix D.

Modified MPPM PRBS VHDL:

It has identical code to the original MPPM PRBS, as given in appendix D.

Modified MPPM Encoder VHDL:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;

Entity mppmEN62 is
    PORT(
        clk :   IN STD_LOGIC;
        dataword    :   OUT STD_LOGIC_VECTOR(3 downto 0);
        codeword    :   OUT STD_LOGIC_VECTOR(5 downto 0);
        inp :   IN STD_LOGIC;
        res :   IN STD_LOGIC;
        outp    :   OUT STD_LOGIC
        );
End mppmEN62;

Architecture bhv of mppmEN62 is
Signal  inreg           :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  dreg            :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  outreg      :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');
Signal  creg            :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');
Signal  control     :   STD_LOGIC_VECTOR(5 downto 0):="110000";
Signal  temp            :   STD_LOGIC:='0';

Begin
Process(res,clk)
Begin
    IF(res='0')then
        inreg       <=  (others=>'0');
        outreg  <=  (others=>'0');
        dreg        <=  (others=>'0');
        creg        <=  (others=>'0');
        control <=  "110000";
        temp        <='0';
    ELSIF(clk'event and clk='1')then
        control <=  control(0) & control (5 downto 1);
        IF(control(0)='0')then
            inreg       <=  inp & inreg(3 downto 1);
        END IF;
```

```
        IF(control="110000")then
            dreg    <=  inreg;
            case inreg is
                when "0000" => outreg   <=  "000000";   creg    <=  "000000";
                when "0001" => outreg   <= "110000";    creg    <= "110000";
                when "0010" => outreg   <= "101000";    creg    <= "101000";
                when "0011" => outreg   <= "100100";    creg    <= "100100";
                when "0100" => outreg   <= "100010";    creg    <= "100010";
                when "0101" => outreg   <= "100001";    creg    <= "100001";
                when "0110" => outreg   <= "011000";    creg    <= "011000";
                when "0111" => outreg   <= "010100";    creg    <= "010100";
                when "1000" => outreg   <= "010010";    creg    <= "010010";
                when "1001" => outreg   <= "010001";    creg    <= "010001";
                when "1010" => outreg   <= "001100";    creg    <= "001100";
                when "1011" => outreg   <= "001010";    creg    <= "001010";
                when "1100" => outreg   <= "001001";    creg    <= "001001";
                when "1101" => outreg   <= "000110";    creg    <= "000110";
                when "1110" => outreg   <= "000101";    creg    <= "000101";
                when "1111" => outreg   <= "000011";    creg    <= "000011";
            end case;
        ELSE
            outreg  <= '0' & outreg(5 downto 1);
        END IF;
    END IF;
temp    <=  outreg(0);
End process;
dataword<=  dreg;
codeword<=  creg;
outp    <=  temp;
End bhv;
```

Modified MPPM Decoder VHDL code:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;

Entity mppmDE62 is
    PORT(
        clk :   IN STD_LOGIC;
        inp :   IN STD_LOGIC;
        dataword    :   OUT STD_LOGIC_VECTOR(3 downto 0);
        codeword    :   OUT STD_LOGIC_VECTOR(5 downto 0);
        res :   IN STD_LOGIC;
        outp    :   OUT STD_LOGIC
        );
End mppmDE62;

Architecture bhv of mppmDE62 is
Signal  inreg           :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');
Signal  outreg      :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  dreg            :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  creg            :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');
Signal  control     :   STD_LOGIC_VECTOR(5 downto 0):="110000";
Signal  temp            :   STD_LOGIC:='0';

Begin
Process(res,clk)
Begin
    IF(res='0')then
        inreg       <=  (others=>'0');
        outreg  <=  (others=>'0');
        dreg        <=  (others=>'0');
        creg        <=  (others=>'0');
        control <=  "110000";
        temp        <='0';
    ELSIF(clk'event and clk='1')then
        control <=  control(0) & control (5 downto 1);
        inreg       <=  inp & inreg(5 downto 1);
        IF(control="001100")then        -- **on fpga 011000
            creg    <=  inreg;
            case inreg is
                when "000000" => outreg <=  "0000"; dreg    <=  "0000";
                when "110000" => outreg <= "0001";  dreg    <= "0001";
                when "101000" => outreg <= "0010";  dreg    <= "0010";
                when "100100" => outreg <= "0011";  dreg    <= "0011";
                when "100010" => outreg <= "0100";  dreg    <= "0100";
                when "100001" => outreg <= "0101";  dreg    <= "0101";
                when "011000" => outreg <= "0110";  dreg    <= "0110";
                when "010100" => outreg <= "0111";  dreg    <= "0111";
                when "010010" => outreg <= "1000";  dreg    <= "1000";
                when "010001" => outreg <= "1001";  dreg    <= "1001";
                when "001100" => outreg <= "1010";  dreg    <= "1010";
                when "001010" => outreg <= "1011";  dreg    <= "1011";
                when "001001" => outreg <= "1100";  dreg    <= "1100";
                when "000110" => outreg <= "1101";  dreg    <= "1101";
                when "000101" => outreg <= "1110";  dreg    <= "1110";
                when "000011" => outreg <= "1111";  dreg    <= "1111";
                when others   => outreg <=  "1111"; dreg    <=  "1111";
            end case;
        ELSIF(control(0)='0')THEN           --**on fpga=control(1)=0
```

```
            outreg  <= '0' & outreg(3 downto 1);
        END IF;
        temp    <=  outreg(0);
    END IF;
End process;
dataword    <=  dreg;
codeword    <=  creg;
outp    <=  temp;
End bhv;
```

Modified MPPM Sync module VHDL:

It uses an identical code to the original MPPM synchronisation module, as given in appendix D.

Modified MPPM Error counting and inducing VHDL:

It uses an identical code to the original error module, as given in appendix D.

Appendix I

Priority decoder for OPPM VHDL code:

```
--  OFFSET PPM Decoder
-- Decodes 4bits to 3bits
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------------------------------------
                                --++            ENTITY          ++--
----------------------------------------------------------------------------------
ENTITY decoder43 IS
PORT (  input   :       IN STD_LOGIC;
            clk3    :       IN STD_LOGIC;
            clk4    :       IN STD_LOGIC;
            res :       IN STD_LOGIC;
            output:     OUT STD_LOGIC
        );
END decoder43;
----------------------------------------------------------------------------------
                                --++        ARCHITECTURE        ++--
----------------------------------------------------------------------------------
ARCHITECTURE behaviour OF decoder43 IS
SIGNAL  c3      :   STD_LOGIC_VECTOR(2 downto 0):="100";        --control
register
SIGNAL  reg3    :   STD_LOGIC_VECTOR(2 downto 0):=(others   =>  '0');
--output register
SIGNAL  sipo    :   STD_LOGIC_VECTOR(5 downto 0):=(others   =>  '0');
--7bit input reg- serial in
Signal  reg4    :   STD_LOGIC_VECTOR(3 downto 0):=(others   =>  '0');
signal  reg5    :   STD_LOGIC_VECTOR(3 downto 0):=(others   =>  '0');

BEGIN
-------------------------------------------------------------------------
--In this process, the encoded data from the encoder is received in the
system in sipo reg
-------------------------------------------------------------------------
PROCESS(res,clk4)
BEGIN
    IF(res='0') THEN
        sipo    <=(others   =>  '0');
    ELSIF(clk4'EVENT AND clk4='1') THEN
        sipo    <= input & sipo(5 downto 1);        --serial input
    END IF;
END PROCESS;

----------------------------------------------------------------------------------
--In this process, the last 4bits of sipo reg are used to decode the
message and serially outputted
----------------------------------------------------------------------------------
PROCESS(res,clk3)
BEGIN
    IF(res='0')THEN
        c3          <="100";
        reg3        <=(others   =>  '0');
        reg4        <=(others   =>  '0');
        reg5        <=(others   =>  '0');
    ELSIF (clk3'EVENT AND clk3='1') THEN
        c3  <= c3(0) & c3(2 downto 1);    --cotrol register\
----------------------------------------------------------------------------------
```

```
        IF (c3(2)='1') THEN
            reg4    <=  sipo(3 downto 0);
-------------------------------------------------------------------------
            reg5(3) <=  reg4(3);
            IF(reg4(1)='1')THEN
                reg5(2 downto 0)    <=  "010";
            elsif(reg4(2)='1')then
                reg5(2 downto 0)    <=  "100";
            elsif(reg4(0)='1')then
                reg5(2 downto 0)    <=  "001";
            else
            reg5(2 downto 0)    <=  reg4(2 downto 0);
            END IF;
-------------------------------------------------------------------------
            reg5    <=  reg4;
            case reg5 is                        --decoding
                when "0000" =>  reg3 <= "000";
                when "0001" =>  reg3 <= "001";
                when "0010" =>  reg3 <= "010";
                when "0100" =>  reg3 <= "011";
                when "1000" =>  reg3 <= "100";
                when "1001" =>  reg3 <= "101";
                when "1010" =>  reg3 <= "110";
                when "1100" =>  reg3 <= "111";
                when others => reg3 <= "111";
            end case;
        ELSE
                reg3    <=  '0' & reg3(2 downto 1);         --shifting with
zeros->serial output
        END IF;
    END IF;
END PROCESS;

output  <=  reg3(0);
END behaviour;
```

Modified MPPM Decoder with CRC and Parity check: VHDL code

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;

Entity mppmde64 is
    PORT(
        clk     :   IN STD_LOGIC;
        inp     :   IN STD_LOGIC;
        dataword    :   OUT STD_LOGIC_VECTOR(3 downto 0);
        codeword    :   OUT STD_LOGIC_VECTOR(5 downto 0);
        res     :   IN STD_LOGIC;
        checksum    :   OUT STD_LOGIC_VECTOR(3 downto 0);
        outp        :   OUT STD_LOGIC
        );
End mppmde64;

Architecture bhv of mppmde64 is
Signal  inreg           :   STD_LOGIC_VECTOR(40 downto 0):=(others=>'0');
--with vlc control(28 to 0)

--without vlc control(29..0)
Signal  AC              :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  sum_reg     :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  outreg      :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  dreg            :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  creg            :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');
Signal  control     :   STD_LOGIC_VECTOR(41 downto
0):="11111111000011000011000011000011000011000110000";
Signal  temp            :   STD_LOGIC:='0';
Signal  reg20           :   STD_LOGIC_VECTOR(23 downto 0):=(others=>'0');
Signal  parity      :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');
SIgnal  p_reg           :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');

Begin
Process(res,clk)
Begin
    IF(res='0')then
        AC          <=  (others=>'0');
        sum_reg <=  (others=>'0');
        inreg       <=  (others=>'0');
        reg20       <=  (others=>'0');
        outreg  <=  (others=>'0');
        dreg        <=  (others=>'0');
        creg        <=  (others=>'0');
        control <= "11111111000011000011000011000011000110000";
        temp        <= '0';
    ELSIF(clk'event and clk='1')then
        control <=  control(0) & control (41 downto 1);
        inreg       <=  inp & inreg(40 downto 1);       --without vlc
--      inreg       <=  inp & inreg(28 downto 1);   --with vlc
        IF(control(5 downto 0)="011000" or control(5 downto0)="111000")then
            p_reg   <=  parity(5) & p_reg(5 downto 1);
            creg    <=  inreg(5 downto 0);
            IF(parity(5)='0')THEN
                case inreg(5 downto 0) is
when "000000" => outreg <=  "0000"; dreg    <=  "0000"; reg20   <=  "0000"
& reg20(20 downto 1);    AC <= AC+"0000";
```

```
when "110000" => outreg <= "0001";  dreg    <= "0001";  reg20   <=  "0001"
& reg20(20 downto 1);    AC <= AC+"0001";
when "101000" => outreg <= "0010";  dreg    <= "0010";  reg20   <=  "0010"
& reg20(20 downto 1);    AC <= AC+"0010";
when "100100" => outreg <= "0011";  dreg    <= "0011";  reg20   <=  "0011"
& reg20(20 downto 1);    AC <= AC+"0011";
when "100010" => outreg <= "0100";  dreg    <= "0100";  reg20   <=  "0100"
& reg20(20 downto 1);    AC <= AC+"0100";
when "100001" => outreg <= "0101";  dreg    <= "0101";  reg20   <=  "0101"
& reg20(20 downto 1);    AC <= AC+"0101";
when "011000" => outreg <= "0110";  dreg    <= "0110";  reg20   <=  "0110"
& reg20(20 downto 1);    AC <= AC+"0110";
when "010100" => outreg <= "0111";  dreg    <= "0111";  reg20   <=  "0111"
& reg20(20 downto 1);    AC <= AC+"0111";
when "010010" => outreg <= "1000";  dreg    <= "1000";  reg20   <=  "1000"
& reg20(20 downto 1);    AC <= AC+"1000";
when "010001" => outreg <= "1001";  dreg    <= "1001";  reg20   <=  "1001"
& reg20(20 downto 1);    AC <= AC+"1001";
when "001100" => outreg <= "1010";  dreg    <= "1010";  reg20   <=  "1010"
& reg20(20 downto 1);    AC <= AC+"1010";
when "001010" => outreg <= "1011";  dreg    <= "1011";  reg20   <=  "1011"
& reg20(20 downto 1);    AC <= AC+"1011";
when "001001" => outreg <= "1100";  dreg    <= "1100";  reg20   <=  "1100"
& reg20(20 downto 1);    AC <= AC+"1100";
when "000110" => outreg <= "1101";  dreg    <= "1101";  reg20   <=  "1101"
& reg20(20 downto 1);    AC <= AC+"1101";
when "000101" => outreg <= "1110";  dreg    <= "1110";  reg20   <=  "1110"
& reg20(20 downto 1);    AC <= AC+"1110";
when "000011" => outreg <= "1111";  dreg    <= "1111";  reg20   <=  "1111"
& reg20(20 downto 1);    AC <= AC+"1111";
when others   => outreg <= "1111";  dreg    <= "1111";  reg20   <=  "1111"
& reg20(20 downto 1);    AC <= AC+"1111";
                end case;
            ELSE
                outreg  <="0000";
                reg20       <="0000" & reg20(20 downto 1);
                dreg        <="0000";
            END IF;
        ELSIF(control(5 downto 0)="011111")THEN
            case inreg(5 downto 0) is
                    when "000000" => AC <= AC + "0000";
                    when "110000" => AC <= AC + "0001";
                    when "101000" => AC <= AC + "0010";
                    when "100100" => AC <= AC + "0011";
                    when "100010" => AC <= AC + "0100";
                    when "100001" => AC <= AC + "0101";
                    when "011000" => AC <= AC + "0110";
                    when "010100" => AC <= AC + "0111";
                    when "010010" => AC <= AC + "1000";
                    when "010001" => AC <= AC + "1001";
                    when "001100" => AC <= AC + "1010";
                    when "001010" => AC <= AC + "1011";
                    when "001001" => AC <= AC + "1100";
                    when "000110" => AC <= AC + "1101";
                    when "000101" => AC <= AC + "1110";
                    when "000011" => AC <= AC + "1111";
                    when others   => AC <= AC + "1111";
                end case;
        ELSIF(control(5 downto 0)="001111")THEN
        sum_reg<= (not AC)+1;
            case p_reg is
```

```
when "100000" => reg20  <= ((not AC)+1) & reg20(19 downto 0);
p_reg   <= (others=>'0'); AC<=(others=>'0');
when "010000" => reg20  <= reg20(23 downto 20) & ((not AC)+1) & reg20(15
downto 0); p_reg   <= (others=>'0'); AC<=(others=>'0');
when "001000" => reg20  <= reg20(23 downto 16) & ((not AC)+1) & reg20(11
downto 0); p_reg   <= (others=>'0'); AC<=(others=>'0');
when "000100" => reg20  <= reg20(23 downto 12) & ((not AC)+1) & reg20(7
downto 0);  p_reg   <= (others=>'0'); AC<=(others=>'0');
when "000010" => reg20  <= reg20(23 downto 8) &  ((not AC)+1) & reg20(3
downto 0);  p_reg   <= (others=>'0'); AC<=(others=>'0');
when "000001" => reg20  <= reg20(23 downto 4) &  ((not AC)+1);
p_reg   <= (others=>'0'); AC<=(others=>'0');
when others     => p_reg    <= (others=>'0'); AC<=(others=>'0');
            end case;
        ELSIF(control(2)='0')THEN
        reg20   <=  '0' & reg20(23 downto 1);
        END IF;
        temp    <=  reg20(0);
    END IF;
End process;
parity(0) <= inreg(0);
gen:    for  i in 1 to 5 generate
            parity(i)   <=  parity(i-1) xor inreg(i);
        end generate gen;
checksum        <= sum_reg;
dataword    <=  dreg;
codeword    <=  creg;
outp        <=  temp;
End bhv;
```

Appendix J

Modified MPPM Encoder with CRC and Parity check VHDL code:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;

Entity mppmde64o is
    PORT(
        clk     :   IN STD_LOGIC;
        inp     :   IN STD_LOGIC;
        dataword    :   OUT STD_LOGIC_VECTOR(3 downto 0);
        codeword    :   OUT STD_LOGIC_VECTOR(5 downto 0);
        res     :   IN STD_LOGIC;
        checksum        :   OUT STD_LOGIC_VECTOR(3 downto 0);
        outp        :   OUT STD_LOGIC
        );
End mppmde64o;

Architecture bhv of mppmde64o is
Signal  inreg           :   STD_LOGIC_VECTOR(40 downto 0):=(others=>'0');
--with vlc control(28 to 0)

--without vlc control(29..0)
Signal  AC              :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  sum_reg     :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  outreg      :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  dreg            :   STD_LOGIC_VECTOR(3 downto 0):=(others=>'0');
Signal  creg            :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');
Signal  control     :   STD_LOGIC_VECTOR(41 downto
0):="11111111000011000011000011000011000011000110000";
Signal  temp            :   STD_LOGIC:='0';
Signal  reg20           :   STD_LOGIC_VECTOR(23 downto 0):=(others=>'0');
Signal  parity      :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');
SIgnal  p_reg           :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');

Begin
Process(res,clk)
Begin
    IF(res='0')then
        AC          <=  (others=>'0');
        sum_reg <=  (others=>'0');
        inreg       <=  (others=>'0');
        reg20       <=  (others=>'0');
        outreg  <=  (others=>'0');
        dreg        <=  (others=>'0');
        creg        <=  (others=>'0');
        --parity    <=  '0';
        control <= "11111111000011000011000011000011000110000";
        temp        <= '0';
    ELSIF(clk'event and clk='1')then
        control <=  control(0) & control (41 downto 1);
        inreg       <=  inp & inreg(40 downto 1);       --without vlc (29)
--      inreg       <=  inp & inreg(28 downto 1);   --with vlc
        IF(control(5 downto 0)="011000" or control(5 downto
0)="111000")then        -- **on fpga 011000
            --p_reg <=  parity(5) & p_reg(5 downto 1);
            creg    <=  inreg(5 downto 0);
            --IF(parity(5)='0')THEN
```

```
                case inreg(5 downto 0) is
                    when "000000" => outreg <=  "0000"; dreg    <=  "0000";
reg20   <=  "0000" & reg20(20 downto 1);    AC <= AC+"0000";
                    when "110000" => outreg <= "0001";  dreg    <= "0001";
reg20   <=  "0001" & reg20(20 downto 1);    AC <= AC+"0001";
                    when "101000" => outreg <= "0010";  dreg    <= "0010";
reg20   <=  "0010" & reg20(20 downto 1);    AC <= AC+"0010";
                    when "100100" => outreg <= "0011";  dreg    <= "0011";
reg20   <=  "0011" & reg20(20 downto 1);    AC <= AC+"0011";
                    when "100010" => outreg <= "0100";  dreg    <= "0100";
reg20   <=  "0100" & reg20(20 downto 1);    AC <= AC+"0100";
                    when "100001" => outreg <= "0101";  dreg    <= "0101";
reg20   <=  "0101" & reg20(20 downto 1);    AC <= AC+"0101";
                    when "011000" => outreg <= "0110";  dreg    <= "0110";
reg20   <=  "0110" & reg20(20 downto 1);    AC <= AC+"0110";
                    when "010100" => outreg <= "0111";  dreg    <= "0111";
reg20   <=  "0111" & reg20(20 downto 1);    AC <= AC+"0111";
                    when "010010" => outreg <= "1000";  dreg    <= "1000";
reg20   <=  "1000" & reg20(20 downto 1);    AC <= AC+"1000";
                    when "010001" => outreg <= "1001";  dreg    <= "1001";
reg20   <=  "1001" & reg20(20 downto 1);    AC <= AC+"1001";
                    when "001100" => outreg <= "1010";  dreg    <= "1010";
reg20   <=  "1010" & reg20(20 downto 1);    AC <= AC+"1010";
                    when "001010" => outreg <= "1011";  dreg    <= "1011";
reg20   <=  "1011" & reg20(20 downto 1);    AC <= AC+"1011";
                    when "001001" => outreg <= "1100";  dreg    <= "1100";
reg20   <=  "1100" & reg20(20 downto 1);    AC <= AC+"1100";
                    when "000110" => outreg <= "1101";  dreg    <= "1101";
reg20   <=  "1101" & reg20(20 downto 1);    AC <= AC+"1101";
                    when "000101" => outreg <= "1110";  dreg    <= "1110";
reg20   <=  "1110" & reg20(20 downto 1);    AC <= AC+"1110";
                    when "000011" => outreg <= "1111";  dreg    <= "1111";
reg20   <=  "1111" & reg20(20 downto 1);    AC <= AC+"1111";
                    when others   => outreg <= "1111";  dreg    <= "1111";
reg20   <=  "1111" & reg20(20 downto 1);    AC <= AC+"1111";
end case;
        ELSIF(control(5 downto 0)="011111")THEN
            case inreg(5 downto 0) is
                    when "000000" => AC <= AC + "0000";
                    when "110000" => AC <= AC + "0001";
                    when "101000" => AC <= AC + "0010";
                    when "100100" => AC <= AC + "0011";
                    when "100010" => AC <= AC + "0100";
                    when "100001" => AC <= AC + "0101";
                    when "011000" => AC <= AC + "0110";
                    when "010100" => AC <= AC + "0111";
                    when "010010" => AC <= AC + "1000";
                    when "010001" => AC <= AC + "1001";
                    when "001100" => AC <= AC + "1010";
                    when "001010" => AC <= AC + "1011";
                    when "001001" => AC <= AC + "1100";
                    when "000110" => AC <= AC + "1101";
                    when "000101" => AC <= AC + "1110";
                    when "000011" => AC <= AC + "1111";
                    when others   => AC <= AC + "1111";
                end case;
        ELSIF(control(5 downto 0)="001111")THEN
            sum_reg<= (not AC)+1;
        ELSIF(control(2)='0')THEN
        reg20   <=  '0' & reg20(23 downto 1);
        END IF;
```

```
        temp    <=  reg20(0);
    END IF;
End process;
checksum        <= sum_reg;
dataword    <=  dreg;
codeword    <=  creg;
outp        <=  temp;
End bhv;
```

Appendix K
MLSD VHDL code:

```
--Reset Active low
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
--------------------------------------------------------------------------
--                    *   ENTITY   *                --
--------------------------------------------------------------------------
ENTITY decoder IS
        PORT    (
                    clk     :   IN STD_LOGIC;
                    res     :   IN STD_LOGIC;
                    input       :   IN STD_LOGIC;
                    output  :   OUT STD_LOGIC
                );
END decoder;
--------------------------------------------------------------------------
--                        *   ARCHITECTURE *                    --
--------------------------------------------------------------------------
ARCHITECTURE behaviour OF decoder IS
signal  s_pulse:    STD_LOGIC:='0';
signal  r_pulse:    STD_LOGIC:='0';
signal  t1  :   STD_LOGIC:='0';
signal  out_reg :   STD_LOGIC:='0';
Signal  reg_S:      STD_LOGIC_VECTOR(29 downto 0):=(others=>'0');
Signal  reg_R:      STD_LOGIC_VECTOR(29 downto 0):=(others=>'0');
Signal  pulse3:     STD_LOGIC_VECTOR(29 downto 0):=(others=>'0');
Signal  pulse2:     STD_LOGIC_VECTOR(29 downto 0):=(others=>'0');
Signal  pulse1:     STD_LOGIC_VECTOR(29 downto 0):=(others=>'0');
signal  case_reg    :   STD_LOGIC_VECTOR(1 downto 0):="00";
signal  p1          :   STD_LOGIC:='0';
signal  p2          :   STD_LOGIC:='0';
signal  p3          :   STD_LOGIC:='0';
signal  preg        :   STD_LOGIC_VECTOR(5 downto 0):=(others=>'0');
signal  p           :   STD_LOGIC_VECTOR(1 downto 0):="00";
Signal  myreg       :   STD_LOGIC_VECTOR(29 downto 0):=(others=>'0');
begin
t1  <=  input;      --set pulses
PROCESS(clk,res)
variable count : integer range 0 to 29;
variable d1 : integer range 0 to 29;
variable d2 : integer range 0 to 29;
variable d3 : integer range 0 to 29;
BEGIN
--------------------------------------------------------------------------
--detecting pulses to apply MLSD
--------------------------------------------------------------------------
    IF (res='0')THEN
        case_reg    <=  "00";
        reg_R       <=  (others=>'0');
        reg_S       <=  (others=>'0');
        myreg       <=  (others=>'0');
        r_pulse <= '0';
    ELSIF(clk'EVENT AND clk='1') THEN
        r_pulse <=  input;
        s_pulse <= t1;
        reg_S   <=  (s_pulse        & reg_S(29 downto 1));
        reg_R   <=  (r_pulse        & reg_R(29 downto 1));
```

```
        myreg   <=  (s_pulse xor r_pulse) & myreg(29 downto 1);
        p<= p + (t1 xor input);
        IF((t1='1' or input='1') and p =0)then
            preg(5 downto 4)<=  t1 & input;
        elsif((t1='1'  or input='1' ) and p=1)then
            preg (3 downto 2)<= t1 & input;
        elsif((t1='1'  or input='1' ) and p=2)then
            preg(1 downto 0)<=  t1 & input;
            preg(5 downto 4)<=  t1 & input;

        case preg is
            when "101010" => reg_R <= reg_R xor pulse2; reg_S <=reg_S xor
pulse2; preg<="100000";
            when "010101" => reg_R <= reg_R xor pulse2; reg_S <=reg_S xor
pulse2; preg<="000001";
            when others => preg<= preg;
        end case;
        end if;
        if(p>0 and p<3)then
            d1  :=  d1+1;
        end if;
        if(p>1 and p<3)then
            d2  :=  d2+1;
        end if;
        if(p<2)then
            d1:=0;
            d2:=0;
            d3:=0;
            --p<="00";
        end if;
            d3  :=  d2- d1;
        IF(p=1)then
            pulse1<= '1' & pulse1(29 downto 1);
        elsif(p=2)then
            pulse1<= '0' & pulse1(29 downto 1);
            pulse2<= '1' & pulse2(29 downto 1);
        elsif(p=3)then
            pulse1<= '0' & pulse1(29 downto 1);
            pulse2<= '0' & pulse2(29 downto 1);
            pulse3<= '1' & pulse3(29 downto 1);
        else
            pulse1  <=  (others=>'0');
            pulse2  <=  (others=>'0');
            pulse3  <=  (others=>'0');
        end if;
        case_reg    <=  reg_S(0) & reg_R(0);
        case case_reg is
            when "00" => out_reg <= out_reg;
            when "01" => out_reg    <=  '0';
            when "10" => out_reg    <=  '1';
            when "11" => out_reg    <=  '1';
        end case;
    end if;
end process;
output  <=  out_reg;
END behaviour;
```

Appendix L

Power estimation VHDL code (Density of 1's being transmitted):

```
----------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.std_logic_arith.all;

Entity counter is
    PORT(
        clk :   IN STD_LOGIC;
        res :   IN STD_LOGIC;
        inp :   IN STD_LOGIC;
        zeros   :   OUT STD_LOGIC_VECTOR(39 downto 0);
        ones    :   OUT STD_LOGIC_VECTOR(39 downto 0)
        );
End counter;

Architecture bhv of counter is
Signal  count_ones  :STD_LOGIC_VECTOR(39 downto 0):=(others=>'0');
Signal  count_zeros :STD_LOGIC_VECTOR(39 downto 0):=(others=>'0'); --20bit
Begin
Process(res,clk)
Begin
    IF(res='0')then
    count_zeros <= (others=>'0');
    count_ones  <=  (others=>'0');
    ELSIF(clk'event and clk='1')then
    count_ones <= count_ones + inp;
    count_zeros<= count_zeros + not (inp);
    END IF;
End process;
zeros   <= count_zeros;
ones    <= count_ones;
End bhv;
```

Appendix M

Altera DE2 board specifications are given below:

Parameter	Value
FPGA chip	Cyclone II EP2C35F672C6
I/O pins	76 pins (2 expansion headers)
Memory	8 MB SDRAM, 512 KB SRAM, 4 MB Flash, Memory Card slot
Switches and buttons	18 toggle switches, 4 push buttons
Display and LEDs	16 x 2 LCD, 18 red and 9 green LEDs