Implementation & Performance Investigation of Dicode PPM over Dispersive Optical Channels

Romanos Athanasiou Charitopoulos

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

July 2009

Abstract

This work is concerned with the development and investigation of a Dicode PPM (DiPPM) system.

A DiPPM coder was developed to code any input PCM signal into DiPPM format. A further investigation took place on the DiPPM spectrum and associated output. Software simulation and mathematical analysis of this PPM code was considered and comparison with previous theoretical results presented. Results show that DiPPM is an advantageous PPM code for optic communication; DiPPM spectrum is not concentrated near to DC and it is possible to extract the DiPPM frame-rate component directly from the pulse stream.

A timing extraction circuit that recovers the clock from a DiPPM sequence and synchronises the slots within the frames, was constructed successfully. This enabled transmission through fibre optics and Free Space Optics (FSO). The technique used for the timing extraction circuit of the DiPPM scheme gives an advantage over many of the PPM formats.

An optical transmitter/receiver system was developed and the DiPPM scheme was investigated through optical channels. Results show that the DiPPM sequence transferred through the optic system was not changed and the clock had been recovered. A DiPPM decoder was constructed and the received DiPPM signal returned to its original PCM form without errors.

Both DiPPM coder and decoder were developed in VHDL and measurements were taken. The timing extraction was programmed in VHDL-AMS with the use of digital, analogue and mathematical equations.

DiPPM MLSD was also constructed in VHDL. Simulation results proved the theoretical expectations.

Acknowledgements

The author would like to thank lecturers and staff of Huddersfield University for their help throughout this project, in particular:

My academic supervisor and mentor Dr. M.J.N. Sibley for his advice, guidance and encouragement over the duration of the research project.

Thanks must also go to my academic supervisor Dr. P.J. Mather for his advice and assistance in the coding of my work into software program.

I also wish to acknowledge Mr. Jack Briggs and Mr. David Bray for their support on technical matters and Mr. D.C. Andrews and Mr. Anver Dadhiwala for their help with the software.

Finally, I would like to thank Dr. Ian Pitchford for his help with administrative matters.

Glossary of Symbols

Symbol	Definition
A	Amplitude
BdW	Bandwidth of the pre-amplifier
C	Capacitor.
C_c	Collector emitter capacitance
C_d	The emitter-base capacitance
C_f	Capacitance at the feedback resistor
Cn	The amplitude spectrum
$DiPPMprbs$	DiPPM PRBS sequence
$Error_{x,y}$	Number of PCM errors the event generates.
f	Frequency
ϕs	Phase shift
$h(t)$	Power Spectral Density (PSD) of a PCM sequence.
$h(t_{DiPPM})$	Power Spectral Density (PSD) of a DiPPM sequence.
K_0	The VCO conversation gain.
$\left\langle n_0^2 \right\rangle$	The mean square noise of the receiver.
$\left\langle n_0(t)^2 \right\rangle$	The mean square receiver output noise.
P_e	Equivalent PCM error probability (DiPPM-ISI).
P_{eb}	The average binary error probability
P_{efN}	The equivalent PCM error probability (DiPPM).

P_{er}	The PCM error probability for erasures.
$P_{erDiPPM}$	The PCM error probability for erasure.
P_{efR-Ts}	The equivalent PCM error probability.
P_f	The probability of a false alarm error.
$\left(P_{fISI1}\right)$	The following symbol of an S-slot.
$\left(P_{fISI2}\right)$	The previous symbol of an S-slot.
$P_{fN\to S/R}$	The probability of a false alarm in either the S or R slot of a blank frame
P_r	The probability of an erasure error at DiPPM without ISI
P_s	The probability of a pulse appearing in the preceding slot (wrong slot).
P_t	The probability of false alarm error in DiPPM.
$P_{x,y}$	Probability of a particular error event occurring.
R	Is the symbol for DiPPM RESET (01).
R_2	Resistor.
Re $set1$	Independent pulse train.
Re $set2$	Independent pulse train.
S	Is the symbol for DiPPM SET (10).
$Set1$	Independent pulse train.
$Set2$	Independent pulse train.
T	The clock period.
t_d	The slope of the received pulse at the threshold crossing instant.
τ_R	Time at which the autocorrelation function of the receive filter has

	become small. (DiPPM-ISI)
T_R	The time at which the autocorrelation function of the receiver filter has become small (DiPPM)
v_d	The threshold voltage.
V_f	The VCO control voltage from the Loop Filter.
Vin	The loop input voltage
$v_0(t_d)$	The ISI voltage at the decision time in the time slot under consideration.
Vout	The loop output voltage
v_{pk}	The peak signal voltage at the output of the receiver for either an S or R pulse.
v_{0ISI1}	The probabilities of (P_{fISI1}).
v_{0ISI2}	The probabilities of (P_{fISI2}).
$V_{oS}(x,y,t)$	Shape of the SET pulse.
$V_{oR}(x,y,t)$	Shape of the RESET pulse.
W	Window equation.
X(t)	DiPPM signal when PCM signal starts with a positive pulse (logic 1)
X(-t)	DiPPM signal when PCM signal starts with a negative pulse (logic 0)
ζ	The damping factor
ω	The ratio of input frequency.
ω_0	The VCO output frequency.
ω	Equal to 2πf.

Operator Notation

$\sum[.]$	Sum of terms.
$[.]^*$	Complex conjugate
$\lvert[.]\rvert$	Absolute value
$[.]*[.]$	Convolution.
$F[.]$	Fourier transform

Contents

Chapter 2

Chapter 3

Volume I of II

Hardware

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Volume II of II

'Software'

Chapter 8

Chapter 9

Chapter 10

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Optical communication is concerned with the transmission of information between two distant points using an optical carrier. The information can be transmitted in analogue or digital form. In analogue transmission, a source waveform has to be transmitted from the transmitter to the receiver with as little distortion as possible. However, in digital transmission the information is converted to binary symbols (bits). This information then modulates an optical carrier which is detected and decoded by the receiver into the original format. This digital information can be transmitted over the optical link on a bit-by-bit basis (binary encoding) or on a data word basis (block encoding).

A number of different techniques are used for the transmission of digital information. On-Off Keying (OOK) encoding and Manchester coding are two techniques which are associated with optical pulsed digital signalling. The Digital Subcarrier Intensity-Modulated System (DSIMS) technique modulates data bits onto an RF subcarrier and an intensity modulating subcarrier onto an optical signal. The heterodyne detection

technique, incorporated at the receiver is used to directly encode digital bits on the phase or frequency of the laser carrier. However, in all these systems only binary encoding was considered, in which bits are sent one at a time over the optical link. An alternative is the use of block encoding in which bits are transmitted in blocks.

Pulse Position Modulation (PPM) is a popular form of optical block encoding; *M* bits are coded by transmitting one pulse which is placed in one of 2^M adjacent time slots to represent the data block. The *M* slots compose a PPM frame and the location of the pulse in the frame determines the data word. The system is compatible with direct detection receivers as it uses pulsed optics. The pulse width of the *M* bit is equal to the slot width. Therefore, the laser needs only to produce pulses at the frame rate which operates at the pulse repetition frequency.

The PPM decoder must then determine which of the M slots occurring during a time frame contains the optical pulse. In PPM, every symbol of the received signal has to be fully synchronised with the clock signal. This is achieved by the timing extraction system. Once the clock has been extracted and synchronised with the start edges of the received PPM signal, the PPM signal can be decoded to the original form by the PPM decoder. However, noise in the detection process can cause errors.

Detection errors can occur on the signal during the time of optical transmission from coder to decoder due to receiver noise. Thus, an error correction system between the timing extraction and the decoder system would be desirable in order to avoid or minimise faults on the coding signal.

A number of PPM formats have been previously investigated; Multiple Pulse Position Modulation (MPPM) [1-12], Differential Pulse Position Modulation (DPPM) [13-16] and many related modulation coding schemes such as Pulse Interval Modulation (PIM) [1-3, 17-19], Digital Pulse Interval Modulation (DPIM) [20-21] and Dual Header Pulse Interval Modulation (DH-PIM) [21-26]. Analysis has shown that while each code has many advantages when compared to Pulse Code Modulation (PCM), this occurs at the expense of bandwidth [27-29]. This makes PPM systems attractive for use in glass fibre or directed line of sight networks where bandwidth is not at a premium. Unfortunately, glass optical fibre cable is expensive and unsuitable for everyday use such as networks. Conversely, Plastic Optical Fibre (POF) is inexpensive but has a low bandwidth making it unattractive for use with PPM schemes.

Previously, Digital PPM (DigPPM) had been proposed [27-29] as the most appropriate coding scheme for optic communication when compared to the other PPM formats. However, digital PPM format means that while a sensitivity advantage is achieved over standard PCM, it does so at the expense of a large bandwidth expansion factor. This results in a final data rate that can be prohibitively high. This led to the investigation of various other PPM schemes of which, Dicode PPM (DiPPM) [27], was proposed by Sibley as a more advantageous format than DigPPM. DiPPM can be effectively implemented as it uses two slots to transmit one bit of PCM. Unlike DigPPM, DiPPM achieves greater sensitivity than PCM at a slot rate of twice the original PCM data rate [27-30].

This thesis is concerned with the implementation of a DiPPM system and performance over fibre and free space optical channels with various level of dispersion. A complete

optical DiPPM system (coder, timing extraction and decoder) is presented and analysed. The coder output is presented in both time and frequency domain, and experimental results are compared to mathematical representations and simulations. VHDL codes of the DiPPM system are presented, including a DiPPM error corrector.

1.1 Aims and Objectives

Investigations undertaken and intermediate objectives fulfilled during PhD research:

- ***To implement a system and practically verify some of the theoretical models.***

  As no experimental construction of DiPPM coder/decoder had been presented at this time, a hardware DiPPM coder and decoder has to be implemented to further investigate the DiPPM format. Thus, it will be proved if DiPPM format can appear in real time (many PPM formats are remained in theoretical levels). The initial objective of the DiPPM coder/decoder is for it to be constructed using low cost components, giving an advantage over the majority of other PPM schemes.

- ***To confirm theoretical predictions with measurements.***

  Output of the DiPPM coder and decoder have to be compared with theory [1-4]. By carrying out this comparison, it will be proved whether the DiPPM coder produces correct DiPPM sequences and the DiPPM decoder can decode DiPPM sequences into PCM. Measurements of DiPPM Power Spectral Density (PSD) have to be taken and comparison with previous DiPPM PSD theoretical results

must be analysed. Should PSD outcomes not agree with those of previous publications, further investigation must be carried out on DiPPM PSD and on differences that might appear.

- ***To confirm theoretical predictions with measurements (fibre optics).***

  The main focus of this thesis is to investigate the performance of DiPPM through optics. Thus, an optical system such as an optical transmitter/receiver, has to be constructed and the performance of DiPPM waveforms through Plastic Optical Fibre (POF) or free-space, determined. Since the DiPPM decoder will need a clock sequence to decode the DiPPM sequence into PCM format, a clock has to be recovered. Thus, the construction of a timing extraction system for DiPPM is required. Outputs of the DiPPM timing extraction system must be examined with concern for synchronisation (DiPPM, clock) and the PSD of the optical received DiPPM.

- ***Design and build a Dicode PPM system using FPGA.***

  It is expected that the discrete component based hardware for the DiPPM coder/decoder and timing extraction will be affected by external interference due to the high frequencies (240 MHz) used. Thus, the DiPPM coder/decoder and timing extraction will be synthesised onto an FPGA which will give reduced levels of delays, distortions, external and internal interferences, enabling the code to be fully evaluated.

- ***To investigate additional coding techniques such as MLSD (Maximum Likelihood Sequence Detection).***

The performance of the DiPPM scheme is to be investigated incorporating error correction. A Maximum Likelihood Sequence Detection (MLSD) will be developed for DiPPM, to correct or minimise errors that occur from a long distance optical communication or from faulty reception by the optical receiver DiPPM sequence. This MLSD will be programmed in VHDL, as it is expected that the circuit will be complex because of loops, long shift registers and parallel processes. It is considered that any MLSD construction with logic components will be adversely affected by asynchronisation. These may appear because of external interferences and from internal delays.

1.2 Document Structure

An introduction to the PPM format and possible errors is contained in the Chapter 1. It also contains a brief review of previous PPM formats and of the format that this thesis presents.

Chapter 2 presents the historical progress of PPM through time and optical systems. Mention is made of components, such as Timing Extraction and Error Corrector, which are essential in an optical PPM scheme. Finally, a brief description of the most frequently used PPM formats is presented.

Chapter 3 provides a background review of Dicode Pulse-Position Modulation (DiPPM) in both versions (DiPPM ISI-DiPPM). Outcomes of previous publications are presented.

Chapter 4 contains DiPPM coder and decoder circuit construction. The construction of DiPPM structures and output results are presented and discussed.

Chapter 5 confirms the results of chapter 4 through software simulation of the DiPPM coder and mathematical representations of DiPPM sequences. This chapter introduces the use of a window equation so that the simulated results match the experimental DiPPM system.

The Timing Extraction circuit of the DiPPM scheme and successful implementation are presented in Chapter 6. The optical system (transmitter/receiver) used for optical communication of the DiPPM system and measurements taken appear in Chapter 7.

The implementation of the DiPPM coder and decoder using FPGA is presented in Chapter 8; including DiPPM timing extraction.

Chapter 9 presents The Maximum Likelihood Sequence Detector (MLSD) error corrector coding technique of DiPPM format.

Chapter 10 concludes the work presented in this thesis, highlighting the original contribution of DiPPM and suggesting further work to be done on the DiPPM scheme.

1.3 Original work presented by the Author

The work presented in this thesis is original and has led to several publications. The main areas of original contribution are:

- Hardware implementation of the DiPPM coder and decoder, developed by the author, shows that the DiPPM scheme can be achieved for real time operation. Outcomes agree with theory [27-30].

- Software simulation of the Dicode PPM has been programmed. PCM deterministic and pseudo-random binary sequence inputs were coded successfully into DiPPM format. Outcomes such as spectrums and sequences are shown. Windowing equations are given and used in the simulation. [31-32]

- Original practical measurements of the power spectral densities (deterministic-pseudo-random binary sequence) are presented. These indicate that DiPPM is a suitable PPM coding format for optic wireless, fibre and free space communication. Results show that the DiPPM spectrum is not concentrated near DC frequencies [32].

- Mathematical formulas of the DiPPM signal were calculated and are presented for both deterministic and Pseudo-Random Binary Sequence (PRBS) signals. A window function was used to produce a pulse width exactly the same as the one present in the experiments. The Power Spectrum Densities (PSD) of DiPPM has been given in the DiPPM equations. Theoretical, computed and experimental results agree, showing the accuracy of the method and why previous published results were not as accurate. [31-32].

- A timing extraction circuit for DiPPM which recovers and synchronises the clock from the DiPPM signal (DiPPM-clock) is presented. The DiPPM spectrums, magnitudes and phase spectrums of optical DiPPM system are also presented.[33]

- VHDL codes have been developed for DiPPM coder and decoder.

- DiPPM Timing extraction has been coded on VHDL-AMS.

- An error correcting circuit (MLSD) was programmed on VHDL software. Results of the complete optic DiPPM system (Coder, Optical System, MLSD, Decoder) are presented.

Chapter 2

LITERATURE REVIEW

Many authors have discussed the performance of analogue PPM systems since Shannon's [34] paper, '*Mathematical Theory of Communication'*. This was followed one year later, in 1949, by a book co-authored with Weaver [35], describing a theoretical fundamental limit on channel efficiency and the modulation formats that could approach this limit.. Golay published a paper [36] the same year; a theoretical model based on a multi-channel and threshold detection system as the first practical way of using digital PPM to attempt to achieve the Shannon limit. This digital PPM system was later named as QPPM and Golay is regarded as the founder of the QPPM systems.

In 1961, Jacobs [37] supported Golay's opinion that QPPM systems were suitable for space communication. Viterbi, in 1962, presented a correlation detection model and the associated word error probability formula [38]. But practical implementation of the detection method proved too complex. Gruenberg [39] worked on Jacobs' analysis and through comparison of the efficiency of QPPM with, AM, Fm-Fm, FSK systems,

changed Golay's statement that QPPM word error probability can approach the theoretical limit of Shannon with as small error probability as desired. Unfortunately, after analysis was complete, Chang [40] found that Gruenberg's results were not accurate. Later work showed that the word error probability of QPPM systems could not compare with the bit error probability of other modulation formats.

In 1973, Hubbard [41] presented the use of analogue PPM. This work analysed the effect of timing error and false alarm due to noise on the leading edge of the received pulse. Experiments on analogue PPM optical fibre were completed in 1975 by Holden [42]. Three years later, an investigation took place by Muoi and Hullett [43] on an analogue PPM system, for the minimisation of timing error due to noise on the pulse. A simple sub-optimum receiver was proposed, which took into account inter-pulse interference whilst suffering only a small reduction in signal when compared to the optimum design. In 1978, a publication by Luciano and Pirani [44], presented an analysis of system behaviour in terms of coding efficiency, timing information and PSD of a combined Pulse Amplitude Modulation (PAM) and optical communication system. The main achievements of this system are an abundant timing frequency component, reduced low frequency component and longest interval without a signal level change being three times the symbol period. Compared with previous signalling formats, this system provides greater advantage for timing purposes.

Digital PPM had not been fully analysed at that time, except in the area of deep space communication [45-47]. Lee and Schroeder [48] determined the laser power required to achieve a given bit error rate without any optimisation of the receiver, with the use of optical digital PPM. The transmission of digital PPM in a system that contains repeaters

was investigated by Gol'dsteyn and Frezinskiy [49] in the same year. Dolinar [50], with the use of *Conditional Nulling Receivers*, presented a receiver structure of digital PPM detection in 1982. This receiver has the ability to determine, whether to coherently combine a pre-determined local oscillator field with the received optical field, prior to detection. A year later, Garrett considered the performance of optical fibre digital systems using PIN-FET optical receivers. In 1986, Pires and de Rocha [51] maintained that an APD-FET receiver offers higher sensitivity than a PIN-FET in a digital PPM transmission over low level optical fibre and also greater sensitivity to that of PCM. At the same time, Mecherle [52-53] analysed various coding and decoding conditions using optical digital PPM.

Numerous publications on the implementation and format of PPM have been presented. In order to provide the necessary background, so that the main focus of this thesis can be understood, a review of implementation and format is briefly presented.

2.1 PPM and Optical Channels

By the late 1970's many authors [34-44] were involved with the improvement of the digital PPM scheme. In the 1980's digital PPM had evolved and been analysed for operation over optical fibre channels

A year after Dolinar's publication [51], Garrett [54-55] considered the performance of optical fibre digital systems using direct-detection and coherent-detection PIN-FET optical receivers for Gaussian received pulse shape. It was shown that the PIN-FET PPM

receiver provides a sensitivity of 10 to 12 dB greater than that of PCM systems. This improvement gave an advantage to PPM format over optical fibre channel; for an attenuation of 0.2dB/km, it represents an increase of 50 to 60 km in regenerator spacing. Thus, for every 3 dB improvement in sensitivity, the number of customers being served in a multi-user environment can be doubled.

In 1988, Calvert [56-58], using a theoretical model based on a modified Garrett analysis, completed digital PPM optical fibre experiments for the first time. He coded 7 PCM bits of a data rate of 8 Mbit/s to digital PPM, using direct detection at a wavelength of 850 nm, transmitted over 2km of multi-mode fibre. His experiments showed that digital PPM increases receiver sensitivity by 4 dB over an equivalent PCM system. Discussion of digital PPM and the fundamental limits for optical fibre communications were published by Garrett and Calvert [59] in 1989.

In 1990, Cryan and Sibley [60-61], completed theoretical analyses for coherent and direct detection digital PPM transmitted over optical fibre channels. The analysis demonstrated that, at a highly dispersive fibre channel, the complex pre-detection filter, derived by Garrett, could be replaced by a matched filter alone. Calculations demonstrated that with the use of a matched filter, the system would give degradation in receiver sensitivity of only 1.3 dB. In 1991, Massarella and Sibley [62] confirmed that for highly dispersed pulses, the optimal pre-detection filter, suggested by Garrett, can be replaced by a simple sub-optimal filter with a bandwidth higher than that of the matched filter. This provides a 1 dB reduction in receiver sensitivity compared to the optimum receiver. Cryan and Sibley [63], through experimental results for a coherent heterodyne test ring using this sub-optimum filtering technique, have shown that using the same receiver and coding

conditions as that of publication [62], the receiver sensitivity could be improved by 16.6 dB while still operating 7.3 dB away from the theoretical shot noise limit.

2.2 Timing Extraction and Synchronisation

Timing extraction is an inevitable process for any digital transmission system. Bennett [65] was one of the first authors to address the statistical and timing description of timing extraction synchronisation. Many other publications have been mentioned on timing extraction in digital transmission systems [65-68] and on the structure of the synchroniser [69-70]. Gagliardi [71] presented a method that used pulse edge tracking to feed an error signal to an oscillator running at the slot frequency. The DC component was removed from the pulse train, integrated between ½ and 1.5 slot durations and average value determined as error signal. Six years after Gagliardi's publication in 1980, Gol'dsteyn and Frezinskiy [72] presented an algorithm that defines the position of a digital PPM pulse under non-ideal clock synchronisation. In 1983 Gagliardi investigated the time synchronisation problem that exists in PPM systems [73] and considered a method using a correlator to establish the error signal that corrects the oscillator [74-75]. The use of full slot width pulses cancel the spectral component and consists of squaring the photo detector current before feeding the signal to a phase lock loop (PLL) [76-77].

A successful timing recovery using an APD receiver was investigated by Davidson [78]. In 1986, Sibley [79] considered a practical measurement of the timing component within

digital time frame. Spectral properties of PRBS digital PPM data were published by Elmirghani [80-84] in 1992. An analytical solution was presented for a given coding level and modulation index and requirements for slot and frame synchronisation were considered. Elmirghani also developed an original technique for frame synchronisation that does not need the use of low modulation indexes in order to increase the timing component for high level coding systems [84].

In conclusion, it could be said that PPM synchronisation is similar to bit synchronisation in PCM systems. The main differences being, that in PPM format, there is a long run of zeros and this places more demand on the PPM synchroniser depending on the different PPM format structure. The basic methods for the frame phase synchronisation of PPM systems are:

- A popular method to extract the clock from a PPM format, particularly with satellite optical space PPM, is with the use of a maximum likelihood detector. The synchronisation sequences are inserted periodically into the data stream; at the receiver the frame phase is estimated. Hence, they are synchronised with the generated frame clock. Because these inserted sequences do not contribute towards the data, they represent a redundant power and therefore reduce the sensitivity of the system. The system operating bit rate is reduced as sequences must be added at the transmitter and then deleted from the receiver at a later point.

- An alternative method, which does not suffer from any limitations, has been used in space PPM to track pairs of back to back pulses. As optical fibre PPM is different of that of the satellite PPM structure, higher coding levels must be used

in order to achieve optimum sensitivity. Sun and Davidson [85] defined some natural acquisition sequences in optical fibre PPM.

- Phase Lock Loop (PLL) in timing extraction and synchronisation systems does not affect the sensitivity of PPM. The PLL generated clock is synchronised with the PPM signal. Hence, the PPM format does not lose its sensitivity and could therefore be used in most PPM formats.

- A final method uses redundancy introduced by a line code [86-87] which limits the maximum pulse deviation from the centre of the frame and estimates the frame phase. However, this method adds complexity at both transmitter and receiver as it requires coders and decoders which make implementation difficult.

2.3 PPM and Error Correction

In the 1980's, analysis of the digital PPM scheme was mainly concerned with implementation over optical fibre channels. Usually, the pre-amplifier and effect of fibre dispersion limited bandwidth. There are three types of detection error in PPM systems: erasure errors, false alarm errors and wrong-slot errors. False alarm appears when the amplitude noise is greater than the threshold level. False pulses can occur in an empty slot and the decoder will take them into account. Wrong-slot errors occur when the PPM pulse changes position to the next, or previous slot from its original (eg: a 7 bit PPM word '0001000' could be '0010000' or '0000100'). Wrong-slot error can also appear

when the PPM pulse remains in the correct slot but produces a false pulse (ex: '0001000' could be '0011000' or '0001100'). While an optimal filter can be derived for the estimation of the pulse position, thus reducing the errors from receiver noise, error correction systems such as maximum likelihood sequence detection (MLSD) and Reed-Solomon (RS) can be used.

Atkins et al [88], Hero et al [89] and Takahashi [90] analysed the use of error correcting codes with multiple PPM (MPPM) modulation. [91] In 1981, Garrett presented the use of coding techniques for an optical fibre PPM channel. Several subsequent publications have discussed RS and convolution coding for direct and coherent detection of digital PPM over fibre optical channels. They conclude that RS coding offers increased receiver sensitivity and that there exists an optimum code rate for a particular system.

In 1989 Sugiyama and Nous [92] analysed the error performance of MPPM over an ideal photon counting channel and introduced maximum likelihood detection (MLD). Park and Barry [93] in a comparison of the sensitivity and bandwidth efficiency of digital PPM, overlapping PPM and multiple PPM, showed that an 8 level digital PPM system requires 4.4dB less power than a (4/2) MPPM system when channel bandwidth is high and MLD have been used in both cases. Sibley [8] proposed the matched filter based slope detection/MLD (MF-MLD) technique to reduce the impact of ISI on MPPM system in 2004. Three years later, he [10] used maximum likelihood sequence detection (MLSD) to recover the original PCM word from the MPPM scheme. In 2008, Sibley and Nikolaidis [11] used MLSD to examine the effects of linear increment and decrement, Gray code and random mapping of data on the performance of the MPPM system.

2.4 Multiple PPM

In 1977, a form of Multiple PPM (MPPM) was considered by Lee and Schroeder [1]. This incorporated digital PPM with multiple pulses per time frame. In 1978, Gol'dsteyn and Frezinskiy [2] theoretically investigated this modulation format. The noise immunity of MPPM was considered by Yemin and Petrich [4] a year later. Yemin and Petrich completed their investigation by using operating signals under intersymbol noise conditions such as pulse broadening. They derived error probabilities for incorrect reception of the signal, assuming a Gaussian received pulse shape and threshold crossing detection. In 1984, Bar-David et al [5] published research on Overlapping PPM (OPPM). This allows multiple positions per pulse width as well as fractional modulation indices (number of pulse widths per frame). He concluded that OPPM offers a 20 percent advantage over conventional PPM in nats/photon.

In 1989, Sugiyama [6] proposed MPPM as a method to improve band-utilization efficiency of an optical PPM channel. He concluded that transmission bandwidth can be reduced by half with the use of MPPPM. The performance degradation of MPPM systems due to slot synchronisation error for an APD type receiver was given in 1991 by Majumder [7]. Majunder also determined expressions for receiver degradation with jitter variance.

In 2004, M.J.N. Sibley [8] presented performance analysis of a (12/2) MPPM scheme (fig. 2.2) coding 1 Gbit/s PCM data and the use of graded-index plastic optical fibre. Maximum likelihood detection (MLD) was used to recover original PCM data and the effects of pulse dispersion on the wrong-slot, false alarm and erasure detection errors

were examined in detail. Results show that this MPPM scheme requires 2754 photons per pulse (-35.51 dBm), as opposed to 1 x 10^4 photons per pulse (-28.15 dBm) for PCM when operating under wide bandwidth conditions. A method for reducing the effects of ISI and IFI at low bandwidths was also presented. Two years later, M.J.N. Sibley and R.A. Cryan [9] presented an analysis which significantly simplifies receiver design by employing raised cosine filtering to eliminate ISI. Slope detection with maximum likelihood sequence detection (MLSD) on MPPM to recover the original PPM word, was presented by M.J.N. Sibley and K. Nikolaidis [10]. In 2008, they also presented [11] the effects of linear increment, linear decrement, Gray code and random mapping of data on the performance of a $\binom{12}{Y}$ MPPM system; they concluded Gray code to be most effective as it minimises Hamming distance.

H. Park [12] proposed partial-response precoding, instead of MLSD, as a method of combining coding with equalization and compared performance against traditional equalization methods. He maintained that, as precoding at the transmitter reduces the ISI span to two band periods, the complexity of the receiver is reduced significantly. Partial-response precoding with PDFD provides a good balance of performance and complexity.

MPPM Format

In order to compare this PPM scheme (Multiple PPM) with that which this thesis presents, the MPPM scheme and its relationship with PCM is presented by plots and tables.

PCM	MPPM
000	00011
001	00101
010	01001
011	10001
100	00110
101	01010
110	10010
111	01100

Table 2A: *Multiple PPM coding formats.*

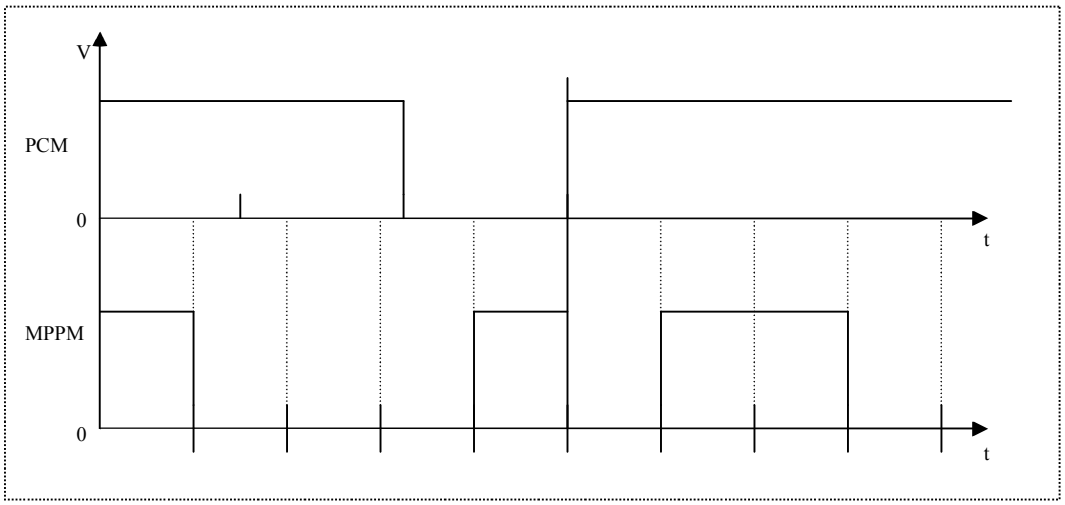


Fig. 2.1: *Conversion of 3 bits of PCM (top trace) to multiple PPM (bottom trace).*

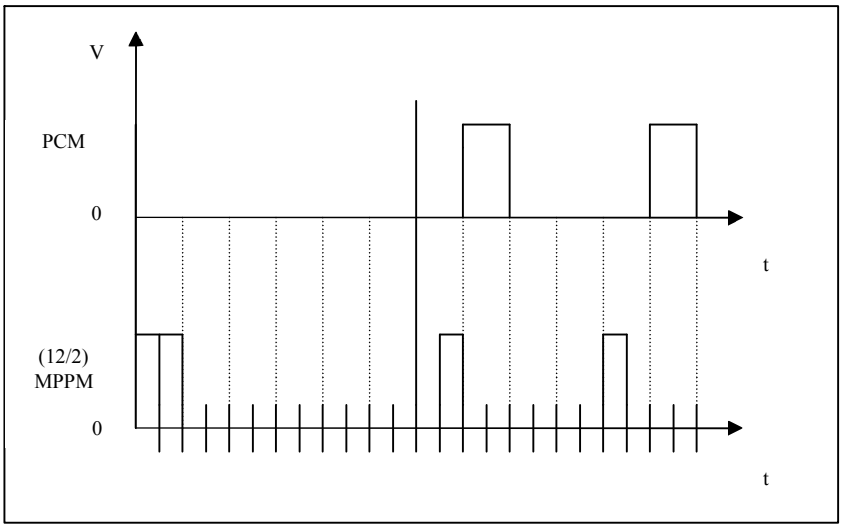


Fig. 2.2: *Conversion of 6 bits of PCM (top trace) to 12 bits multiple PPM (bottom trace).*

2.5 Differential PPM

In 1984, Shirokov [13] presented Differential PPM (DPPM) and evaluated reliability when transmitted over optical fibre channels. In 1988, Zwillinger [14] proposed Differential PPM (DPPM) as a way to increase the throughput for a band-limited and average-power-limited optical channel. Hard-decision decoding of DPPM on photon-counting channels was analysed by the same author. In the same year, Peile [15] analysed Differential PPM with error correcting codes combined with interleaving techniques. In 1999 Da-shan Shiu [16] also examined the use of DPPM for optical communication systems, using intensity modulation with direct detection in the presence of additive white Gaussian noise. He presented expressions for error probability and the power spectral density of the DPPM format. He concluded that for a given bandwidth, DPPM requires significantly less average power than pulse-position modulation (PPM); ISI penalties incurred by PPM and DPPM, exhibit very similar dependencies upon the channel delay spread. In 2008, Sethakaset and Guilliver [17] published a concatenation of

marker and Reed-Solomon codes in order to correct insertion-deletion errors in Differential pulse-position modulation. Code decoding algorithms with hard-decision and soft-decision detection were also presented.

DPPM Format

Table 2B and figure 2.3 present the relationship of the DPPM with PCM.

PCM	DPPM
000	1-00
001	1-00-0
010	1-00-00
011	1-00-000
100	1-00-0000
101	1-00-00000
110	1-00-000000
111	1-00-0000000

Table 2B: *Differential PPM coding format.*

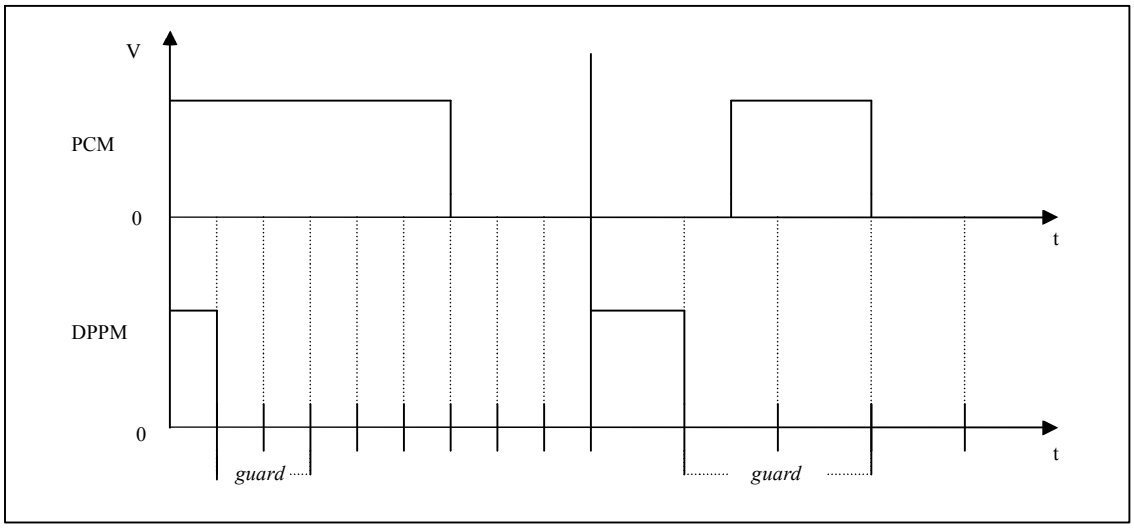


Fig. 2.3: *Conversion of 3 bits of PCM (top trace) to differential PPM (bottom trace).*

2.6 DH-PIM PIM, DPIM

In 1977, analytic and computer simulation results for a pulse-interval modulation (PIM) system, representing a discrete PPM system were published. [1]. The laser power required to achieve a given bit error rate was presented. Gol'dsteyn and Frezinskiy [2] theoretically investigated this modulation format a year later. A complete noise performance analysis for a PIM system was published in 1983 by Marougi and Sayhood [3]. Although PPM schemes offer an improvement in power efficiency, this is achieved at the expense of relatively poor bandwidth efficiency. Most PPM formats require symbol and slot synchronisation. Pulse interval modulation (PIM) requires no symbol synchronisation, while offering an improvement in bandwidth efficiency [18-19].

In 2000, Ghassemlooy [20] analysed digital pulse interval modulation (DPIM) in the presence of multipath propagation. Three years later, [21] he continued his investigation into DPPM and the effect of multipath dispersion; considering techniques that could be used to combat this. The effects of multipath propagation on link performance, based on a ceiling bounce model (CBM), have been widely used to predict ISI power penalties for diffuse digital pulse interval modulation DPIM and Dual Header pulse interval modulation (DH-PIM) [21-23]. Aldibbiat and Ghassemlooy [24-25] confirmed that DH-PIM offers higher transmission capacity while requiring less transmission bandwidth as compared with PPM and PIM. In 2001, they investigated [26] the performance of DH-PIM in distortion-free optical wireless channels in terms of error performance. They

presented complete derivations of the slot and packet error rates, optical and bandwidth requirements and confirmed calculated results through computing simulation.

PIM, DPIM & DH-PIM Formats

Figure 2.4 presents PPM, PIM, DH-PIM (a=2) and DH-PIM (a=1) sequences in comparison with a PCM sequence.

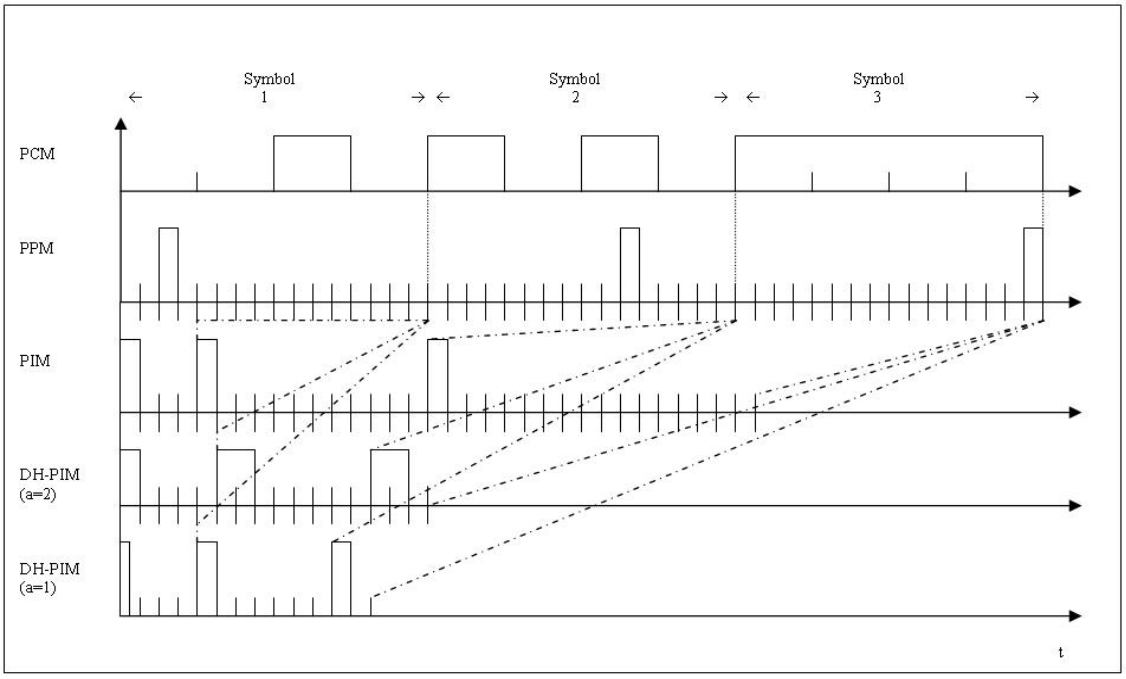


Fig. 2.4: *Conversion of 4 bits of PCM (top trace) to digital PPM (2nd top trace), to PIM (3rd top trace), to DH-PIM (4th and 5th bottom trace).*

2.7 Digital PPM

A brief background to Digital PPM has been provided. However, it will be helpful to add additional information regarding the spectrum of DigPPM in order that comparisons can be made with the PPM scheme.

In several publications [80-83], the spectrum of digital PPM has been presented and analysed. Elmirghani [83] presented a characterization of optical fibre digital PPM based on cyclostationary properties. He stated that optical fibre PPM, unlike satellite PPM, can exhibit discrete components at frame repetition rate. He also developed an original characterization for the cyclostationary PPM format, when impaired by timing jitter [94-95].

In 1998, Win [96] presented an analysis of digital PPM power spectral density (PSD). Win generated expressions for the PSD of digital pulse streams using wide sense cyclostationary (WSCS) sequences, in the presence of arbitrary timing jitter; methods based on stochastic theory were employed. Setting the period of WSCS sequence equal to one, results were obtained for a wide sense stationary (WSS) sequence. It was concluded that asymmetry of timing jitter does not affect the PSD. Mathematical calculations and simulation results are presented. The following is an equation of continuous PSD from Win's publication [93]:

$$S_m^c(f) = \frac{1}{T}|P(f)|^2 \sum_{t=-\infty}^{\infty}\left[\frac{1}{N}\sum_{n=1}^{N} K_{a,\varepsilon}(n;l,f,f)\right]e^{-2j\pi flT} \qquad (2.1)$$

DigPPM Format

Digital PPM and PCM relationship is shown in table 2C and figure 2.5.

PCM	Digital PPM
000	0000 0001
001	0000 0010
010	0000 0100
011	0000 1000
100	0001 0000
101	0010 0000
110	0100 0000
111	1000 0000

Table 2C: *Differential PPM coding format*

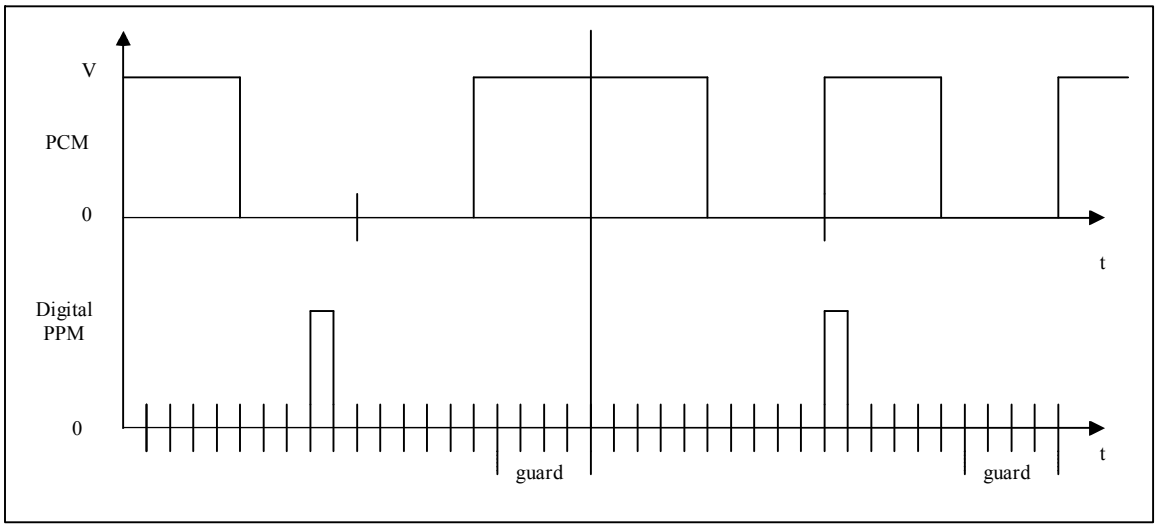


Fig. 2.5: *Conversion of 4 bits of PCM (top trace) to digital PPM (bottom trace).*

2.8 Dicode technique

The dicode technique is not a PPM format. It is usually used in magnetic recording channels where bandwidth is limited. Nevertheless, reference to this technique and the presentation of its format is indispensable for the better understanding of the Dicode PPM (DiPPM) scheme.

In 1971, Kobayashi [97] published several coding techniques, including the dicode technique, for the transmission and recording of digital data. Additionally, he presented error detection and correction of codes intended to combat random, or burst noise. Peter Kabal [98] states that because in real systems, data is not truly random, selected input sequences could conceivably suffer severe error propagation; precoding is therefore unhelpful. In 2001, Mats Oberg [99] presented an analytical method for estimating the average Euclidean distance spectrum for a serially concatenated, trellis-coded partial response channel. The technique was applied to the dicode channel, with and without precoding.

Few applications have been discussed which examine the dicode technique in optical fibre links. In 1999, Katsaros published outcomes of a comparison of the impact of FWM on binary, duobinary and dicode modulation in DWDM systems [100]. Comparison between duobinary, dicode and partial response class 4 modulation schemes for optical systems have been presented [101]. The robustness of duobinary, dicode and partial response class 4 modulation schemes through binary transmission in a two channel WDM system [102] has been also mentioned. Finally, duobinary and dicode comparison has

been completed from Costa [103-104]. Costa stated that when dispersion and Kerr included chirps are of the same size, an improvement in system performance is presented. Band limiting also has marginal benefits on signal propagation under SPM impact. When under strong SPM influence, dicode modulation formats, due to specific characteristics, demonstrate greater robustness.

Dicode Technique Format

In dicode technique, data transitions are coded; no signal is transmitted when PCM data is constant. When PCM data transition turns from logic zero to logic one, the dicode is coded as +V and while PCM transition change from logic 1 to logic zero, the dicode is coded as –V (table 2D – fig. 2.6).

PCM	Dicode Technique
00	0
01	+V
10	-V
11	0

Table 2D: *Dicode Technique coding format*

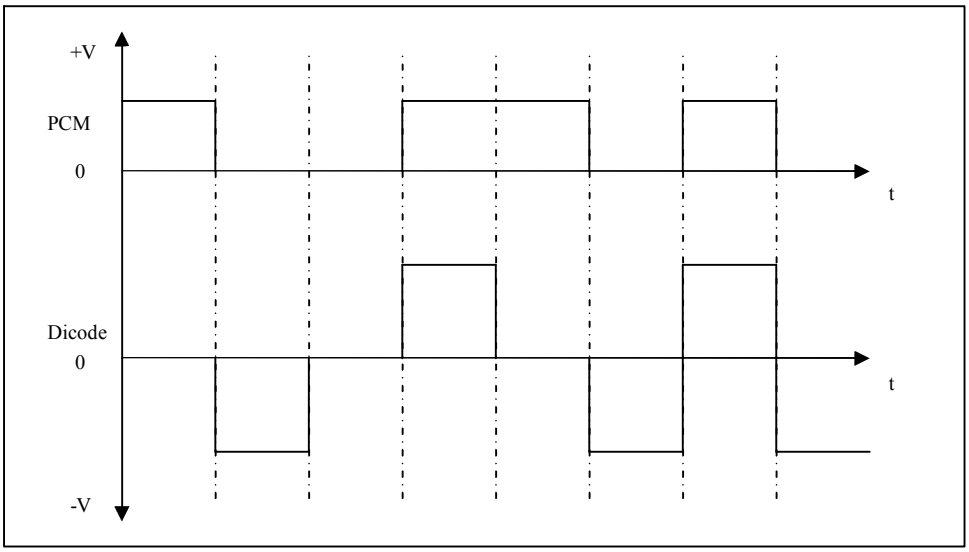


Fig. 2.6: *Conversion of PCM (top trace) to dicode technique (bottom trace).*

Chapter 3

Dicode PPM

As presented in chapter 2, various pulse position modulation (PPM) schemes have been proposed for use in optical communication links. Although these schemes operate with higher data rates than their PCM counterparts, they offer greater sensitivity than PCM; with digital PPM providing a 5 to 11 dB improvement [54-55, 57-59, 63, 105]. The optimum receive filter for digital PPM consists of a noise-whitened matched filter and a proportional derivative delay (PDD) network [54-55, 105]. As this PDD network is adjusted for individual links, implementation is complicated.

M.J.N. Sibley devised an original coding scheme; Dicode Pulse Position Modulation. Dicode PPM had been presented in two versions. The first version is the Dicode Pulse Position Modulation (DiPPM-ISI) [27], with two guard slots to be used to reduce the effects of inter-symbol interference (ISI). In the second version, the Dicode PPM (DiPPM) [30], only two data slots used to code data transitions in a PCM signal. In both versions (DiPPM-ISI and DiPPM) dicode PPM is a combination of the dicode technique

and digital PPM. Like digital PPM, Sibley's new scheme offers greater sensitivity than PCM. However, unlike digital PPM, sensitivity is achieved with only a fourfold (DiPPM-ISI) or twice (DiPPM) increase in speed. Although this Thesis (measurements, theories, simulations, mathematics, outcomes etc.) presents only DiPPM (without inter-symbol interference (ISI)), it will be helpful the original form (DiPPM-ISI) of the dicode PPM scheme to be analysed first.

3.1 Dicode PPM with ISI guards (DiPPM-ISI)

Using the dicode technique [see chapter 2, section 2.8], data transitions from logic zero to logic one are coded as +V and transitions from logic zero are coded as –V. No signal is transmitted when there is no changes in PCM signal (fig. 2.6 and 3.1). The positive pulse can be regarded as setting data to logic one (SET), whereas the negative pulse resets the data to logic zero (RESET).

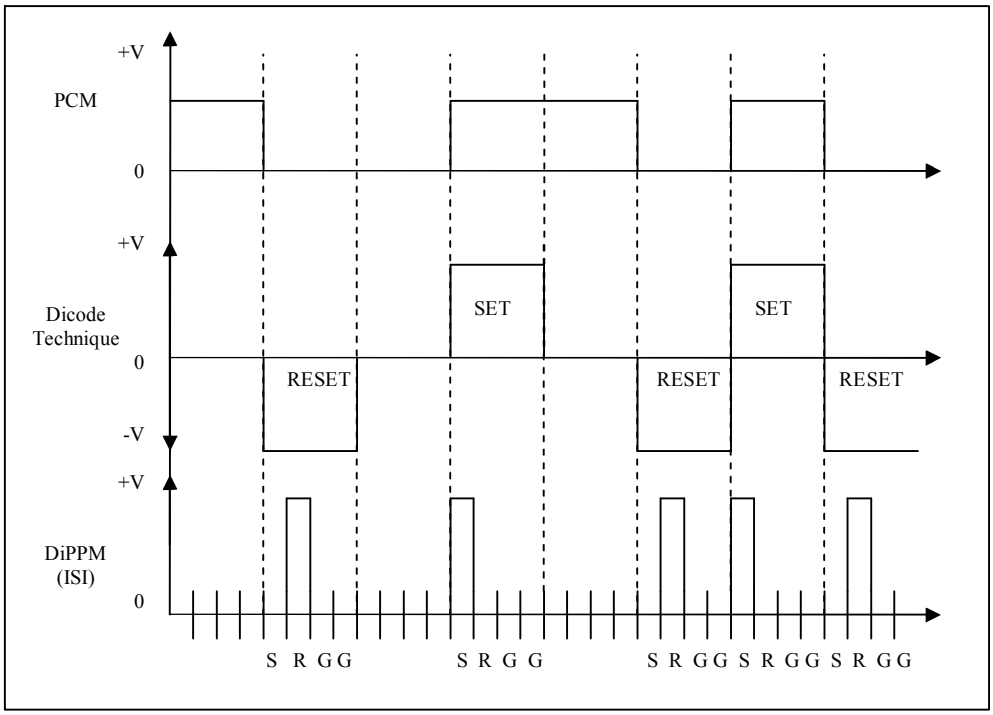


Fig. 3.1: *Conversion of PCM (top trace) to dicode technique (middle trace) and DiPPM-ISI (bottom trace).*

In DiPPM-ISI, the SET and RESET signals are converted into two pulse positions in a data frame. Thus, a PCM transition from zero to one produces a pulse in slot S and a one to zero transition generates a pulse in slot R (fig. 3.1). No pulse is transmitted when the PCM data is constant at either 1 or 0. Two guard slots are used to reduce the effects of inter-symbol interference (ISI). When there is minimal ISI, zero guard slots are used. As four slots are used to transmit one bit of PCM, and the line rate is four times that of the original PCM, speed is reduced when compared to digital PPM. Hence, DiPPM can be used in dense wavelength division multiplexing (DWDM) systems because of low bandwidth.

3.1.1 DiPPM-ISI Error Sources and Probabilities

DiPPM-ISI uses a four symbol alphabet (table 3A); a typical sequence would be S, xN, R with symbol probabilities of $1/2, (1/2)^x, 1/4$. The S signal has a probability of 1/2 because there are only two possible PCM sequences (00 or 01) after an R pulse has been transmitted. If the original PCM is line coded so that the run of like symbols is limited to n, the maximum DiPPM run would be R, nN, S. In this sequence, the S has a probability of one because it's presence is guaranteed at the end of a run of n lots of N symbols.

PCM	DiPPM	Symbol
00	no pulse	N
01	SET	S
10	RESET	R
11	no pulse	N

Table 3A: *DiPPM-ISI symbol alphabet.*

ISI - Included Error Probability.

In DiPPM-ISI, the shape of the S and R pulses will depend on transmitted pattern. The new pulse shapes, of a general DiPPM sequence S xN R yN S, are given by [29]:

$$V_{oS}(x,y,t) = V_0(t) + V_0(t+(3+4y)T_s) + V_0(t-(5+4x)T_s) \qquad (3.1)$$

$$V_{oR}(x,y,t) = V_0(t) + V_0(t+(5+4x)T_s) + V_0(t-(3+4y)T_s) \qquad (3.2)$$

Where only ISI between adjacent pulses has been considered. The general form of the equivalent PCM error probability, that considers a complete sequence for all x and y, is [29]:

$$P_e = \sum_{y}^{n-1}\left(\sum_{x}^{n-1}\left(\left(\frac{1}{2}\right)^{x+2}\left(\frac{1}{2}\right)^{y+2} P_{x,y}Error_{x,y}\right)+\left(\frac{1}{2}\right)^{n+1}\left(\frac{1}{2}\right)^{y+2} P_{n,y}Error_{n,y}\right)$$

$$+\sum_{x}^{n-1}\left(\left(\frac{1}{2}\right)^{x+2}\left(\frac{1}{2}\right)^{n+1} P_{x,n}Error_{x,n}\right)+\left(\frac{1}{2}\right)^{n+1}\left(\frac{1}{2}\right)^{n+1} P_{n,n}Error_{n,n} \qquad (3.3)$$

Where $P_{x,y}$ is the probability of a particular error event occurring and $Error_{x,y}$ is the number of PCM errors the event generates. The lower x and y limits depend on the type of error being considered, while the factor n is included to account for the line-coding of the PCM data [29].

Detection errors that occur in DiPPM systems in the presence of significant ISI, are the same as digital PPM: wrong-slot, erasure and false alarm.

Wrong-slot Errors

Wrong-slot errors occur when noise on the rising edge of a detected pulse causes the pulse to appear in adjacent time slots. In order to minimise this error, the pulse is detected in the centre of the time slot width T_s. Thus, errors are generated when the edge moves by $|T_s/2|$. The probability P_s of a pulse appearing in the preceding slot is given by [27, 29]:

$$P_s = 0.5erfc(\frac{Q_s}{\sqrt{2}}) \qquad (3.4)$$

where

$$Q_s = \frac{T_s}{2} \frac{slope(t_d)}{\sqrt{\langle n_0^2 \rangle}} \tag{3.5}$$

$\langle n_0^2 \rangle$ is the mean square noise of the receiver, and slope (t_d) is the slope of the received pulse at the threshold crossing instant, t_d.The presence of ISI means the slope of the pulse is different for S and R pulses and this must be accounted for by differentiating either (3.1) or (3.2) with respect to time.

A wrong-slot event can cause four possible errors in DiPPM-ISI. If the pulse appears in slot S, noise can cause the edge to appear in the preceding guard slot or in the following R slot. In the first case, no detection error occurs because the preceding slot is a guard and the decoder will not recognize the false threshold crossing. In addition, as the pulse is still present in the S slot, it will be detected correctly. In the second case, the S pulse appears in an R slot and this leads to detection errors. The probability of this happening is P_s as defined by equation (3.4) using the slope of the pulse defined by (3.1). This error causes the current PCM bit, and all following bits, to be in error until an R pulse is received. If the number of following N signals is x, the transmitted and received sequences would be as given in Table 3.B. Thus, (x+1) PCM errors are generated. The equivalent PCM error probability is given by (3.3) with x=0 and y=0 as the lower limits of the summations.

Transmitted	S	xN	R
Received	R	xN	R
Probability	1/4Ps	$(1/2)^x$	1/2

Table 3.B: *Transmitted and received sequences with a wrong-slot error.*

When the pulse is in the R slot, the slope must be obtained using the pulse defined by (3.2). This pulse could appear in the preceding S slot or the following guard slot, with (y+1) PCM errors resulting in both cases. The probability of error is given by (3.3) with x=0 and y=0 as the lower limits of the summations.

Erasure Errors

Erasure errors occur when noise is so great as to reduce peak signal voltage to below threshold level. The probability of this error is [27, 29]:

$$P_{er} = 0.5 erfc\left(\frac{Q_{er}}{\sqrt{2}}\right) \tag{3.6}$$

where

$$Q_{er} = \frac{v_{pk} - v_d}{\sqrt{\langle n_0^2 \rangle}} \tag{3.7}$$

v_{pk} is the peak signal voltage at the output of the receiver for either an S or R pulse and v_d is the threshold voltage. When ISI appears, the pulse is dispersed so that peak amplitude occurs after the time slot where the threshold crossing occurs. Hence, the peak amplitude of the signal within the time slot occurs at time $t_d + T_s/2$ - the edge of the slot. Erasure of an S or R pulse, results in (x+1) PCM errors for a SET pulse and (y+1) for a RESET pulse. The lower limits in (3.3) are x=0 and y=0. A SET or a RESET pulse

generates the same number of PCM errors. Hence, PCM error probability for erasure is [27]:

$$P_{erDiPPM} = 2\left(\sum_{x=0}^{n-1}\left(\frac{1}{2}\right)^{x+3} P_{er}(x+1) + \left(\frac{1}{2}\right)^{n+2} P_{er}(n+1)\right) \tag{3.8}$$

False Alarm Errors

Due to ISI, there can be a signal voltage in the slots surrounding the one containing a pulse. Noise on this voltage can lead to a threshold-crossing event when no pulse is presented; so called false alarm errors. Although there are many different types of ISI-induced false alarm present in DiPPM-ISI, only the most significant have been considered. The error involved is minimal. The probability of false alarm error in DiPPM-ISI is [27, 29]:

$$P_t = 0.5 erfc\left(\frac{Q_f}{\sqrt{2}}\right) \tag{3.9}$$

where

$$Q_f = \frac{v_d - v_0(t_d)}{\sqrt{\langle n_0^2 \rangle}} \tag{3.10}$$

where $v_0(t_d)$ being the ISI voltage at the decision time in the time slot under consideration. This voltage is dependent on whether an S or R pulse has been received.

The number of uncorrelated samples per time slot can be approximated to T_s/τ_R where τ_R is the time at which the autocorrelation function of the receive filter has reduced. The probability of a false alarm error then becomes [27, 29]:

$$P_f = \frac{T_s}{\tau_R} 0.5 erfc\left(\frac{Q_t}{\sqrt{2}}\right) \tag{3.11}$$

Simulations performed using all possible sources of ISI-induced errors, indicate that the major source of error is an R pulse spreading into the S slot immediately before it. Further sources of ISI-induced errors can be neglected as they involve pulse spreading over several time slots and the effect is therefore minimal [27, 29].

If a pulse in slot R generates a voltage in the S slot immediately before it, the decoder will register an S pulse before the arrival of the pulse leading to (y+1) PCM errors. The ISI voltage in the S slot comes from the preceding S pulse, the current R pulse and the following S pulse. Thus [29]

$$Q_{fr} = \frac{v_d - \left[v_0\left(t_d + 4xT_s\right) + v_0\left(t_d - 4yT_s\right)\right]}{\sqrt{\left\langle n_0^2 \right\rangle}} \tag{3.12}$$

The equivalent PCM error probability, P_{efR-Ts}, is obtained by substituting (3.12) into (3.11) to obtain P_f and using (3.3) with x=0, y=0 and (y+1) errors.

False alarms can occur without ISI in the interval between the R pulse and the S pulse. Thus [29]

$$Q_{fN\to S/R} = \frac{v_d}{\sqrt{\langle n_0^2 \rangle}} \tag{3.13}$$

The severity of the decoding error depends on where the false alarm error occurs in the string of N symbols. If the *k*th N symbol is in error, the equivalent PCM error probability is [29]

$$P_{efN\to S/R} = 2\sum_{y=1}^{n-1}\sum_{k=1}^{y}\left(\frac{1}{2}\right)^{y+3} P_{fN\to S/R}(y+1-k) + 2\sum_{k=1}^{n}\left(\frac{1}{2}\right)^{n+2} P_{fN\to S/R}(n+1-k) \tag{3.14}$$

where $P_{fN\to S/R}$ is the probability of a false alarm in either the S or R slot of a blank frame, obtained using (3.13) and (3.12). The factor of two appears because a false alarm in an S slot has the same effect as one in an R slot

3.1.2 DiPPM (ISI)-PINFET Receiver

In the 2003, Sibley [29] presented an analysis of the performance of a PINFET receiver used with DiPPM-ISI. Simulations were performed with a receiver of an equivalent input low frequency noise current spectral density of 1 x $10^{-26}\, A^2/Hz$ and a noise corner frequency of 1 x 10^9 rad/s. Sibley showed that the theoretical model he presented, offers

a sensitivity of -59.51 dBm when operating with a large fibre bandwidth and a PCM data rate of 155.52 Mbit/s.

The DiPPM-ISI receiver presented by Sibley (fig 3.2) has a PINFET receiver to detect the optical signal. The receiver is followed by a post-amplifier and a receive filter, prior to threshold crossing detection and decoding back to PCM.

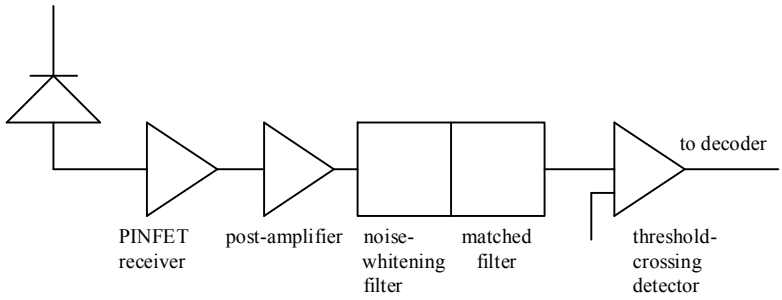


Fig. 3.2: *Block diagram of DiPPM-ISI receiver.*

In 2004, Sibley [29] analysed the performance of a DiPPM-ISI system using a third order Butterworth filter as the pre-detection filter. Inter-symbol interference effects were accounted for in the original analysis. Sibley showed that a receiver using a Butterworth filter requires less photon per pulse than one using a noise-whitened matched filter. The Butterworth receiver can also operate within lower channel bandwidths. For normalized bandwidths greater than three, the bandwidth of the receiver and the Butterworth filter are independent of the channel bandwidth. Thus, greatly simplifying the design of a link. With a low bandwidth link, the receiver and filter bandwidths are relatively insensitive to changes in noise corner frequency because minimisation of ISI is the criteria. With a high bandwidth link, ISI is low and so the receiver and filter bandwidths must be adjusted to minimise noise. Finally, Sibley states that the greatest predicted sensitivities, when using

a Butterworth filter and 1 Gbit/s PCM data, are -37.48 dBm for $f_n = 100$ and -32.24 dBm for $f_n = 1.2$ [28].

An analysis for optical preamplified DiPPM-ISI receiver was published in 2007 [109]. Andrew Phillips and Al-Suleimani analysed an optically preamplified DiPPM-ISI receiver. They derived new equations for DiPPM-ISI average power. With the use of these equations they concluded that digital PPM format outperforms DiPPM-ISI by between 3 to 5 dB. However, DiPPM-ISI remains superior in terms of performance when compared to PCM.

3.1.3 Dicode PPM-ISI through Optical Wireless Communication

While this thesis describes an investigation of DiPPM through fibre optic communication, it is useful to mention the outcomes of a publication about the use of DiPPM-ISI format on optical wireless communication. In 2005, Menon and Cryan [110] presented original results for a wireless system employing DiPPM-ISI and a PIN-BJT receiver. This system operated at a bit rate of 10 Mb/s and employed a receiver with noise current of 7.29 x 10^{-24} A^2/Hz and a bandwidth of 70 MHz. Menon and Cryan mathematically presented three error types (wrong slot, false alarm, erasure) of DiPPM-ISI using a preamplifier (PIN-BJT), a matched filter and a threshold detector. Results show that DiPPM-ISI shows an improved performance over a comparable PCM system. Additionally, the high sensitivity and low line rate expansion associated with the DiPPM-

ISI technique make it an attractive modulation format for indoor infrared wireless transmission.

3.2 Dicode PPM (DiPPM)

In 2006, Cryan and Sibley [30] presented DiPPM in a different form. They considered reducing the impact of intersymbol interference on dicode PPM, by proposing an alternative detection strategy to the slope detection technique normally employed in PCM systems. By moving from slope detection to central decision detection, they showed that the DiPPM receiver can be significantly simplified; with equivalent sensitivity performance in higher fibre bandwidths and considerably enhanced performance at lower bandwidths.

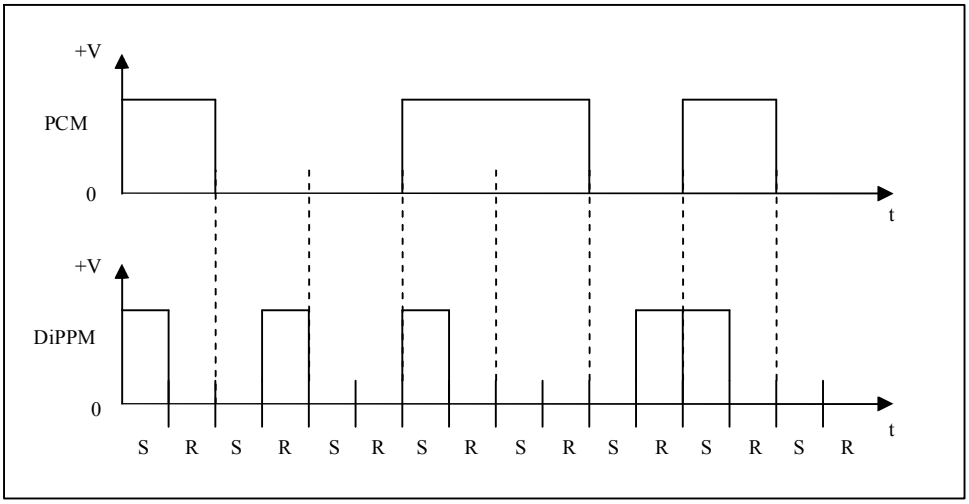


Fig. 3.3: *Conversion of PCM data (top trace) to dicode PPM (bottom trace).*

3.2.1 DiPPM Error Sources and Probabilities

Table 3.C shows the dicode PPM symbol alphabet. As the symbols are four, each symbol has a probability of ¼. However, the probability of a no pulse sequence (N) is ½ as it occurs at both 00 and 11 PCM sequence. The S symbol has the same probability as the no pulse (N), because there are only two possible PCM sequences (00 or 01) after an R pulse has been transmitted. A typical dicode sequence would be S, xN, R with probability of ½, $(1/2)^x$ and ¼ respectively [28].

PCM	Probability	DiPPM	Symbol
00	¼	no pulse	N
01	¼	SET	S
10	¼	RESET	R
11	¼	no pulse	N

Table 3.C: *Transmitted and received sequences with a wrong-slot error.*

In common with DiPPM-ISI, DiPPM is subject to three main error sources: Wrong-slot Errors, Erasure Errors and False Alarm Errors.

Wrong-slot Errors

Wrong-slot errors exist when noise affects the leading edge of the pulse, producing a threshold crossing in the time slot immediately preceding or following that containing the pulse. The probability of the wrong-slot error P_s of DiPPM is [30]:

$$P_s = erfc\left(\frac{Q_s}{\sqrt{2}}\right) \tag{3.15}$$

where

$$Q_s^2 = \left(\frac{t_s}{2}\right)^2 \frac{1}{\left\langle n_o(t)^2 \right\rangle}\left(\frac{dv_0(t)}{dt}\bigg|_{td}\right)^2 \tag{3.16}$$

while t_s is the DiPPM slot time. In DiPPM, a wrong slot error can lead to the same four errors as DiPPM-ISI. However, the number of decoding errors generated depends upon the position of the subsequent symbol. The equivalent PCM error probability due to wrong-slot errors is [30]:

$$P_{es} = 3\left[\sum_{x=0}^{n-1}\left(\frac{1}{2}\right)^{x+3} P_s(x+1) + \left(\frac{1}{2}\right)^{n+2} P_s(n+1)\right] \tag{3.17}$$

Erasure Errors

Erasure errors occur when, due to a negative noise spike, the amplitude of the pulse falls below the detection threshold. The probability of an erasure error P_r at DiPPM is [30]:

$$P_r = 0.5erfc\left(\frac{Q_r}{\sqrt{2}}\right) \tag{3.18}$$

where

$$Q_r^2 = \frac{\left(v_p - v_d\right)^2}{\left\langle n_o(t)^2 \right\rangle} \tag{3.19}$$

where $\left\langle n_0(t)^2 \right\rangle$ is the mean square receiver output noise, v_d is the receiver output at the threshold crossing time t_d and v_p is the peak receiver output which occurs at time t_p.

Erasure errors in DiPPM may result in different errors to the decoded PCM signal. Consider that an erasure error occurs on an S symbol; If the DiPPM code is RS, the decoded PCM outcome is expected to be 01. If the S is erased then the PCM will be 00, resulting in a single PCM decoding error. Alternatively, if the DiPPM sequence is SNNR, the decoded PCM outcome will be expected to be 1110. If S is erased the PCM will be presented as 0000 and three PCM decoding errors will appear.

Hence, the number of resulting errors depends upon the number of N-symbols, xN, and results in (x+1) PCM errors. In order to maintain timing content, the maximum run of N-symbols is limited to a run-length, n. The probability of a SET occurring is $1/4 = (1/2)^2$. The probability of the next and subsequent dicode symbols being RESET is a ½. Thus, the probability of a RESET occurring in slot x is $(1/2)^{x+1}$. Summing over x gives [30]:

$$\sum_{x=0}^{n-1} \left(\frac{1}{2}\right)^{x+3} P_r(x+1) \tag{3.20}$$

Due to run-length constraint, a run of n like symbols would lead to a forced transition in the encoding process with a probability of unity. Thus, in the case x=n [30]:

$$\left(\frac{1}{2}\right)^{n+2} P_r(n+1) \tag{3.21}$$

If the erasure of a SET or a RESET pulse generates the same number of PCM errors, then the PCM error probability for erasures is [30]:

$$P_{er} = 2\left[\sum_{x=0}^{n-1}\left(\frac{1}{2}\right)^{x+3} P_r(x+1) + \left(\frac{1}{2}\right)^{n+2} P_r(n+1)\right] \tag{3.22}$$

False-alarm Errors

False alarm errors occur when noise causes a threshold violation and a pulse appears at the position where no pulse was expected. The probability of a DiPPM false alarm error P_f is [30]:

$$P_f = \frac{t_s}{T_R} 0.5 erfc\left(\frac{Q_f}{\sqrt{2}}\right) \tag{3.23}$$

where

$$Q_f^2 = \frac{(v_d - v_{0ISI})^2}{\langle n_0(t)^2 \rangle} \tag{3.24}$$

t_s / T_R is the number of uncorrelated samples per time slot. T_R is the time at which the autocorrelation function of the receiver filter has reduced.

The probability of a false-alarm error is dependent upon intersymbol interference, which is reflected by subtracting V_{oISI} from v_d. In the case that the pulse appears in slot R, the pulse can spread into the S-slot of the following symbol (P_{fISI1}), or into the previous S-slot of the same symbol (P_{fISI2}). Thus, the PCM error probability is [28]:

$$P_{efR} = \left[\sum_{x=0}^{n-1}\left(\frac{1}{2}\right)^{x+3} P_{fISI1}\, x + \left(\frac{1}{2}\right)^{n+2} P_{fISI1}\, n\right] + \left[\sum_{x=0}^{n-1}\left(\frac{1}{2}\right)^{x+3} P_{fISI2}(x+1) + \left(\frac{1}{2}\right)^{n+2} P_{fISI2}(n+1)\right] \tag{3.25}$$

where the probabilities of (P_{fISI1}) and (P_{fISI2}) are determined from (3.23) by using:

$$v_{0ISI1} = v_0(t_d + t_s) \tag{3.26a}$$

and

$$v_{0ISI2} = v_0(t_d - t_s) \tag{3.26b}$$

When the pulse appears in slot S, this can spread into the R-slot of the preceding symbol. No effect is produced as the decoder is already in the RESET state. In the event of spread into the following R-slot of the same symbol, there will still be no effect as the decoder detects the S pulse associated with the symbol. False-alarms may occur between S and R pulses where there is no ISI. Hence, the number of PCM decoding errors will depend on the symbol position, *k,* where the false-alarm error appears within the run of N-symbols. Thus, the equivalent PCM error probability [28]:

$$P_{efN} = \sum_{x=1}^{n-1}\left(\frac{1}{2}\right)^{x+3}\sum_{k=1}^{x}(x+1-k)P_{fN} + \left(\frac{1}{2}\right)^{n+2}\sum_{k=1}^{n}(n+1-k)P_{fN}$$

$$+\sum_{x=2}^{n-1}\left(\frac{1}{2}\right)^{x+3}\sum_{k=2}^{x}(x+1-k)P_{fN} + \left(\frac{1}{2}\right)^{n+2}\sum_{k=2}^{n}(n+1-k)P_{fN} \qquad (3.27)$$

Hence, the DiPPM errors are:

$$P_{eb} = P_{er} + P_{es} + P_{efR} + P_{efN}$$

where P_{eb} is the average binary error probability [28].

3.2.2 DiPPM-PINFET Receiver

R.A. Cryan and M.J.N. Sibley [30], mathematically presented the DiPPM receiver configurations, the DiPPM matched filter and the DiPPM Butterworth filter. By using a simple first order preamplifier in cascade with a 3rd order Butterworth with bandwidths set at 0.6 times the DiPPM slot rate, it was shown that the sensitivity results were within 0.2 dB of those using ideal raised cosine spectrum filtering while varying the normalized fibre bandwidth from 1 to 100.

Cryan and Sibley determined that by moving from slope detection to central decision detection, the DiPPM receiver can be significantly simplified with equivalent sensitivity

performance at higher fibre bandwidths and considerably enhanced performance at lower bandwidths.

3.3 Dicode PPM-ISI Spectral Characterisation

Two years before Cryan and Sibley [30] presented DiPPM format with minimised intersymbol interference, Cryan published a brief analysis concerned with the spectrum of DiPPM-ISI [108]. He provided mathematical representations, following the method of Win [96], of discrete and continuous (fig. 3.4) power spectral density (PSD) of DiPPM-ISI using Fast Fourier Transform (FFT). Mathematical outcome graphs were compared with DiPPM-ISI PSD simulations and found to correlate. Cryan determined that, unlike PCM, the DiPPM-ISI spectrum is not concentrated at near DC frequencies. Hence, low frequency external interference will not affect the signal. It is therefore possible to extract the DiPPM-ISI frame rate component directly from the pulse stream. Due to the low-frequency content of the continuous spectrum, DiPPM-ISI may be suitable for optical wireless communication.

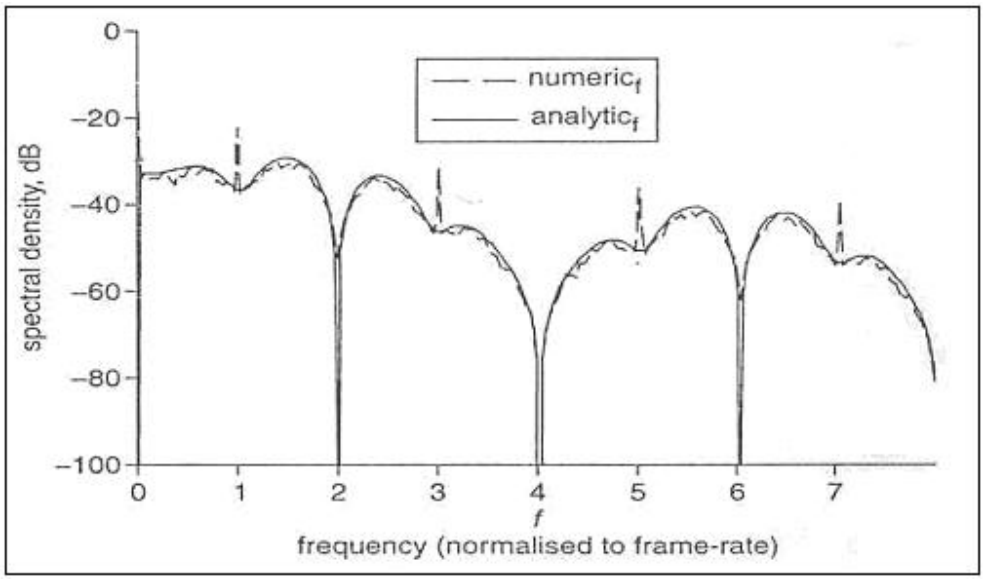


Fig. 3.4: *DiPPM-ISI Spectral by [108].*

The relevant equation predicting the spectrum is given below as equation (3.28).

$$S_M^c(f) = A^2 T_s \left| \frac{\sin(\pi f T_s)^2}{\pi f T_s} \right| \times \left\{ \begin{array}{l} \frac{3}{32} - \frac{1}{32}\cos(2\pi f T_s) + \frac{1}{32}\cos(6\pi f T_s) \\ -\frac{1}{16}\cos(8\pi f T_s) + \frac{1}{32}\cos(10\pi f T_s) \end{array} \right\} \tag{3.28}$$

In 2006, a brief publication [104] mentioned a disagreement regarding the accuracy of the continuous PSD of equation (3.28) [103]. Later, hardware and simulation outcomes of DiPPM of [109] did not agree with that of fig.3.4. A year later [31] it was concluded that the cause of this disagreement might be the different widths of the pulses that had been used in both simulations and that a window equation would have to be used for the accuracy of the results. In this thesis, a chapter will be spent examining this disagreement.

Volume I of II

'Hardware'

Contents of Volume I

Chapter 4

Dicode PPM Hardware

Due to no previous version of a DiPPM coder and decoder construction, these two structures of the DiPPM system were investigated. The bit rate of data was coded in the range 100-120 Mbit/s. Thus, logic components from the F100K ECL family were used. F100K ECL are high speed logic components with a power supply of 0 volts to -5 volts. In the experimental coder and decoder, the power supply voltage to the ECL components was -2.5 volts to 2 volts, in order that the correct termination for the output of the gates was 50Ω to ground. The input sequences of PCM and clock, from an ECL signal generator, were shifted to the positive levels of 0 volts to 2 volts.

4.1 DiPPM Coder

4.1.1 Implementation of DiPPM Coder

The DiPPM coder is the element of a DiPPM system which formats PCM sequences into those which contain DiPPM alphabet symbols (see Table 3.C). The logic components used to complete the DiPPM coder are two Flip-Flops, five NOR gates (figure 4.1).

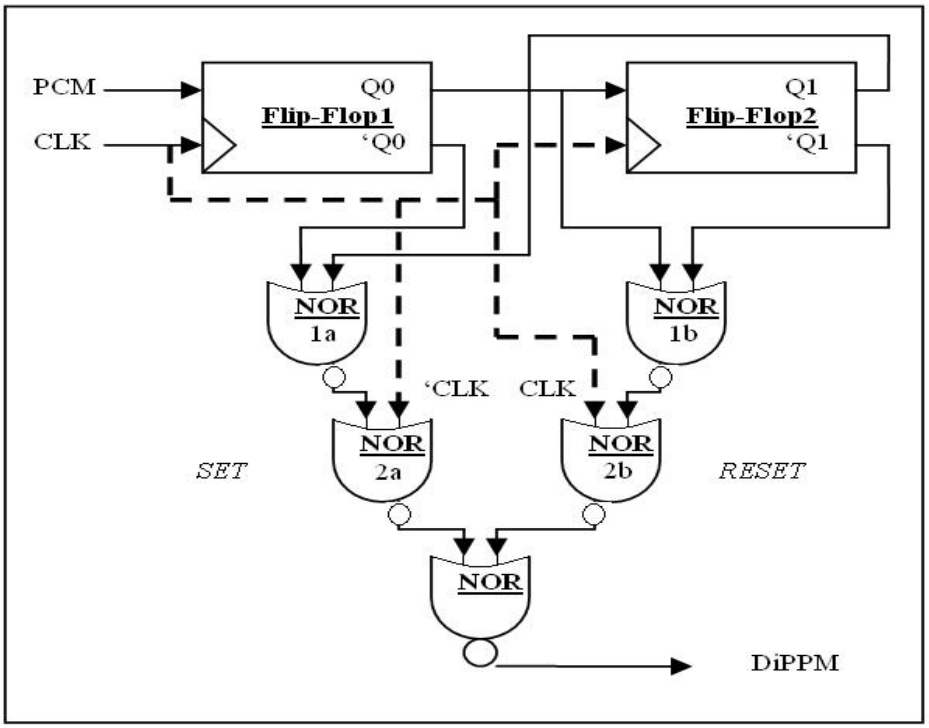


Fig. 4.1: *DiPPM coder's circuit.*

Flip-Flop 1 and Flip-Flop 2 form a two bit store, the outputs of which are used to produce the *SET* and *RESET* sequences.'Q0 and Q1 are passed through a NOR gate (1a) to produce the *SET* sequence and the pair Q0 and 'Q1 produce the *RESET* sequence through the NOR gate (1b). The use of the CLK and NOT CLK signal, is necessary to retime the

DiPPM *SET* and *RESET* sequences. The DiPPM *SET* and *RESET* sequences are produced by the two NOR gates (2a and 2b). An NOR gate combines these two sequences and produces the final DiPPM sequence.

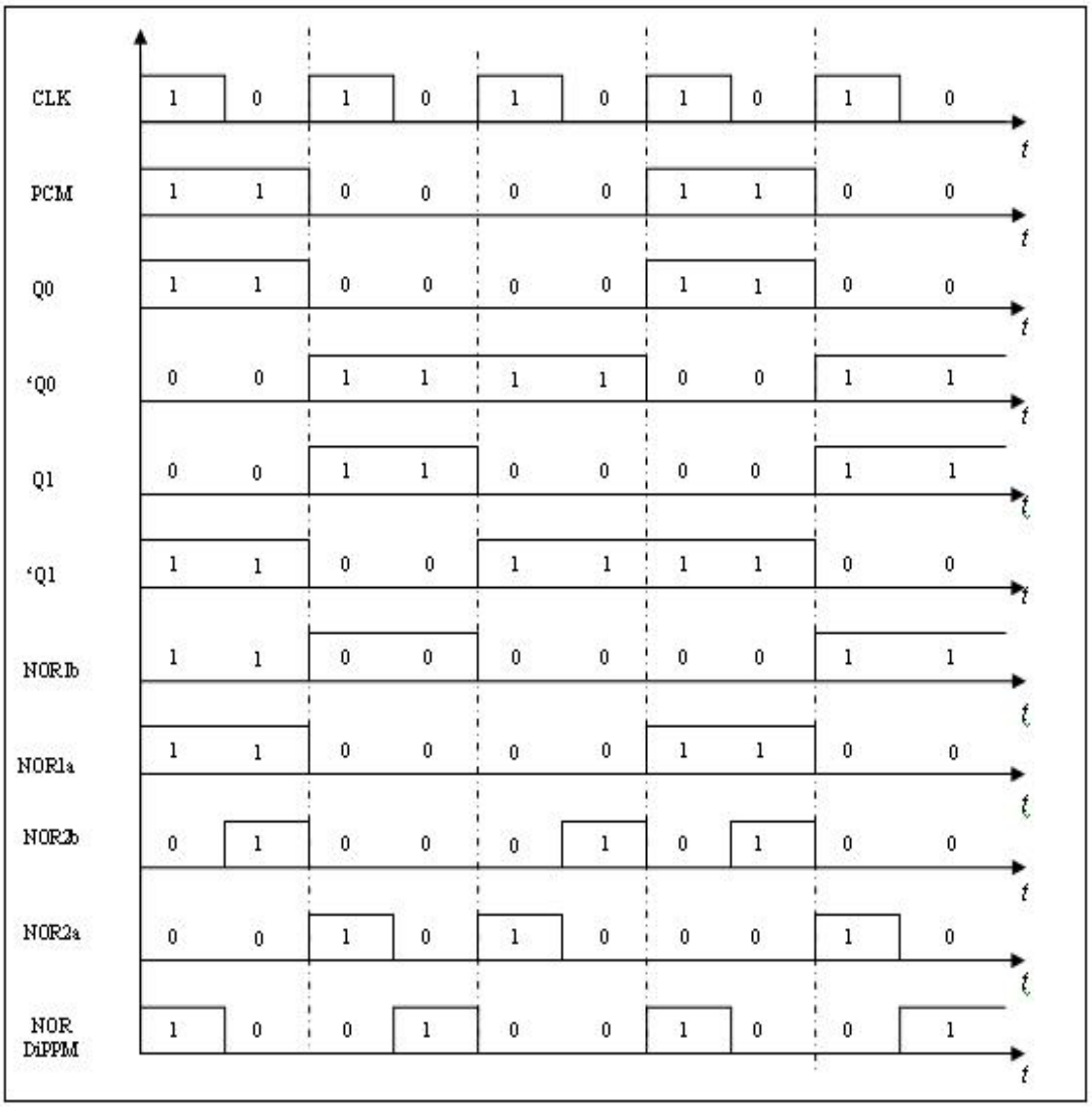


Fig. 4.2: *DiPPM coder's waveforms.*

4.1.2 Measurements from the DiPPM Coder

Experimental measurements of the DiPPM coder (Appendix3-A) were taken with a PCM data rate of 120 Mbit/s. Running the system with a deterministic PCM input (1, 0, 1, 0, 1…), where input signal and input clock were synchronised at all times (see 4.1), the output DiPPM sequence is shown at figure 4.3 in comparison with clock sequence.

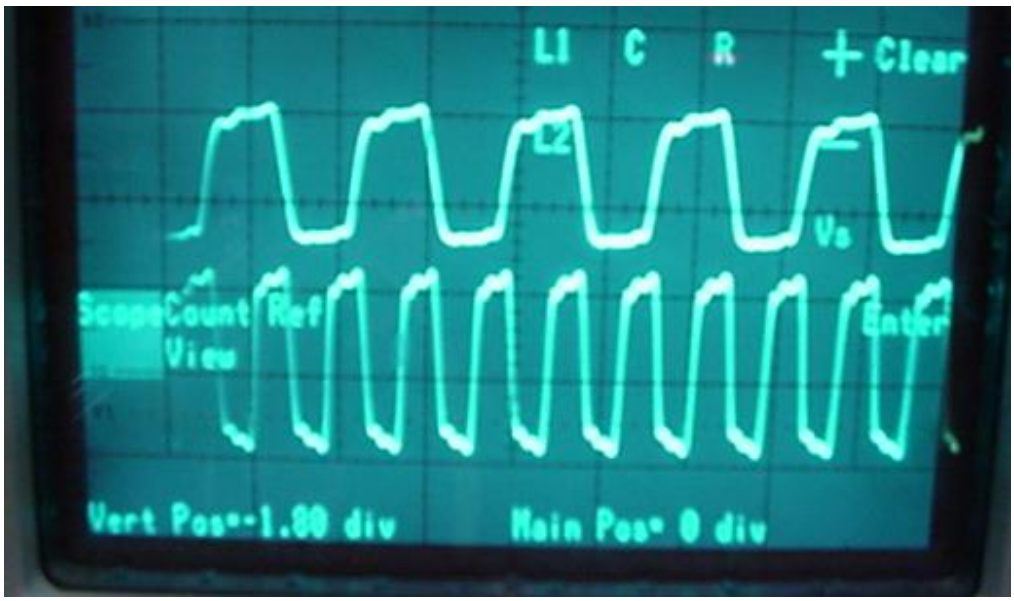


Fig. 4.3: *DiPPM coder's deterministic outcome (top trace), clock sequence (bottom trace).*

The DiPPM deterministic sequence was accurately synchronised with the clock. Figure 4.3 clearly shows that the DiPPM SET (10) a bit (1) at the beginning of a period clock and the DiPPM RESET (01) generates a bit (1) at the half clock period, as expected (Table 3.C).

The Power Spectral Density (PSD) of the deterministic DiPPM signal is shown in figure 4.4 [31-32]:

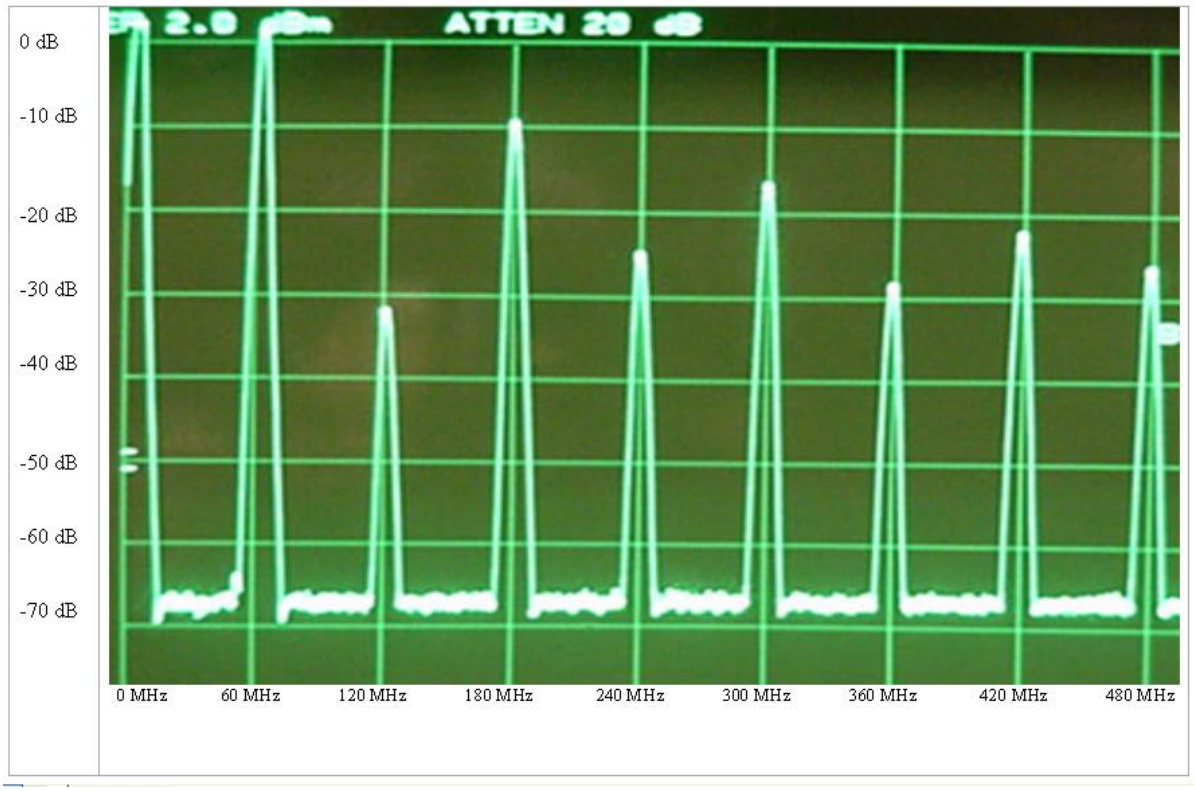


Fig. 4.4: *DiPPM PSD of deterministic sequence (hardware).*

It can be seen from Fig. 4.4, that there is a fundamental signal at the 60 MHz frequency and associated odd and even harmonics frequencies. As can be seen from figure 4.3, the period frequency for the DiPPM deterministic sequence is 60 MHz (‘10’+’01’=’0110’); it therefore does not have a perfect symmetrical shape. Due to practical limitations and the high frequency at which the experiment took place, the mark to space ratio for the SET and RESET signals was uneven. Hence, with the DiPPM SET pulse shown in figure 4.5, the time period for the positive logic 1, (tb), is not equal to half the slot period T. This has the effect of moving the nulls in the spectrum to a slightly higher frequency and, as a result, must be accounted for when comparing theoretical and experimental results [106].

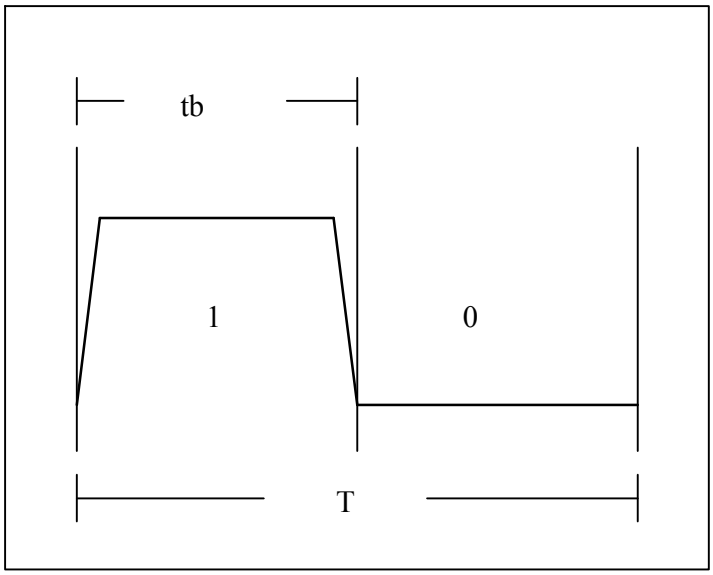


Fig. 4.5: *Uneven mark space on SET DiPPM pulse.*

Taking the same measurements as previously stated, but with Pseudo-random Binary Sequence (PRBS) PCM input signal to the DiPPM coder, the outcome sequence of the DiPPM PRBS in comparison with the clock is:

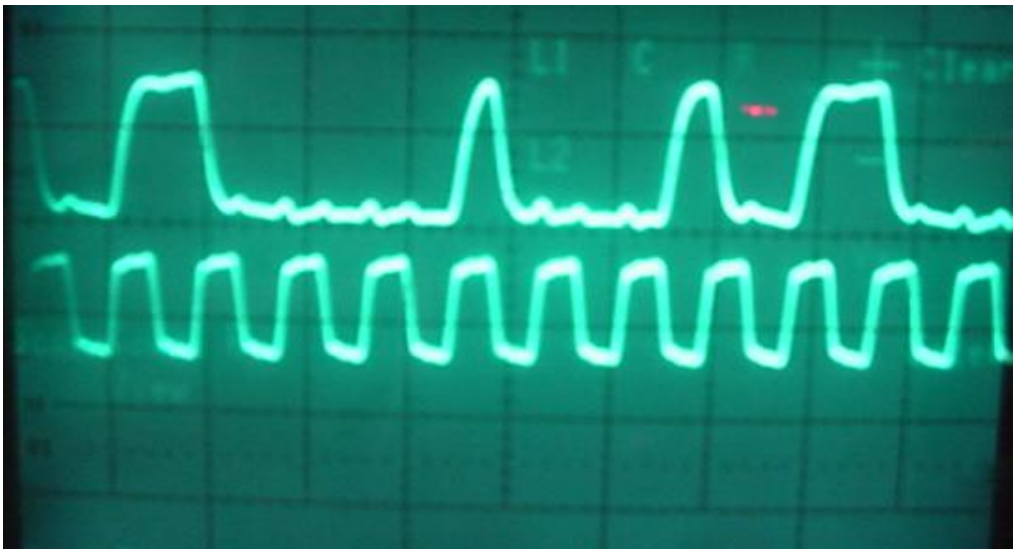


Fig. 4.6: *DiPPM coder's PRBS outcome (top trace), clock sequence (bottom trace).*

Figure 4.6 shows the DiPPM PRBS outcome compared with the clock. As can be seen, the positive bit of SET (10) is not the same width as that of the positive bit of DiPPM RESET (01). This is because synchronisation achieved with coax wires in the DiPPM coder, cannot give 100% satisfactory results. Synchronisation achieved for this element of the research process is considered satisfactory; 100% synchronization is described at a later point in this document. However, DiPPM coding was achieved, though pulses are not square or equal (as compared SET and RESET).

Outcome PSD of the DiPPM PRBS is [32]:

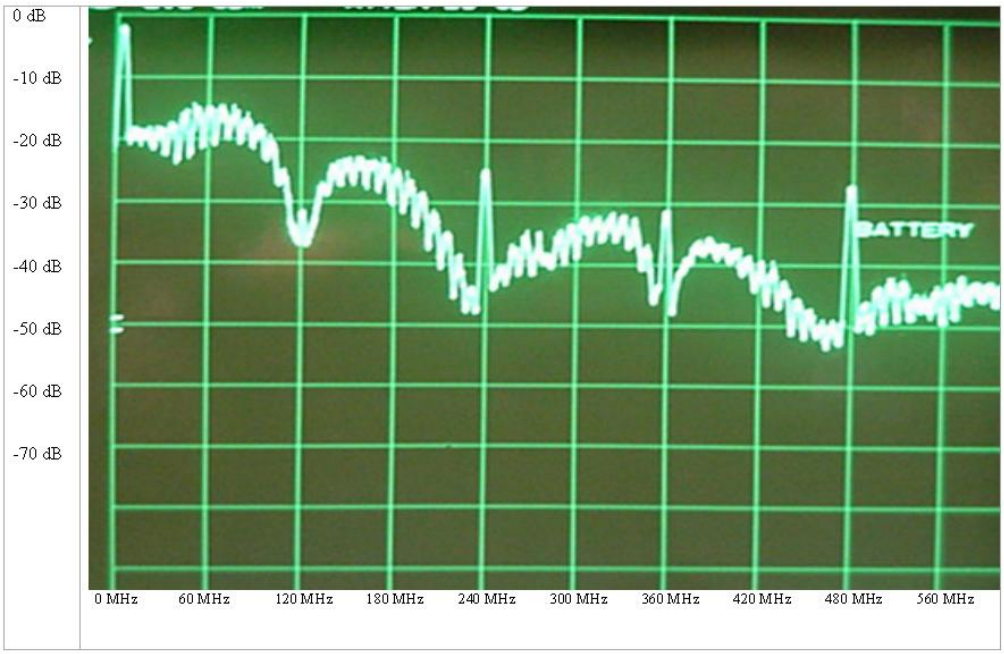


Fig. 4.7: *DiPPM PSD of PRBS sequence (hardware).*

DiPPM PRBS spectrum achieves results expected at the frequencies of 120 Mbit/s and the harmonics. When comparing the spectrum of figure 4.7 with that of figure 3.4, it is

noted that these do not correlate; figure. 4.7 provides the highest power output at the frequency of 240MHz, whilst in figure 3.4 the highest power output is at a frequency of 120 MHz. If the curves of the spectrum agree (figure 4.7), power appears between those curves when at figure 3.4 spectrum the powers appear in the middle of the curves. Further investigation will be described later on this disagreement of results.

4.2 DiPPM Decoder

4.2.1 Implementation of DiPPM Decoder

The DiPPM decoder converts DiPPM coded signal back to PCM format. The DiPPM decoder contains a NOR/OR gate, three NOR gates, a D Flip-Flop and a Direct Set/Clear component as shown at figure 4.8.

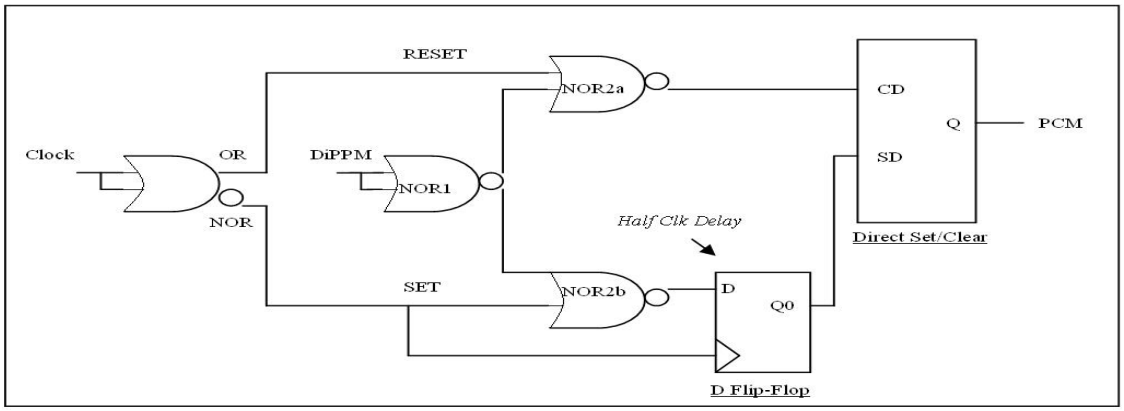


Fig. 4.8: *DiPPM decoder circuit.*

The clock signal was buffered (Appendix3-C) through a double OR gate. The synchronised output of the buffer passed through the same length of coaxial wires; one to

the coder and one to the decoder. Hence, both clock signals (outputs of the buffer) were synchronised. In common with the DiPPM coder, the clock and NOT clock signals were used, and were generated by passing the clock signal through a NOR/OR gate. The DiPPM input signal to the decoder was inverted by a NOR gate, prior to being gated with the clock and NOT clock to produce the SET and RESET sequence independently. The Direct SET/CLEAR component produces the PCM signal by giving a high amplitude when the sequence SET is one and zero when the RESET sequence is 1. Even when the SET and RESET sequences are independent, it can be seen in figure 4.9 that the correct decoding is not produced. Thus, a half clock period delay must be introduced with the use of a D Flip-Flop.

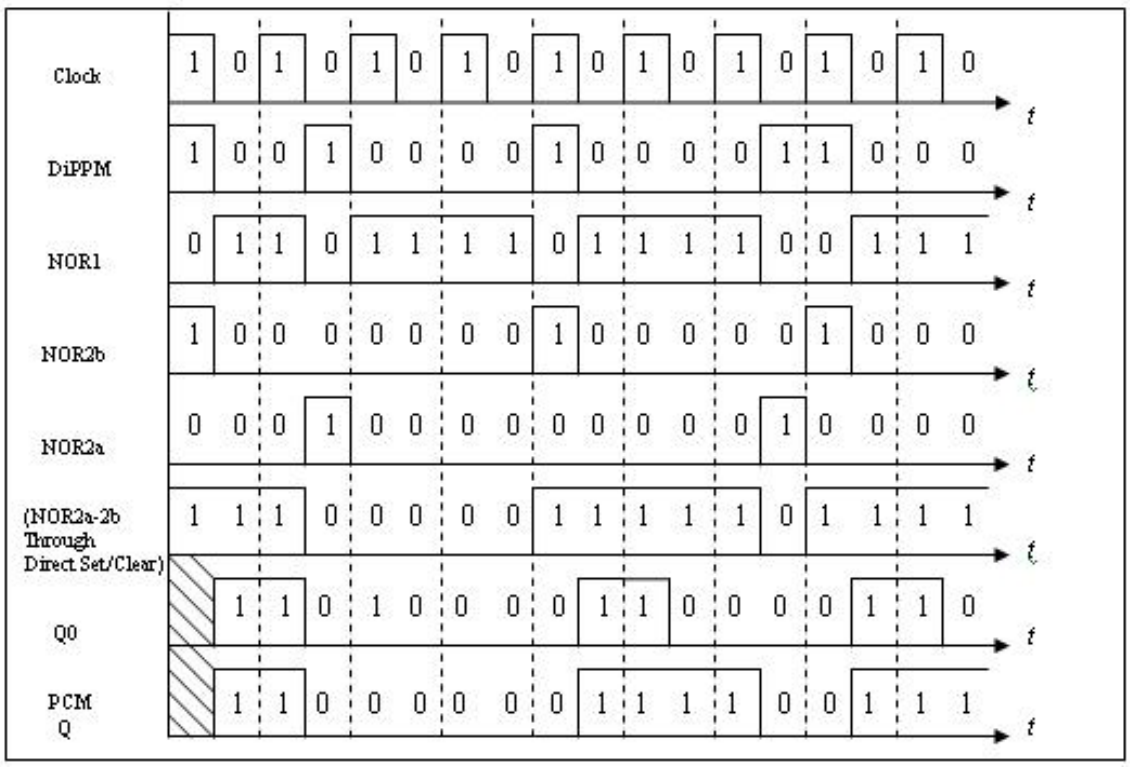


Fig. 4.9: *DiPPM decoder's waveforms.*

4.2.1.1 DiPPM Coder Technical Information

A synchronisation fault exists in the DiPPM decoder because the SET signal passes through one more component than the RESET signal; before both reach the Direct Set/Clear component. Thus, a delay was added from the D Flip-Flop component at the SET sequence.

4.2.2 Measurements and confirmation of DiPPM Decoder

The output of the DiPPM decoder (Appendix3-B), when input signals are a deterministic DiPPM and the clock is given in figure 4.10:

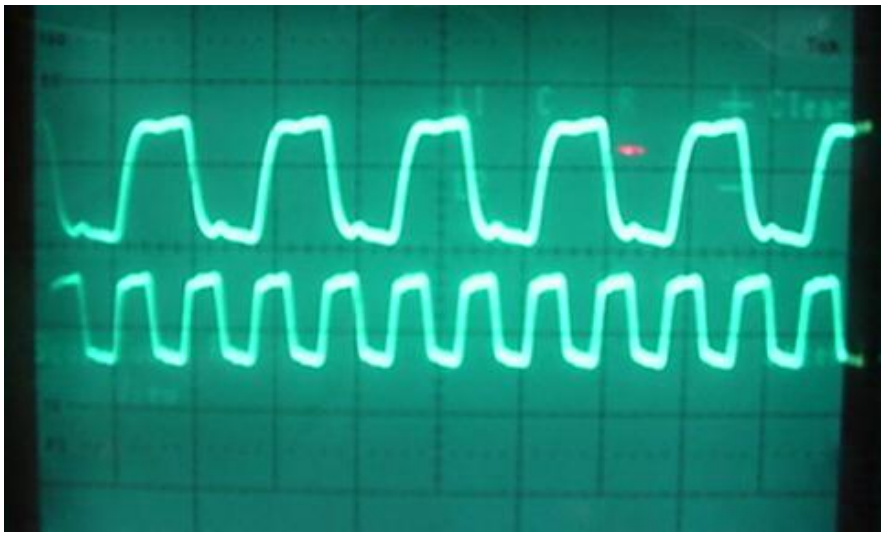


Fig. 4.10: *DiPPM decoder's deterministic outcome (top trace), clock sequence (bottom trace).*

The PCM formatted output of the DiPPM decoder is accurately synchronised with the clock. As anticipated, the start edge of every pulse commences when the clock period starts.

Figure 4.11 shows the DiPPM PRBS input of the decoder as compared with the output of the DiPPM decoder.

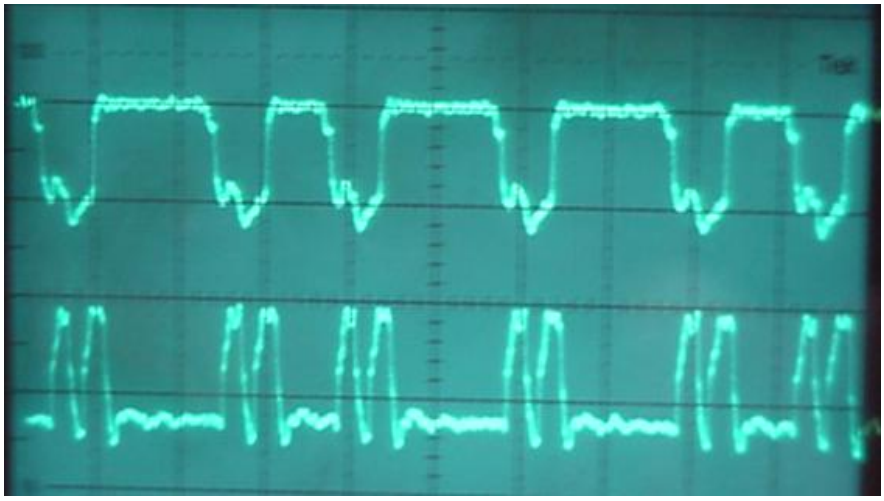


Fig. 4.11: *DiPPM decoder's output (top trace), DiPPM PRBS (bottom trace).*

From a comparison of waveforms it can be seen, based on the theory of section 3.2, that the DiPPM sequence has been decoded correctly back to PCM format. The delay between the two sequences is as a result of the delay in the decoder.

Comparison of the input signal from the PRBS generator, with that of the DiPPM decoder, is shown in figure 4.12; for both deterministic and PRBS signals.

(a) (b)

Fig. 4.12: a: *PCM deterministic input (top trace), PCM deterministic output (bottom trace),*

b: *PCM PRBS input (top trace), PCM PRBS output (bottom trace),*

Although the presented outputs of this chapter show that the DiPPM coder and decoder process correctly, measurement with the error detector had to be taken to ensure correct functioning. Outcomes of the Bit Error Rate Test set have shown that no errors were registered. ($0x10^{-6}$)

4.2 Conclusion

The construction of a DiPPM coder and decoder were presented for the first time. Theoretical representations of the DiPPM coder and decoder waveforms and practical outputs have been given. The two main sections of the DiPPM system, coder and decoder, have been developed with low cost ECL components.

Although it has been proved that coder and decoder are correct, previous publication [108] presented a theoretical DiPPM-ISI PSD (fig. 3.4) that does not agree with the experimental DiPPM PSD (fig. 4.7) of this thesis. Hence, more investigations need to be done on the DiPPM PSD that has been presented in this chapter.

Chapter 5

Dicode PPM Simulations

This chapter is concerned with the development of software and mathematical simulations. In order to achieve a reliable correlation between theoretical and experimental results, a window function was introduced to account for uneven mark: space ratio of the practical signal.

5.1 DiPPM Software Simulations

As no previous DiPPM coder simulation have been presented, it was necessary to construct a coder for the first time in order to investigate this PPM scheme. The DiPPM coder simulation (Appendix-A1) was programmed in MATLAB. Four subroutines were constructed for the DiPPM coder software simulation; time (Appendix-A2), binary (PCM) function (Appendix-A3) and two others derived from these. Binary function was

used to construct the third subroutine which gives the DiPPM (Appendix-A4). The fourth subroutine was built to synchronise the remaining three subroutines (Appendix-A5).

In the DiPPM coder simulation main program (Appendix A1), plot details were set (lines 1-6). In line 9, the main program calls the time routine; in lines 10 and 11, the binary routine. The difference between line 10 and line 11, is that the first represents a PRBS PCM sequence which is included in Matlab software, and the second calls the deterministic PCM sequence that exists in line 11. Either could be used at a time; thus the symbol (%) appears before the one which is not operational. DiPPM coding of the PCM and clock signal appear in the function of line 12. Signal synchronisation appears in line 13. Finally, simulation details such as frequency and amplitude are set in lines 15 to 17. The plot (lines 19-35) of the synchronised clock, deterministic PCM and DiPPM waveform appear as [31]:

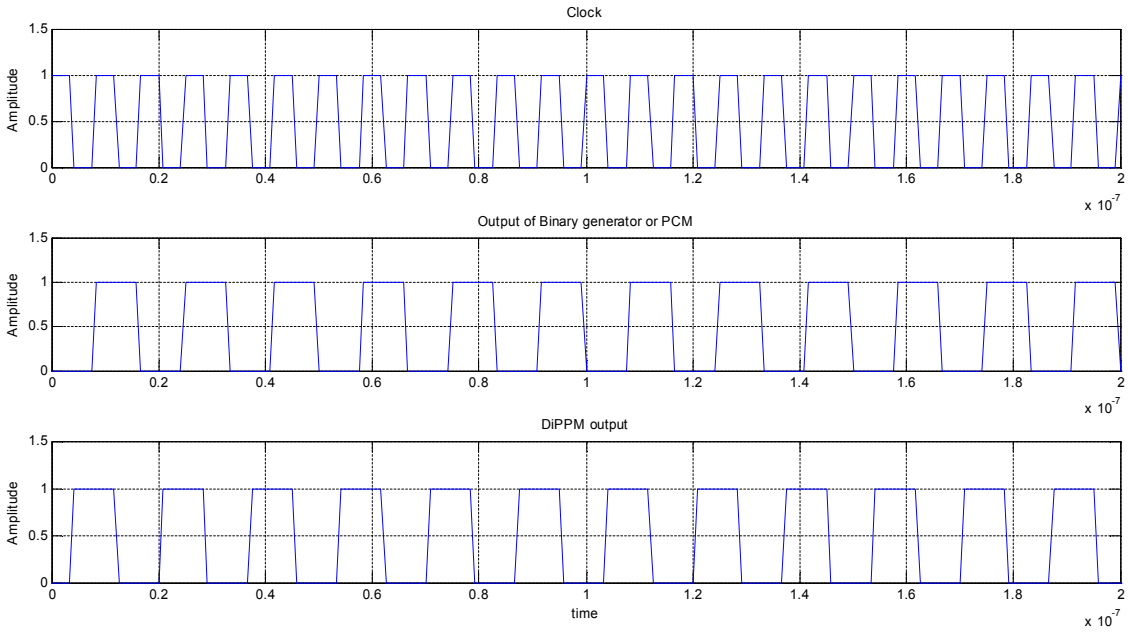


Fig. 5.1: *Simulation's plots of clock (top trace), deterministic PCM (middle trace), deterministic DiPPM (bottom trace).*

The synchronisation of the three signals (clock-PCM-DiPPM) and coding of the deterministic PCM signal to DiPPM were achieved (Figure 5.1). Used as an input to the software system the PRBS PCM waveform (line 10), the output waveforms of the DiPPM coder simulation program are [31]:

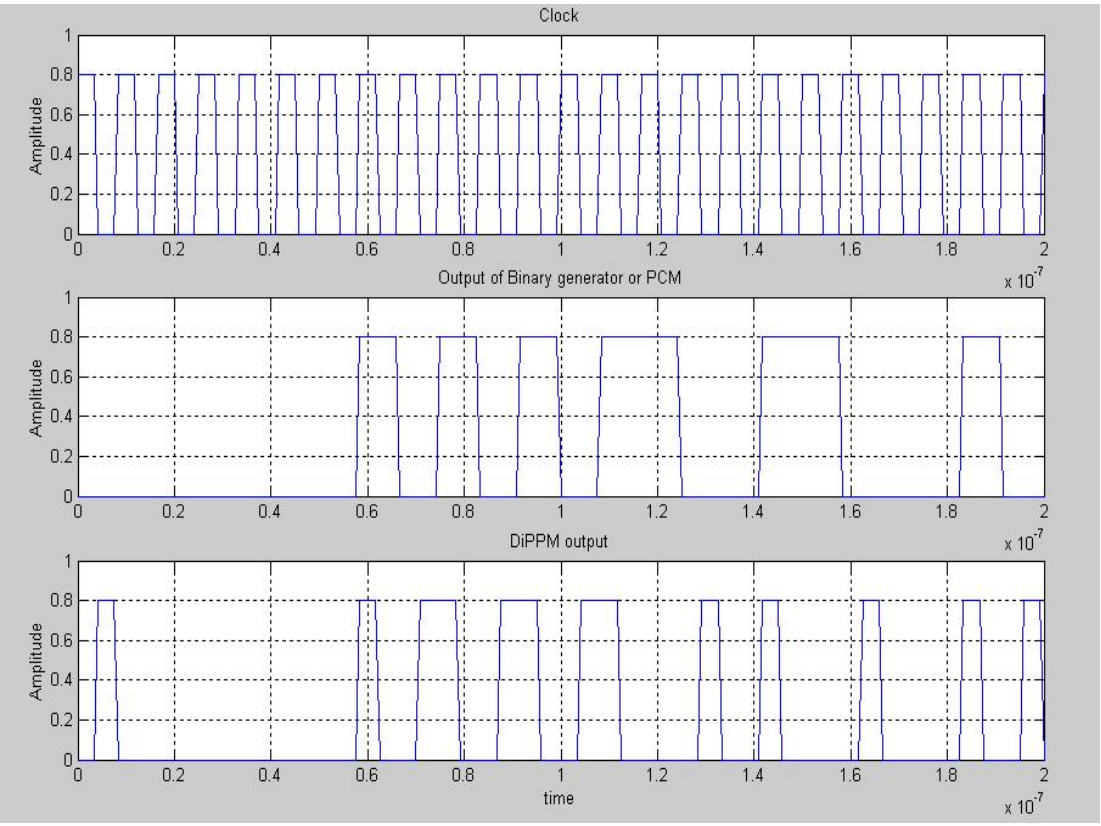


Fig. 5.2: *Simulation's plots of clock (top trace), PRBS PCM (middle trace), PRBS DiPPM (bottom trace).*

Figure 5.2 presents the clock, PRBS PCM and PRBS DiPPM waveforms. All three waveforms are synchronised. DiPPM coding of the PRBS signal is correct, based on DiPPM theory (Section 3.2). Figures 5.1 and 5.2 prove that DiPPM coder simulation was achieved and the waveforms are correct for any input PCM sequence.

Using the periodogram command of Matlab, where ‘’*dippm_signal_time’’* is the DiPPM synchronised waveform, the Power Spectrum Density (PSD) of the DiPPM signal is:

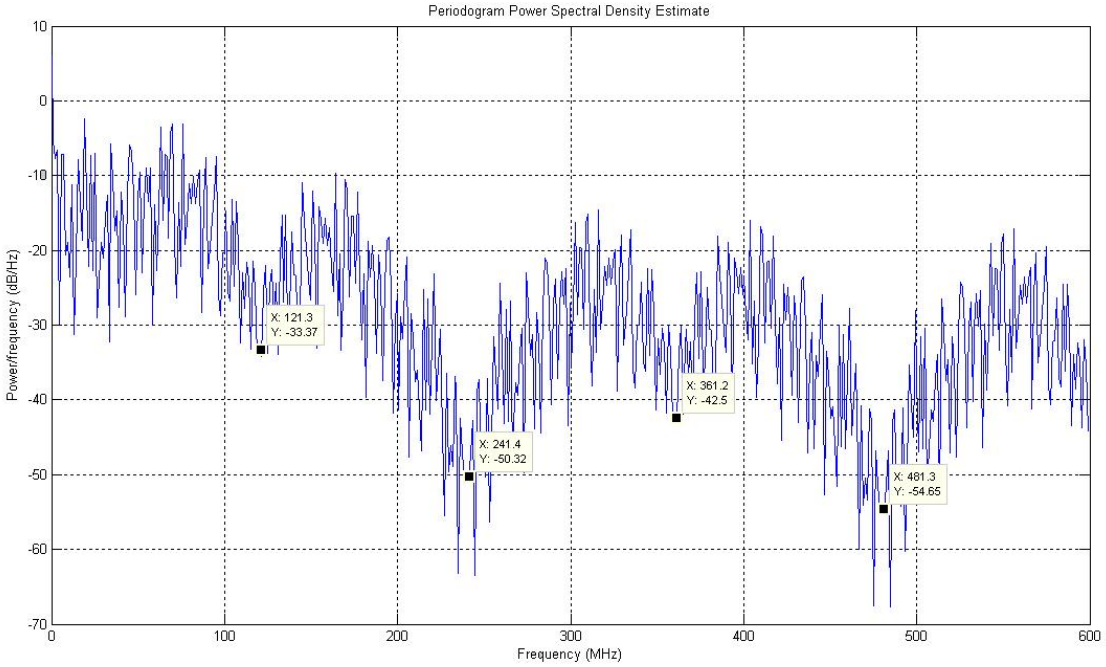


Fig. 5.3: *PRBS PSD of DiPPM coder Simulation*

Although it is proved that the DiPPM coder has been simulated correctly (fig. 5.1-5.2), the PRBS DiPPM PSD of figure 5.3 does not agree with that of the hardware DiPPM coder (fig 4.7) or with that of publication [108] (fig. 3.4). In order to verify that the software gives spectrums that do not agree with the experimental PSD, the PSD of the deterministic DiPPM waveform [31-32] was obtained:

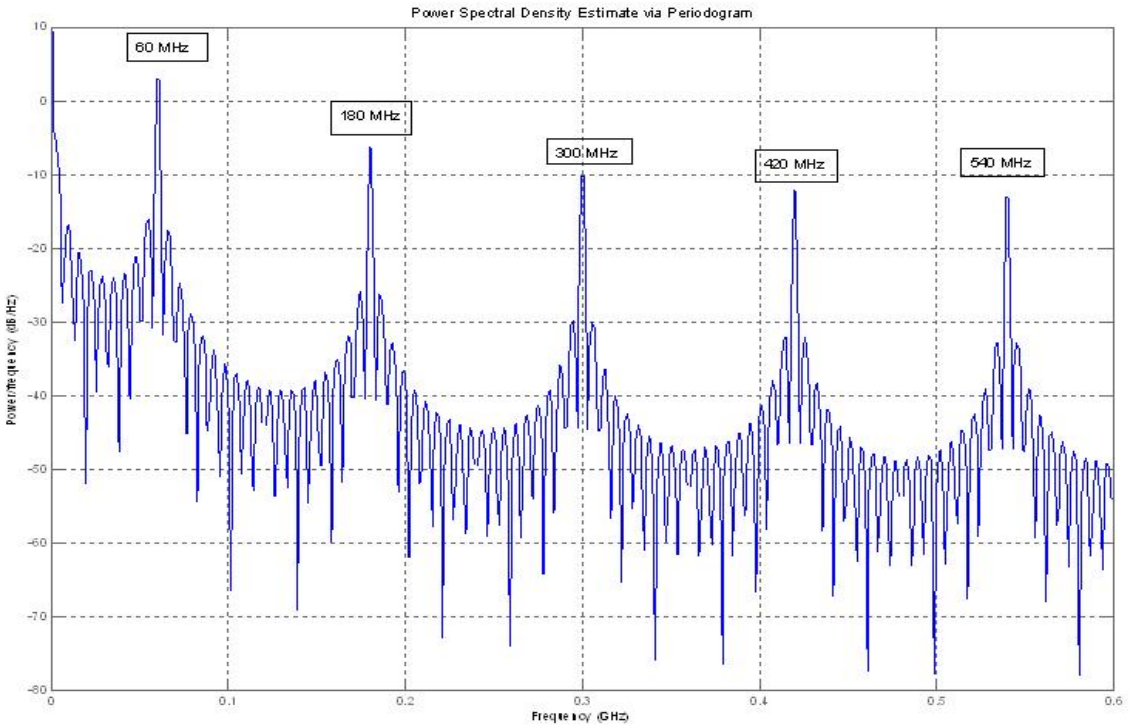


Fig. 5.4: *Deterministic PSD of DiPPM coder Simulation.*

The PSD of the DiPPM coder software gives PSDs that do not accurately agree with those of the experimental DiPPM coder, neither does the deterministic DiPPM spectrum (fig 5.4). Hence, further investigation was required.

5.2 DiPPM Mathematical Representation

Deterministic DiPPM waveform

The mathematical equation that represents DiPPM deterministic waveform is very easy to be implemented, because of its similarity with deterministic PCM waveform. The

deterministic DiPPM sequence appears to be the same as that of the PCM but delayed by 1.5 cycles of the PCM clock (Fig. 5.5) [32].

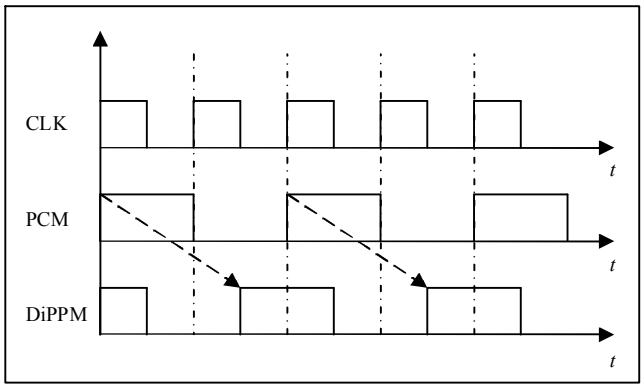


Fig. 5.5: *Relationship of deterministic PCM and DiPPM waveforms.*

$$h(t) = 1/T \sum_{n=-\infty}^{\infty} Cn \exp(j2\pi ft) \quad PCM\ sequence \tag{5.1}$$

$$h(t_{DiPPM}) = 1/T \sum_{n=-\infty}^{\infty} Cn \exp(j2\pi ft) \exp(j2\pi f 1.5) \quad DiPPM\ sequence \tag{5.2}$$

Equation 5.1 represents the Amplitude Spectrum of a PCM sequence. Shifting the time axis does not affect the amplitude spectrum (Cn) but it does affect the phase spectrum by adding a phase shift of $\phi s = \pi nft$ to each component in the series. This relationship can be described mathematically with equation 5.2, where *Cn* is the amplitude spectrum and *f* is the frequency. The equations that produce the deterministic DiPPM signals are X(t) when PCM signal starts with a positive pulse (logic 1) and X(-t) when PCM input signal starts with no pulse (logic 0) [32]:

$$X(t) = \frac{A}{2} + \frac{2A}{\pi} * \sin\frac{\omega td}{2}\{\cos(\omega t) - \frac{1}{3}\cos 3\omega t + \frac{1}{5}\cos(5\omega t)....\} \quad (5.3a)$$

and

$$X(-t) = \frac{A}{2} + \frac{2A}{\pi} * \sin\frac{\omega td}{2}\{-\cos(\omega t) + \frac{1}{3}\cos 3\omega t - \frac{1}{5}\cos(5\omega t).....\} \quad (5.3b)$$

where A is the amplitude, ω is equal to $2\pi f$ and td is the width of the pulse.

A simulation program was constructed using Matlab software (Appendix-B1), to plot the 5.3 equations. As previously (Appendix-A1), it has been set the details of the plot (lines 2-7) but instead of the input PCM sequences the X(t) and X(-t) equations have been set (lines 17-18). In lines 8 to 13, details of the equation have been set such as amplitude, frequency, width and period. Finally, in lines 21 to 36, it has been asked the waveforms of the equations 5.3 to be plotted. Figure 5.6 shows the plots of these equations [32].

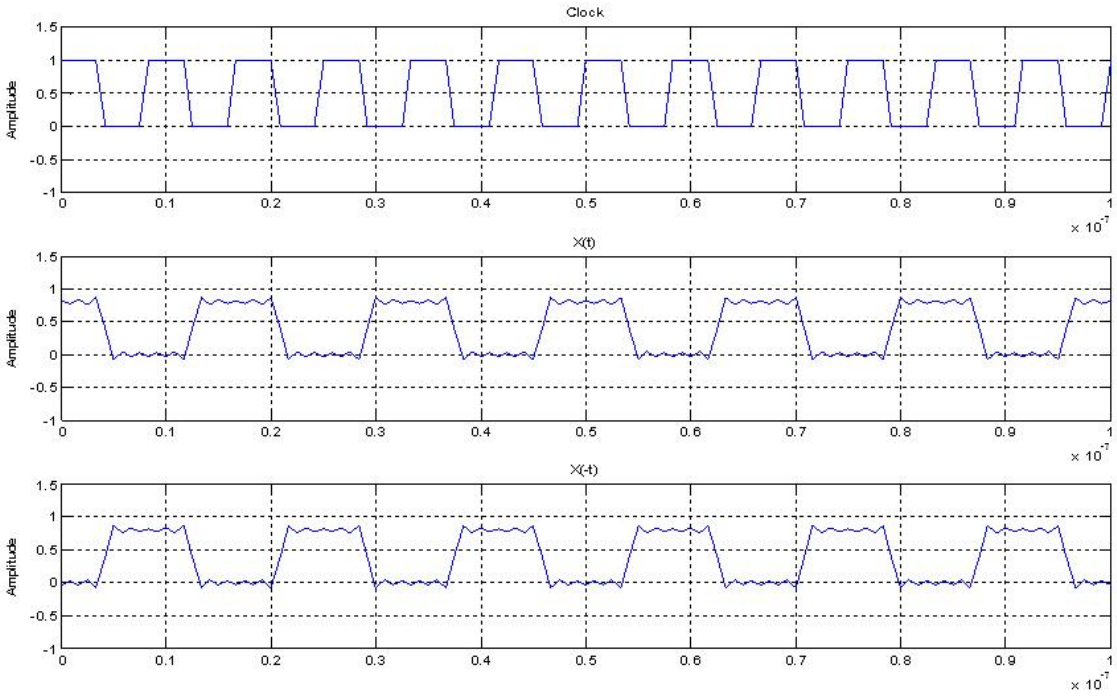


Fig. 5.6: *DiPPM deterministic sequence: X(t) (middle trace), X(-t) (bottom trace).*

The plot waveforms of figure 5.6 confirm that equations 5.3 are correct and the deterministic DiPPM sequence has been mathematically represented. Running the periodogram (spectrum) of equations 5.3, the PSD that has been plotted is [32]:

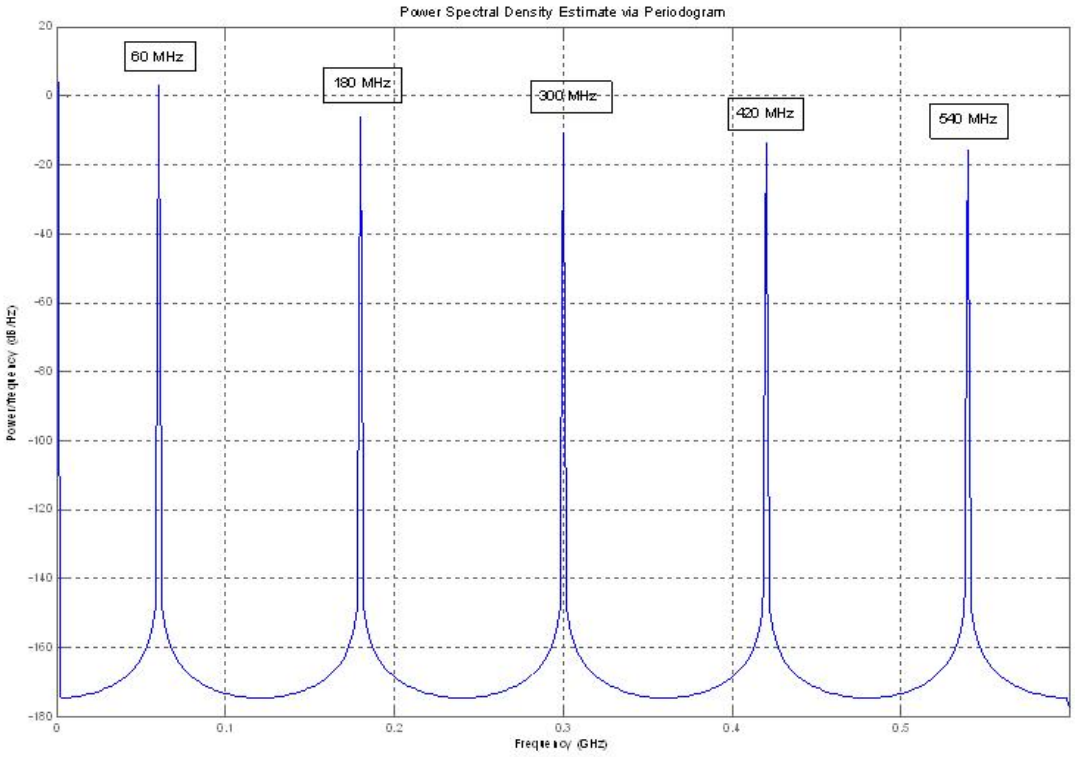


Fig. 5.7: *Deterministic DiPPM PSD of X(t) and X(-t).*

The spectrum of X(t) and X(-t) does agree with the PSD of the software deterministic DiPPM simulation (fig. 5.4) but still does not agree with spectrum of the experimental DiPPM coder (fig. 4.4). From the comparison of spectrums 5.7, 5.4 and 3.4 it can be noticed there is a similarity in the powers (in all three spectrums the powers decrease as the frequency increases). On the other hand the spectrum of figure 3.4 gives the powers at the middle of the curves while the other two do not. Thus, figure 3.4 does not give the spectrum of deterministic DiPPM signal. From the comparison of figure 4.4 with that of

5.4 and 5.7, it should be noted that the experimental spectrum (fig. 4.4) yields a greater number of harmonics than the PSDs of the deterministic DiPPM of software and the mathematical simulation (fig. 5.4, 5.7). The harmonics in figure 4.4 were expected because of the shape of the DiPPM pulses (fig. 4.5). The spectrums of figures 5.4 and 5.7 do not present the harmonics occurring in the experimental spectrum. It appears that the software gives the spectrum without the factor of the uneven mark space. Hence, further investigation has to be undertaken on the periodogram of the software in the area of the pulse shape used. It appears from those spectrums (fig 5.4 and 5.7) that the use of rectangular pulses does not generate sub-harmonics as expected.

PRBS DiPPM waveform

A random sequence cannot be represented by a mathematical equation. The pulses always change place in a random sequence so the equation has to change all the time. On the other hand, a pseudo-random binary sequence (PRBS) is a random sequence that always will be repeated after a number of bits. In this way, it could be said that a 64 bits sequence (which is the number of bits that the signal generator repeats the PRBS sequence), that gives every 8 bits different binaries, could be called PRBS sequence.

For this DiPPM PRBS sequence, all the DiPPM symbols must be used at least once. These are the SET (10), the RESET (01), the 'no-pulse' (00) but the combination of RESET and SET (RS-0110) that occur in a DiPPM sequence as well. This can be achieved by adding the equations (5.4-a, b, 5.5-a, b) for four independent pulse trains [32].

$$\text{Re}\,set1 = \frac{Atd}{T} + \frac{2A}{n\pi}\sin(\frac{n\omega td}{2})\cos(\,n\omega(t-1.4td)) \tag{5.4a}$$

$$\text{Re}\,set2 = \frac{Atd}{T} + \frac{2A}{n\pi}\sin(\frac{n\omega td}{2})\cos(n\omega(t-7.4td)) \tag{5.4b}$$

$$Set1 = \frac{Atd}{T} + \frac{2A}{n\pi}\sin(\frac{n\omega td}{2})\cos(n\omega(t-4.4td)) \tag{5.5a}$$

$$Set2 = \frac{Atd}{T} + \frac{2A}{n\pi}\sin(\frac{n\omega td}{2})\cos(n\omega(t-8.4td)) \tag{5.5b}$$

Where A is the amplitude of the pulse, td is the width of the pulse, T is the clock period, and ω is equal to 2πf where f is the clock frequency (1/T). A DiPPM pseudo-random binary sequence (PRBS) is produced when equations 5.4a, 5.4b, 5.5a and 5.5b are added. The phase shifts of the equations have been calculated the way that no SET will be followed by SET and no RESET will be followed by RESET. Adding these equations gives the PRBS DiPPM signal as [32].

$$\cdot\ DiPPMprbs = \frac{Atd}{T} + \frac{2A}{n\pi}(\sin(n\omega td/2) + (\cos(n\omega(t-1.4td)) + \cos(n\omega(t-7.4td)) + \cos(n\omega(t-4.4td)) + \cos(n\omega(t-8.4td))) \tag{5.6}$$

The software that plots the equations 5.4, 5.5 and 5.6 is the one in Appendix-B2 (similar to the software that plots the deterministic DiPPM equations in Appendix-B1). The setting for the plot appears in lines 1-5 where in line 6 it has been set the time scale for

the waveform plots. In lines 8 to 16 it has been set the frequency, amplitude width details of the PRBS DiPPM pulse trains equations. The part of the equation that represents the Fourier harmonics of its four pulse train is given in lines 19-22 and finally in line 23 these four pulse trains have been added to produce the PRBS DiPPM sequence (5.6). Plotting the pulse trains, the PRBS DiPPM sequence in comparison with the clock (lines 30-60), the waveforms are [32]:

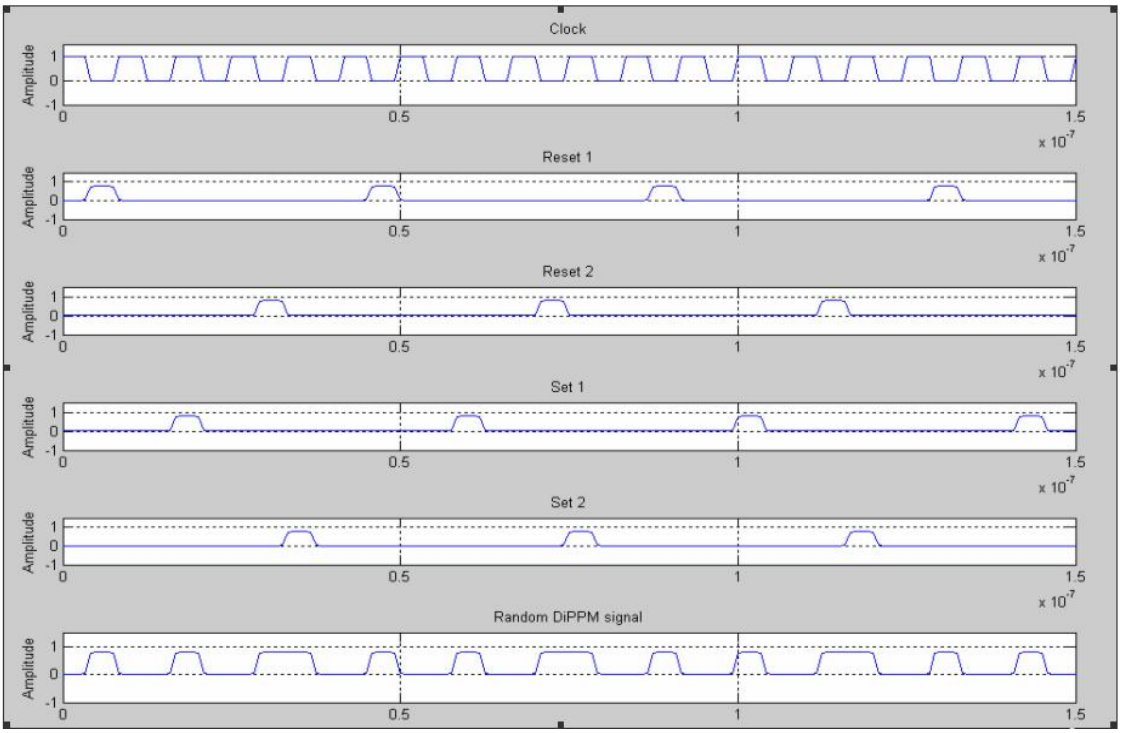


Fig. 5.8: *Internal clock (1st top trace), pulse train 5.4a (2nd top trace), pulse train 5.4b (3rd top trace), pulse train 5.5a (4th top trace), pulse train 5.5b (5th top trace), PRBS DiPPM 5.6 (bottom trace).*

Equation 5.6 produces a 10 bit PRBS DiPPM waveform. In order to make a longer PRBS sequence, a 12 bit time window is considered for analysis. In this way the sequence appears to repeat every 60 bits (fig. 5.9) [32].

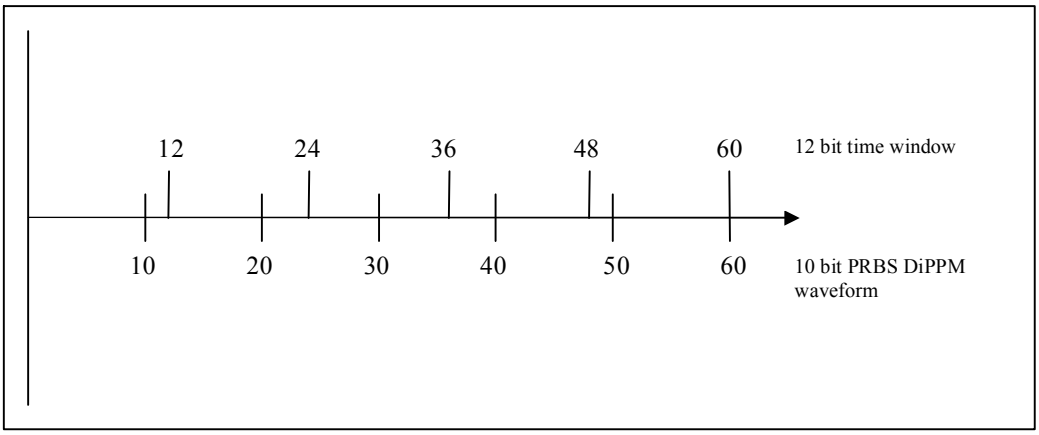


Fig. 5.9: *DiPPM PRBS of 60 bits.*

Plotting just the PRBS DiPPM waveform in comparison with the clock (Appendix-B2, lines 61-71) gives:

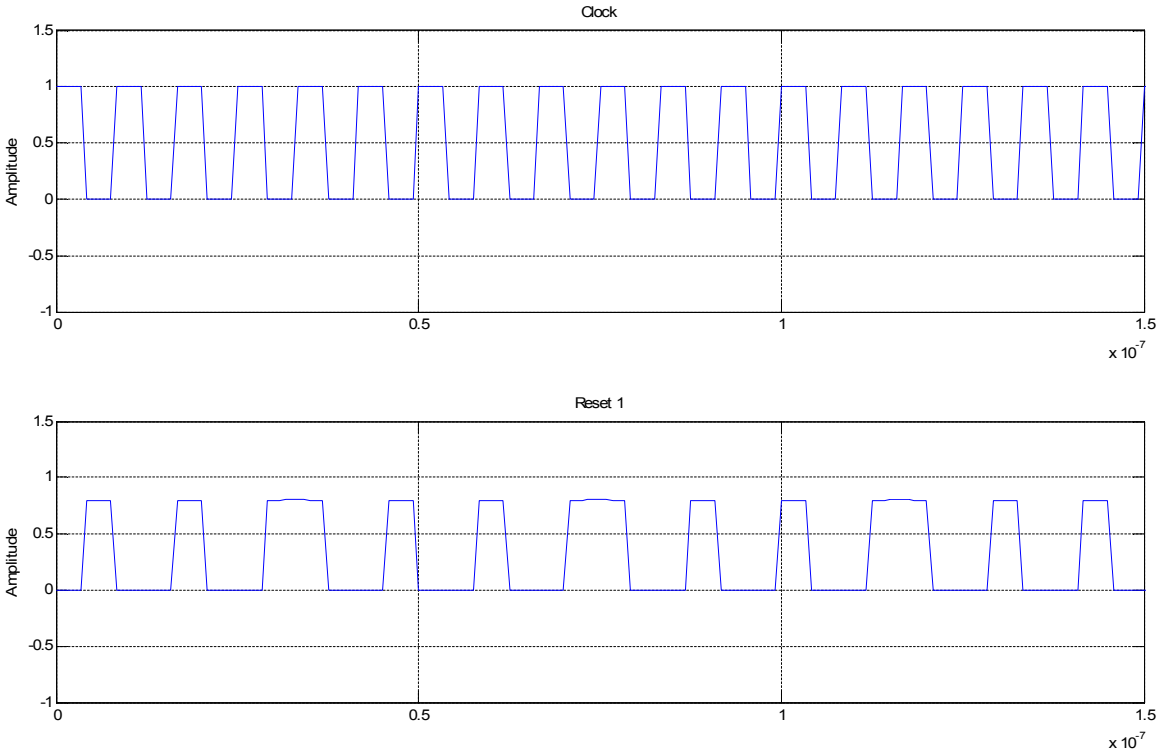


Fig. 5.10: *Internal clock (top trace), PRBS DiPPM 5.6 (bottom trace).*

From figure 5.10 (where PRBS DiPPM waveform is the same with that of figure 5.8) it can be seen that no SET (10) follows any SET and no RESET (01) follows any RESET,

while the combination of RESET/SET (0110) exists in the sequence. It may appear that the sequence R/0/S/0RS/0R, is repeated constantly. However, as figures 5.9 and 5.10 show, the pulses have different positions in every 6 clock period. Hence, the binary '010010011001' which exists in the first 6 clock cycles (fig. 5.10), differs from '001001100100' (which is the sequence that follows) and so on, until the sequence reaches 64 bits where it is repeated.

Calling from the software the spectrum of the PRBS DiPPM equation (Appendix-B2, line 73), the PSD of equation 5.6 is:

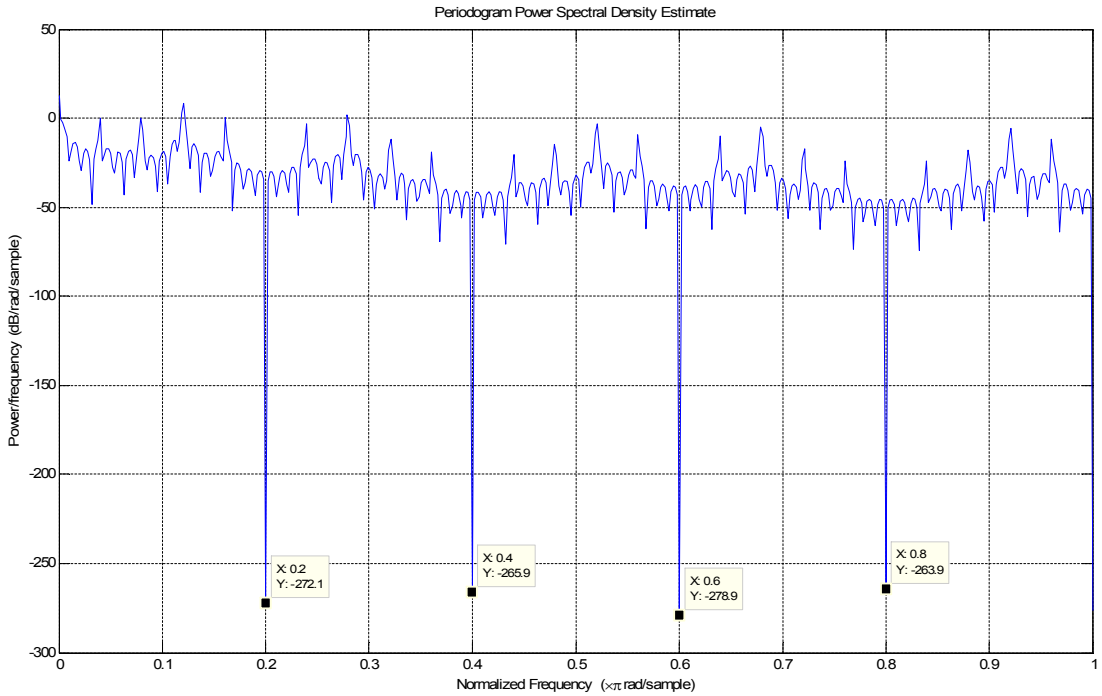


Fig. 5.11: *PRBS DiPPM PSD of equation 5.6.*

The spectrum of the PRBS DiPPM equation is the same as that of the PRBS DiPPM simulation (fig. 5.3). The harmonics that appear on the curves of the spectrum (fig. 5.11) are not powers, but harmonics that occur because the PRBS DiPPM equation does not

take as many samples as the software simulation. Infinity harmonic samples were added (Appendix-B2, line 19-22) to exactly simulate the software (Appendix-A1). Both spectrums (fig. 5.3 and 5.11) produce the same curves. Additionally, same powers have been given, both below -50dB (the spectrum of figure 5.3 much lower powers that its figure presents). Plot resolution is not fine enough to show all the details. Comparison of these two spectrums, it has been proved once again that the software gives correct outcomes. Hence the periodogram command of Matlab needs investigating further.

5.3 DiPPM Windowing Method

As can be seen from figures 5.3, 5.4, 5.7, and 5.11, results obtained from simulation and mathematical analyses do not match those from the practical system (fig. 4.4 and 4.7). As discussed in section 5.2, this disagreement may have occurred because of Matlab periodogram settings. On closer investigation of the periodogram command,

```
periodogram(X, window, 'one-sided', nfft, fs);
```

it can be seen that the periodogram uses the signal 'X' at which is going to give its spectrum. In the second part of the periodogram there is the word 'window'. This is the vector which specifies the coefficients of the window used in computing a modified periodogram of the signal used as input; in this case the DiPPM. Both input signals DiPPM and window have to be vectors of the same length. When no specific window has been set (the user has not supplied the second argument window or it has been set as empty vector '[]'), a rectangular window is used by default. The "one-sided' element of

this command provides the spectrum type.'One-sided', returns the one-sided mean-square spectrum which contains the total signal power, in half the Nyquist range. The integer 'nfft' specifies the length of the Fast Fourier Transform (FFT), while 'fs' is the sampling frequency specified in hertz (Hz). This computes the PSD vector 'X' and the corresponding vector of frequencies. The power spectral density which has been produced is calculated in units of power per Hz. If there is no specification of 'fs', the sampling frequency defaults to1 Hz.

Hence, incorrect spectrum production occurred when the software model used a window that did not correctly simulate the shape of the practical data pulses [31, 109]. In order to achieve uneven mark:space ratio signals generated in practice, the window was modified, enabling the computer to predict results containing even harmonics. This window was found as follows [32].

Equation 5.7 gives the PSD of an arbitrary signal sequence [X1……..Xn] corresponding to the DiPPM signal.

$$S(e^{j\omega}) = \frac{1}{n}\left|\sum_{\mu=1}^{n} X\mu * e^{-j\omega\mu}\right|^2 \tag{5.7}$$

In order to match the harmonics of the PSD to those of the spectral analyzer, a window [w1,…., wμ] was introduced to weight the signal sequence. Thus equation 5.7 becomes

$$S(e^{j\omega}) = \frac{\frac{1}{n}\left|\sum_{\mu=1}^{n} W\mu * X\mu * e^{-j\omega\mu}\right|^2}{\frac{1}{n}\left|\sum_{\mu=1}^{n} W\mu\right|^2} \tag{5.8}$$

Matlab uses a filter when it performs spectral analysis (Appendix2 - A). This filter is used to alter mark-space ratio of the pulse and affect rise and fall times. The filter impulse response (required in order that the Matlab simulation agrees with the measured spectrum) is given in equation 5.9.

$$W = \frac{15}{1} * \frac{1}{2.4}\sin(2\pi ft) + \frac{1}{0.8}\sin(4\pi ft) - \frac{1}{1.5}\sin(6\pi ft) + \frac{1}{1.2}\sin(8\pi ft) \tag{5.9}$$

where ω is equal to $2\pi f$. Figure 5.8 shows the spectrum obtained from equation 5.9 and the mathematically predicted spectrum of the deterministic DiPPM signal [32].

Using the window equation 5.9 instead the default window '[]' for all the DiPPM simulations, the spectrums of deterministic and PRBS DiPPM waveforms are [32]:

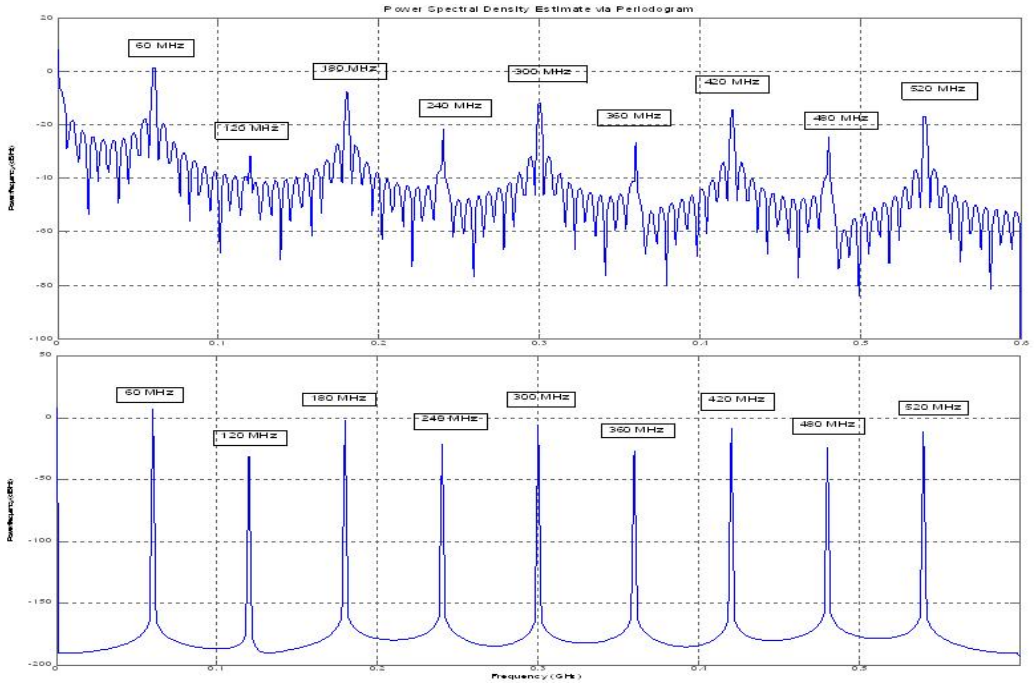


Fig. 5.12: Deterministic DiPPM spectrums from simulation (upper) and equation 5.3 (lower) with the use of window.

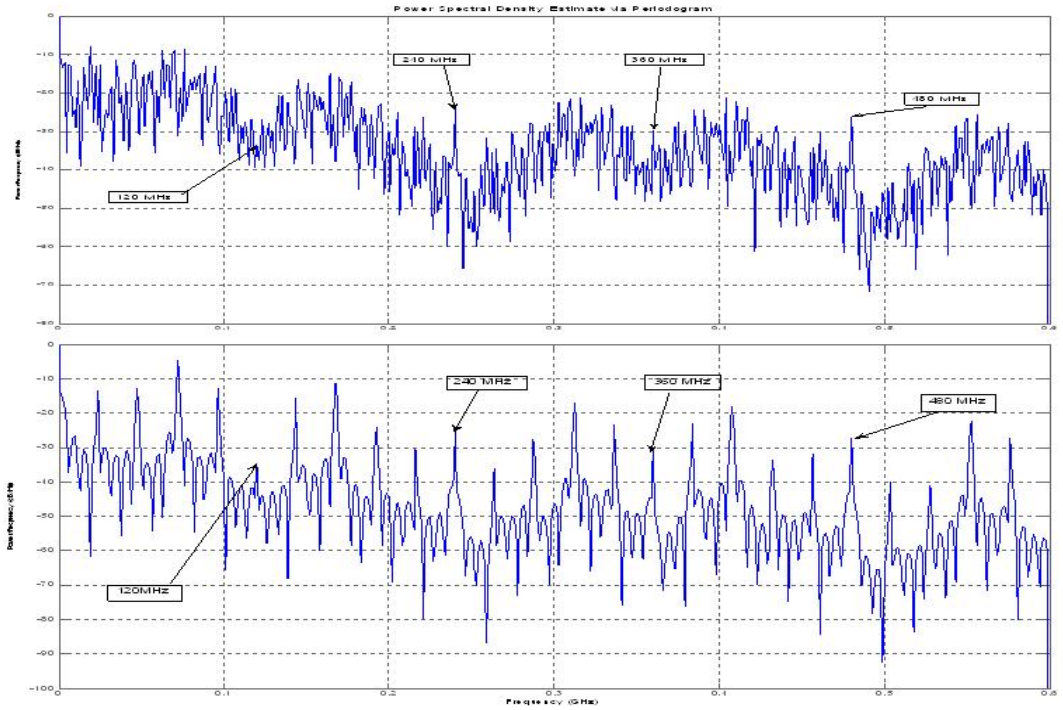


Fig. 5.13: PRBS DiPPM spectrums from simulation (upper) and equation 5.6 (lower) with the use of window.

The deterministic DiPPM spectrums of figure 5.12 correlate with the experimental deterministic DiPPM PSD in both powers and shape of graph. As do the PRBS DiPPM spectrums of figure 5.13, with the spectrum of the experimental PRBS DiPPM. Hence, waveforms and spectrum outcomes of the experimental DiPPM coder are correct. Further investigations indicate that the window function used in the software, as part of the routine that predicts the PSD of a signal, did not accurately reflect the properties of the practical pulse. Consequently, the window was redefined (equation 5.9) so that agreement was obtained between experimental, theoretical and computer predicted results. Results obtained with a coded PRBS signal show a high degree of correlation, only when the window is used. When the same software simulation was run with a random PCM input sequence, the spectrum of random DiPPM was as follows:

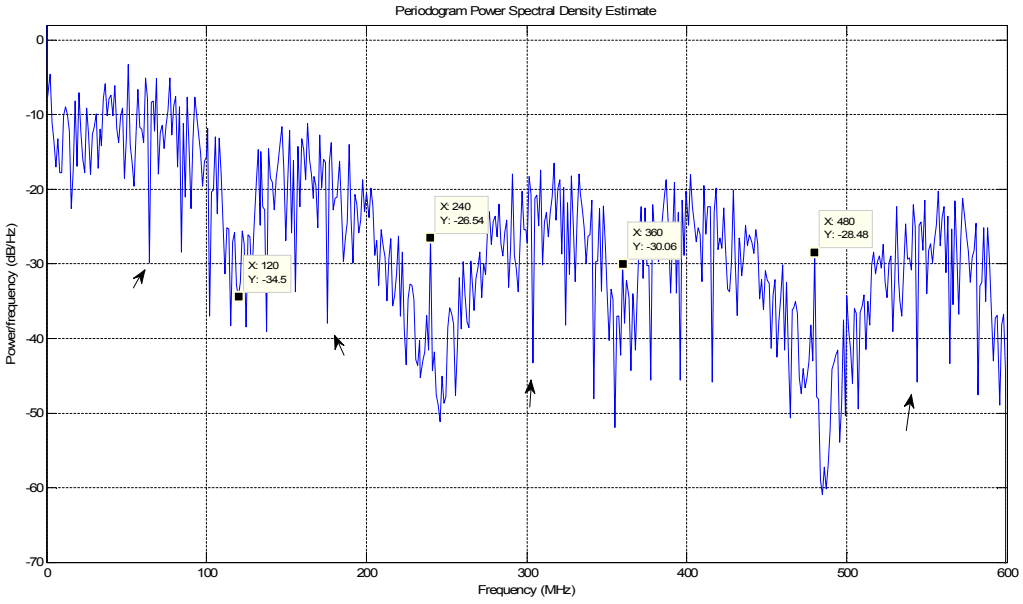


Fig. 5.14: *Random DiPPM spectrums from simulation with the use of window.*

The random DiPPM spectrum of figure 5.14 correlates with that of PRBS DiPPM. Powers appear at 120 MHz and multiples of, as expected. It can be seen from figure 5.15, there is a very close match between the four results (experimental-PRBS simulated-random simulated-Maths).

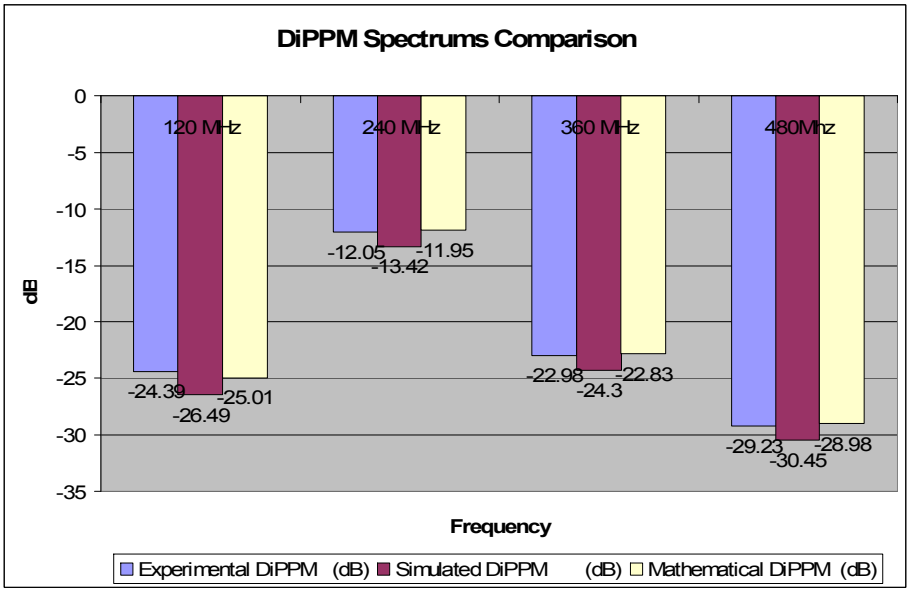


Fig. 5.15: DiPPM spectrum powers comparison

In a previous publication [108], a derivation was presented for the PSD of a PRBS DiPPM signal. The author of the publication presented the PSD of the DiPPM-ISI (section 3.1); not the DiPPM spectrum (section 3.2) as contained in this thesis. It is helpful to examine the differences between the spectrum graphs (figures 3.4 and 4.7 (5.13)). The results in publication [103] agree with the work presented in this thesis, only if the pulses are ideal (i.e. no window is used). Still the spectrum [108], presents the powers in the middle of the curves and the author did not refer to powers that were lower than -60 dB. (Analysis of differences and similarities are contained in Appendix 4.) There is similarity between this PSD and that of figures 5.3 and 5.11. As there are components

at 120 MHz and 240 MHz, the hypothesis of publication [108], to extract the DiPPM frame-rate component directly from the pulse stream is still possible.

The correct PSDs of experimental DiPPM have been presented and have been confirmed by software and mathematical simulations. This shows that the DiPPM coder produces correct outputs and the construction of the component that recovers the clock from the DiPPM sequence will be necessary, which is the DiPPM timing extraction.

Chapter 6

Dicode PPM Timing Extraction

Timing extraction, slot and frame synchronisation of DiPPM are simple to be achieved because of the position of the pulses in the frames. The falling edge of SET and rising edge of RESET pulses are coincident and thus generate a strong timing content. Hence, DiPPM appears to be more efficient than previous coding techniques. In order to achieve clock recovery from a random DiPPM sequence and synchronise the extracted clock with the DiPPM signal, a *second-order* phase-locked loop (PLL) system was employed in the DiPPM Timing Extraction circuit (Appendix3-D) shown in figure 6.1 [33].

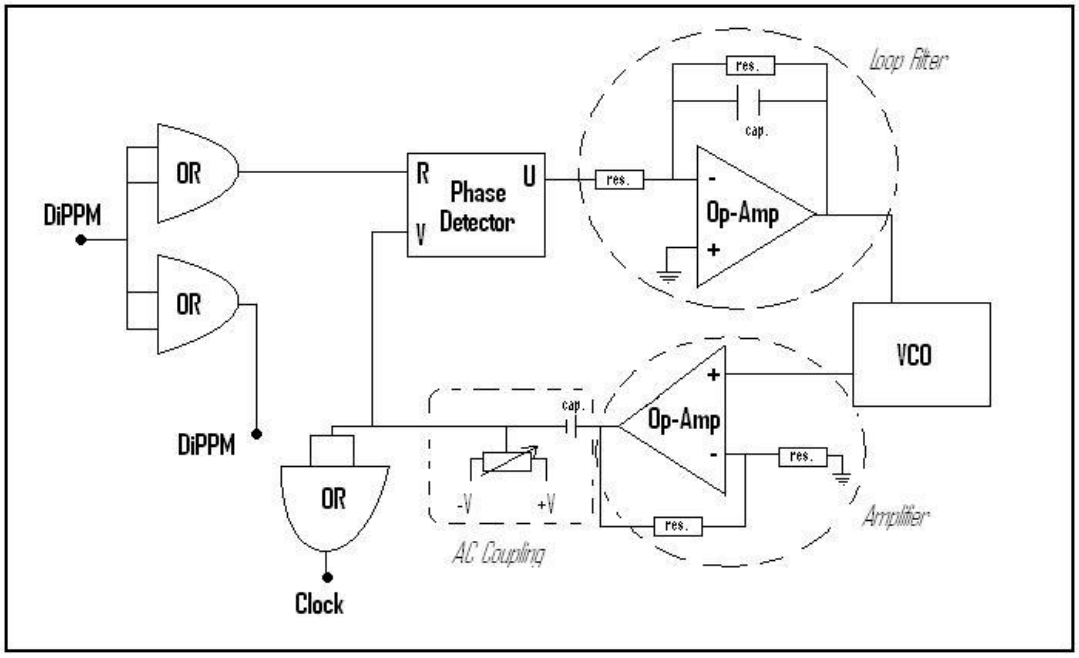


Fig 6.1: *DiPPM timing extraction diagram.*

Two OR gates were used to buffer the input DiPPM signal (Fig 6.1). The output of one OR gate feeds the DiPPM decoder. The output of the remaining OR gate goes to the PLL system, in order to produce the synchronized clock signal. The PLL circuit consists of a phase detector, an active low-pass filter, a Voltage-Controlled Oscillator (VCO) and an amplifier for any necessary amplification of VCO output signal (fig. 6.1). As the practical DiPPM coder was constructed using ECL circuitry, and the phase detector is an ECL component, an AC coupling circuit was added to shift the VCO output signal to ECL standards [33].

6.1 DiPPM Clock Recovery Process

The phase detector used in this circuit (fig 6.1), is the *'Max 9382'* ECL Phase-Frequency detector (Datasheets). This has two types of input: Single-Ended Reference input (R) and Single-Ended VCO input (V). When both are compared, the outcome of Phase detector (U) is that of figure 6.2.

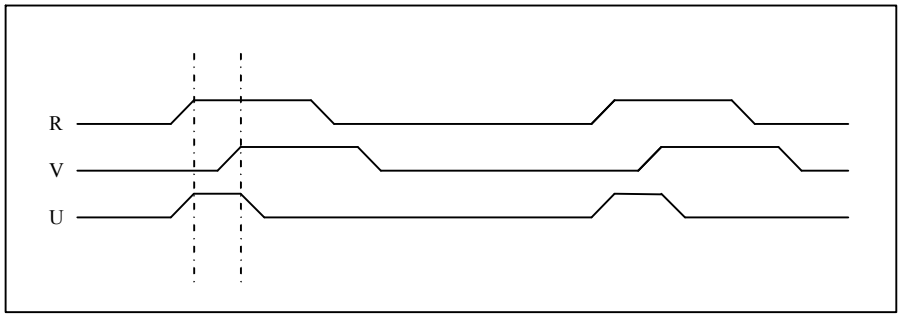


Fig 6.2: *Phase Detector Process.*

The DiPPM signal passed through the phase detector without any change (as no Single-Ended VCO input existed at that point) and was received by the *Loop Filter* (OpAmp-741) (datasheets). The *Loop Filter* (fig. 6.1) is the primary building block which determines the dynamic performance of the loop. It removes any noise and high-frequency components from the output voltage of the phase detector, thus giving an average DC voltage:

$$\frac{Vout}{Vin}(dB) = -20\log[\omega^4 + 2\omega^2(2\zeta^2 - 1) + 1]^{1/2} \qquad (6.1)$$

Where Vin is the loop input voltage, Vout is the loop output voltage, ζ is the damping factor and ω is the ratio of the input frequency (ω_i) to the undamped natural frequency (ω_n). The cutoff frequency of the active *Loop Filter* is 100 kHz (making the system stable). The filter cut-off frequency is given by

$$\omega_{LPF} = \frac{1}{R_2 C} \tag{6.2}$$

Where R_2 is the resistor in parallel with the capacitor C. For the stability and the accuracy of the output of the *Loop Filter,* a variable resistor was connected to the positive (+) input of the Op-Amp (fig. 6.3) to introduce an offset for the VCO.

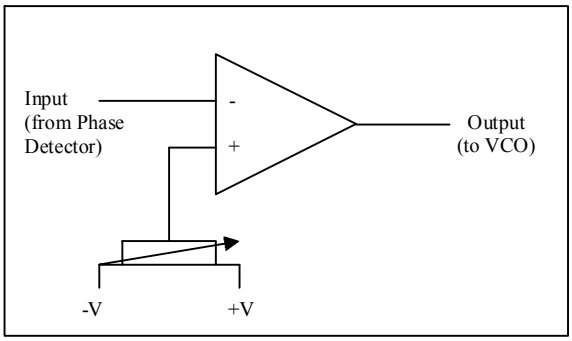


Fig 6.3: *Loop Filter transformed to Comparator.*

The third component is the VCO *'Max2608'* (datasheets). The mathematical representation of the VCO is:

$$\omega_0 = K_0 V_f \tag{6.3}$$

Where ω_0 is the VCO output frequency, V_f is the VCO control voltage from the *Loop Filter* and K_0 is the VCO conversation gain. As the practical DiPPM coder (chapter 4) operated with a PCM data rate of 120 Mbit/s, the output data rate of the DiPPM was 240 Mbit/s. Thus, the VCO must produce a clock at the same data rate (fig.6.4) as that of the DiPPM signal. A necessary amplification of the VCO output took place in order to reach the mark/space levels by Max4305 amplifier (fig.6.1). As the output levels of the VCO are in TTL format, an AC coupling circuit (fig.6.1) was used to shift the amplified VCO output to the ECL level. The VCO output and the DiPPM input were compared in the phase detector (fig.6.1). The output of the phase detector (fig.6.4) or *error voltage,* is an average DC voltage, proportional to the difference in frequency (*DiPPMf -VCOf*) and phase $\Delta\phi$ of the inputs DiPPM and VCO.

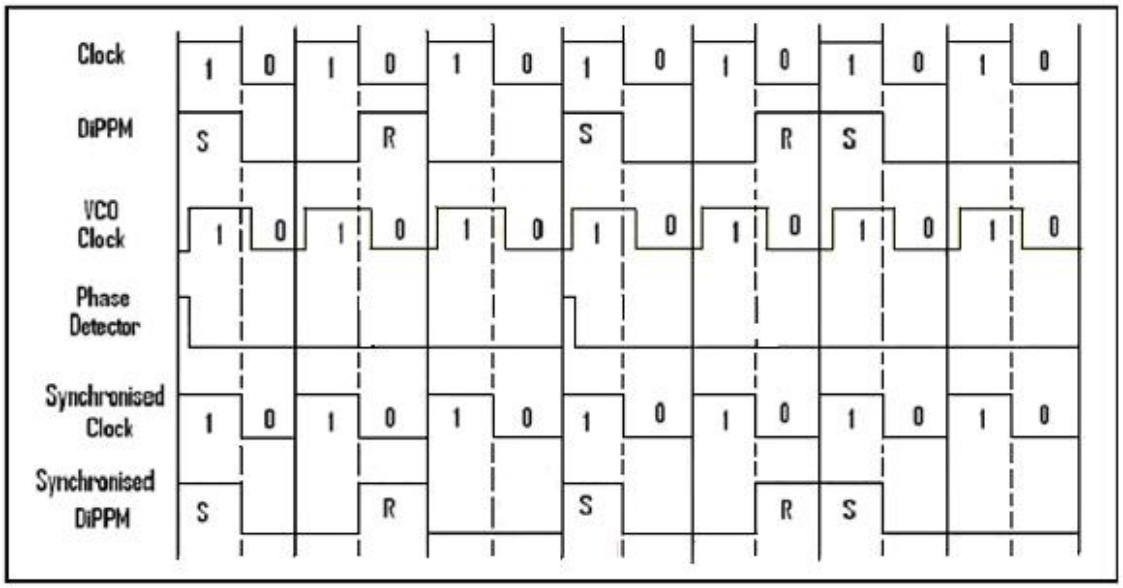


Fig 6.4: *DiPPM timing extraction waveforms.*

The error voltage was filtered by the *Loop Filter,* which removed any high frequency noise. Next, the output of the *Loop Filter* connected to the VCO to complete the loop.

The *error voltage* forced the frequency of the VCO to change reducing the frequency difference between the DiPPM and the VCO. As the VCO starts to change frequency, the loop is in the capture state. The process is continuous until the VCO and the DiPPM frequencies are identical. Hence, the DiPPM and the clock waveforms have been synchronised (phase-locked) (fig.6.4).

6.2 DiPPM Coder-Timing Extraction-Decoder Measurements

To verify achievement in clock recovery and synchronisation, the DiPPM coder (Appendix3-A), timing extraction and decoder (Appendix3-E) were connected through coaxial wire as is shown in figure 6.5.

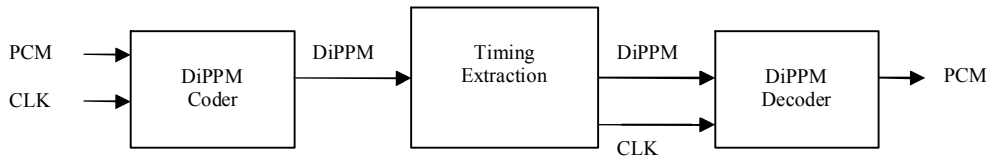


Fig 6.5: *Complete DiPPM System coaxes wire communication.*

When both outcomes of the DiPPM timing extraction circuit (DiPPM-CLK) are not synchronised, the measurement by the oscilloscope is as given in figure 6.6.

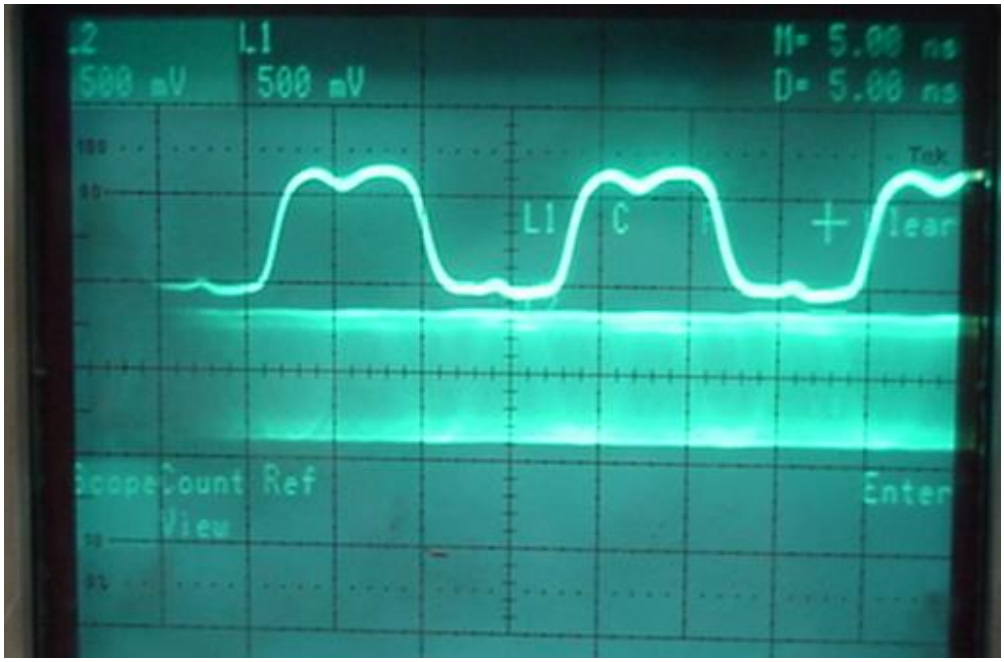


Fig 6.6: *Asynchronous DiPPM (top trace), clock (bottom trace) Timing Extraction's outcomes.*

With the use of the variable resistors of the AC coupling (changing resistance) (fig. 6.1) and of the *Loop Filter* synchronisation was achieved (fig. 6.7).

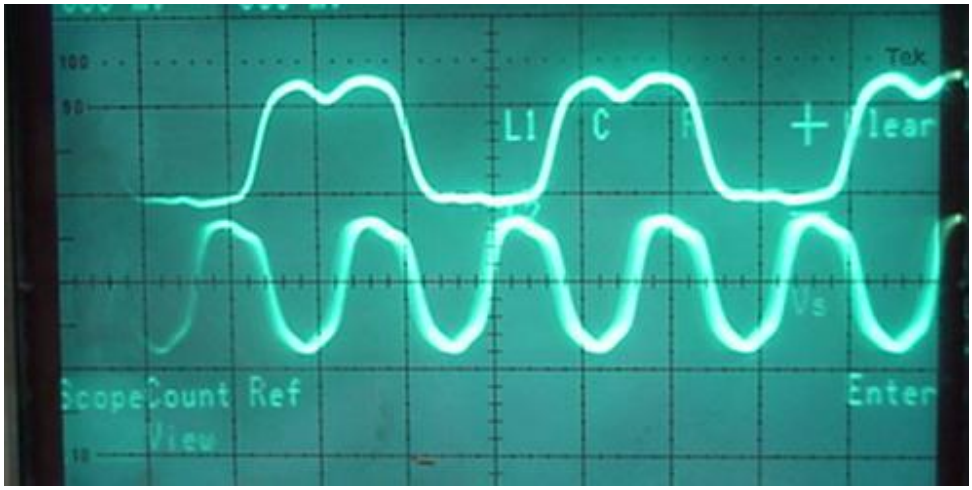


Fig 6.7: *Synchronous DiPPM (top trace), clock (bottom trace) Timing Extraction's outcomes.*

Operating the system with a PRBS PCM input signal, the synchronised outcomes (DiPPM-Clock) of the timing extraction circuit are presented in figure 6.8.

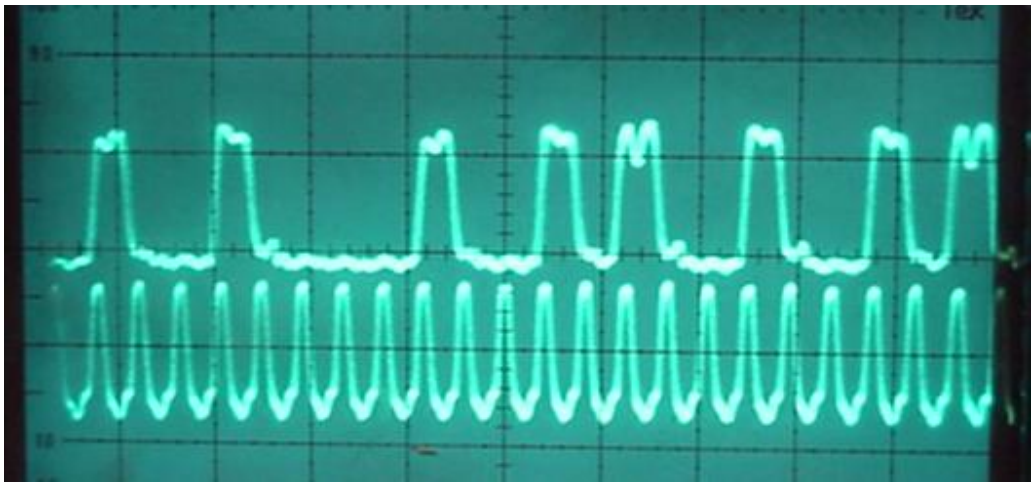


Fig 6.8: *Synchronous PRBS DiPPM (top trace), clock (bottom trace) Timing Extraction's outcomes.*

Hence, DiPPM clock recovery, slot and frame synchronisation were achieved and presented.

The spectrum of the Timing Extraction DiPPM signal was not anticipated to differ from that presented in figures 4.4 and 4.7. This is because the DiPPM waveform passed (buffered) through an OR gate (fig. 6.1) and the shape (width) of the DiPPM pulses was therefore not expected to change. In order to verify this, the PRBS DiPPM output of the timing extraction circuit passed through a double ECL OR gate (a component contains two OR gates, Appendix-F). The output of one OR gate was connected to the spectrum analyser, so the PSD measurement could be taken. The remaining OR gate output was connected to the oscilloscope, so that when the spectrum was measured, the DiPPM will be synchronised to the clock signal (fig. 6.7-6.8). As the waveforms (DiPPM-Clock) are synchronous, the PSD of the timing extraction PRBS DiPPM signal is:

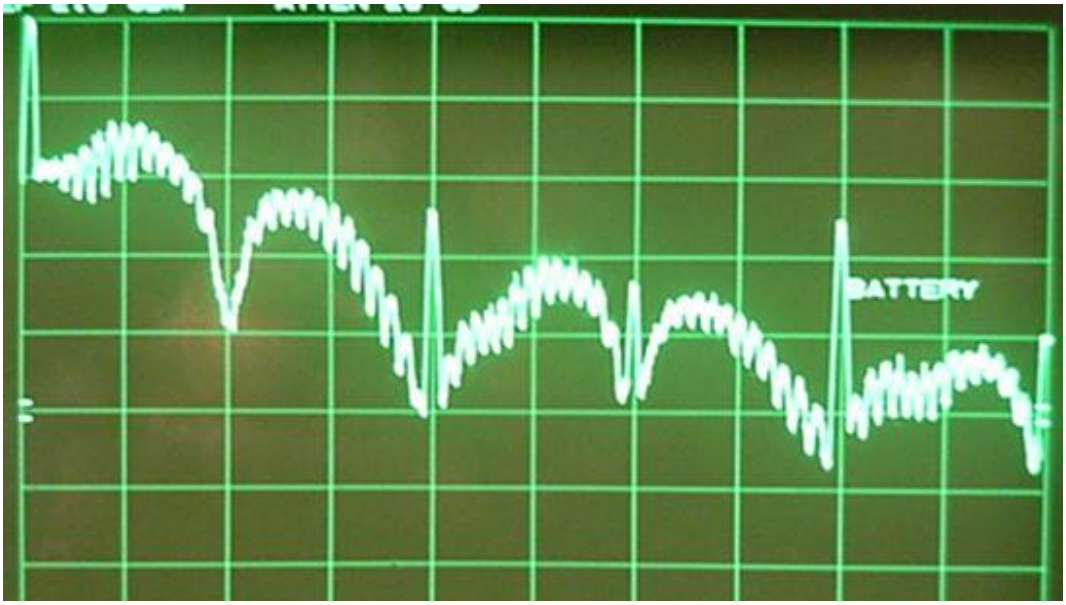


Fig 6.9: *Synchronous PRBS DiPPM spectrum as Timing Extraction's outcomes*

Through the comparison of figure 6.9 with that of figure 4.7, the theory and the correct operation of the DiPPM timing extraction circuit of this chapter have been proved.

DiPPM Timing Extraction was presented and proved to be accurate through measurement. PRBS DiPPM sequences were inserted into the DiPPM timing extraction component; the clock was recovered in all cases. The output (fig. 6.7-6.8) indicates the quality of the timing extraction as no jitters appear on the pulses. Hence, as a result of these investigations the DiPPM system (Coder-Timing Extraction and Decoder) is ready for use into further investigation of the DiPPM scheme through optics.

Chapter 7

Complete Dicode PPM using optical components

An optical Transmitter/Receiver system was built for the transmission of DiPPM sequences from the DiPPM coder to the DiPPM Timing Extraction through fibre (62.5/125 um, duple*8 type, tensile 300N, diameter of 2.8x5.7 mm, with GJFJBV-PVC and GJFJBZY-LSHZ jackets, of a 5.1 metres length). The end of the fibre was not directly connected to the optical receiver but was at a distance from the receiver photodiode, transmitting as free space communication (Appendix3-K). The pulse detector used slope detection as detailed in Chapter 3. Two Gaussian impulse reponse filters, with cut-offs of 240 MHz, were used in the transmit path and the receive path.

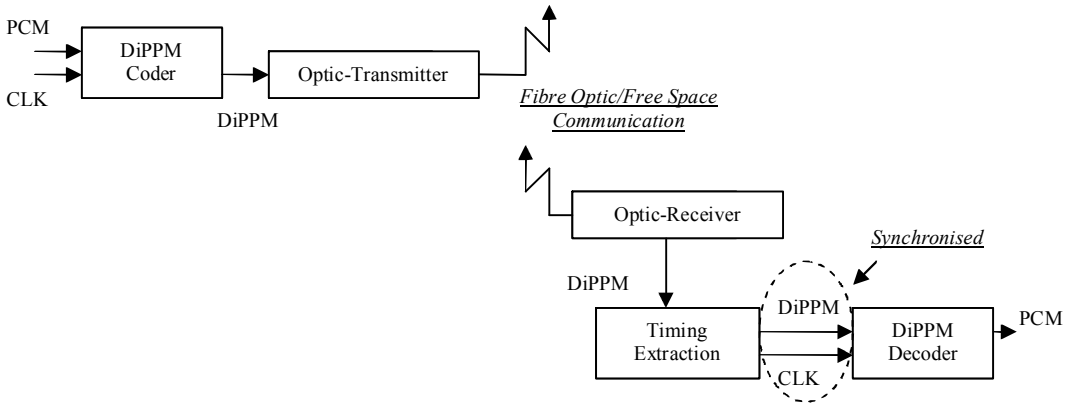


Fig 7.1: *Complete DiPPM system with transmitter/receiver for optic communication*

The optical transmitter/receiver system is composed of a Transmitter (threshold detector, simple-low pass filter and a VCSEL emitter transmitter (fig 7.2a)) and a Receiver (pre-amplifier, amplifier, simple low-pass filter and threshold-crossing detection (fig 7.2b)) [33].

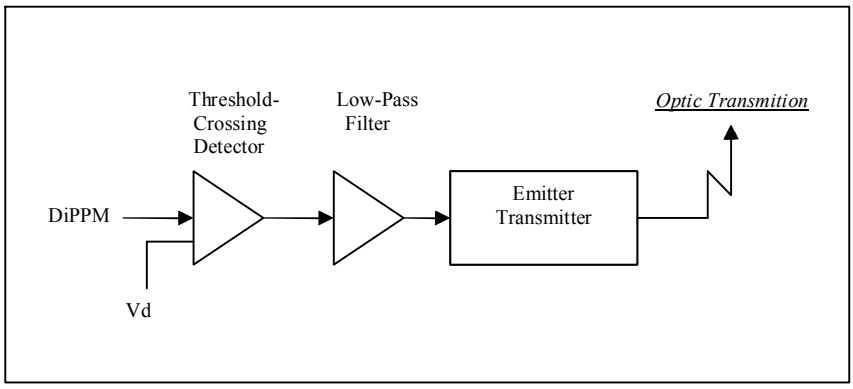


Fig 7.2a: Optical *Transmitting System*.

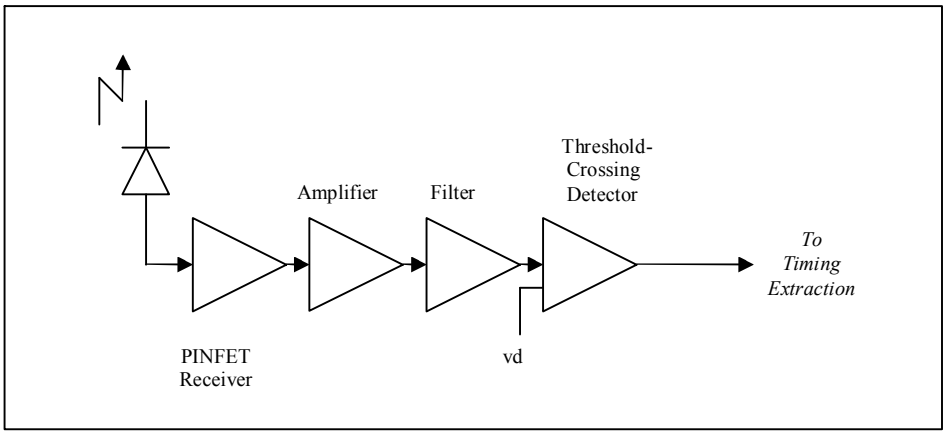


Fig 7.2b: *Optical Receiving System.*

7.1 Analysis of Optical Parts and Measurements.

Optical Transmitting System

Threshold-Crossing Detector

The threshold-crossing detector (Appendix3-G) used in the optical transmitter system is a comparator with one pin connected to a variable resistor (fig. 7.3).

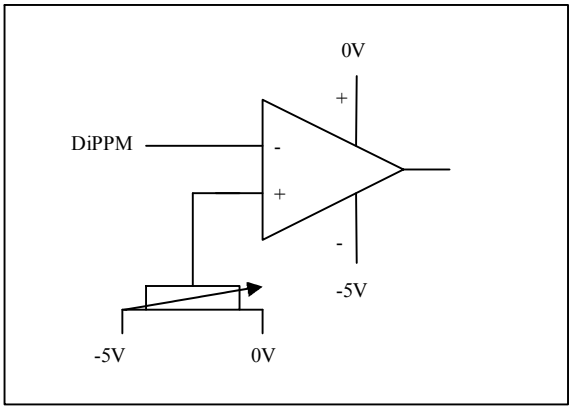


Fig 7.3: *Threshold-Crossing Detector.*

The use of the threshold-crossing detector and filter is not necessary in the optical transmitting system (fig.7.2a); the emitter transmitter alone can satisfactory transmit the waveform. The threshold-crossing detector (and the filter) were utilised in the optical transmitting system in order to remove noise interference. Noise was reduced (fig. 7.4) by passing the DiPPM signal, (which came from the DiPPM decoder), through the threshold detector with appropriate adjustment to the variable resistor (fig. 7.3).

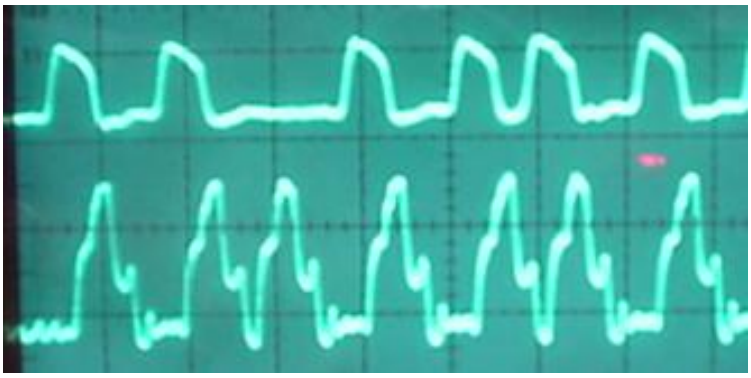


Fig 7.4: *Threshold-Crossing Detector output (top trace), DiPPM input (bottom trace).*

Low-pass Filter of Transmitter

The low pass filter (Appendix3-H) of the transmitter is a simple RC filter with a cut off frequency of 240 MHz. It was used to reduce noise from the DiPPM sequence greater than that of the signal frequency.

Emitter Transmitter

The emitter transmitter (Appendix3-J) is composed of a bias T and an 850nm VCSEL emitter. The output signal of the filter and current from a lightwave source were connected to the bias T. The bias T, had been connected to the 850nm VCSEL emmiter and the signal was transmitted (Appendix3-K) through fibre optics (or free space) to the optical receiver.

Optical Receiving System

Si PIN Pre-amplifier Receiver

The Si PIN pre-amplifier receiver (figure 7.5) (Appendix3-L) is composed of two buffers (Q1 and Q3) and an amplifying circuit (Q2).

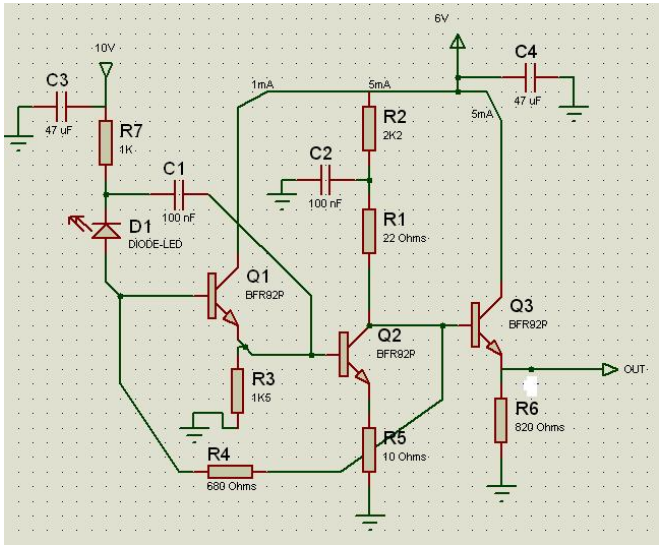


Fig. 7.5: *Si PIN pre-amplifier receiver.*

The bandwidth of the pre-amplifier is equal to

$$BdW = \frac{1}{2\pi \frac{R_4}{(1+A_v)}\left(C_d(1-A_1) + C_c + (1+A_v)C_f\right)} \tag{7.1}$$

where

$$A_v = -g_{m_2} R_1 \tag{7.2}$$

$$g_m = I_E \big/ 25mv \tag{7.3}$$

while A_1 is the input voltage at Q_1, C_d is the emitter-base capacitance, C_c is the collector emitter capacitance and the C_f is the capacitance at the feedback resistor R_4 [33]. The output of the Si PIN pre-amplifier receiver as compared to the output of the DiPPM coder is shown to figure 7.6.

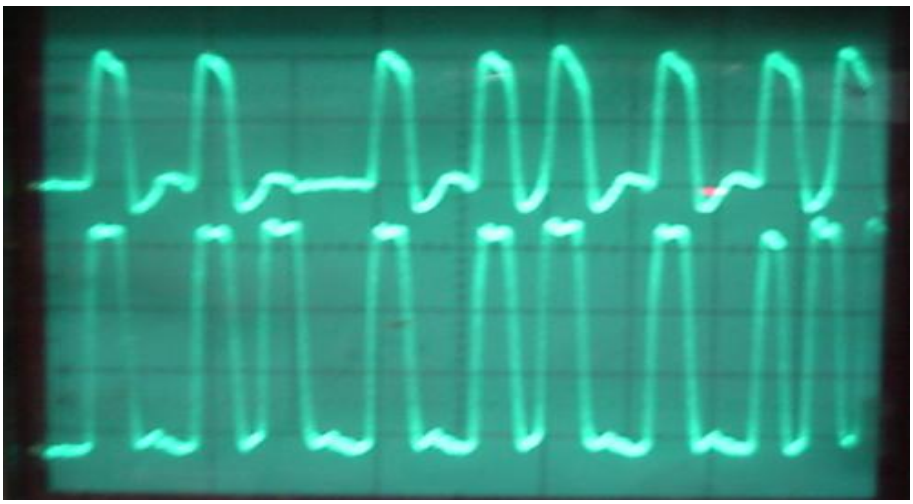


Fig. 7.6: *Output of pre-amplifier (top trace), DiPPM from coder (bottom trace).*

Optical communication was achieved as the DiPPM sequence collected from the receiver without any interference. However, the received DiPPM waveform amplitude was

insufficient (the waveforms of fig. 7.6, have the same amplitude as the amplitude settings at the oscilloscope are not the same) to be effectively detected by the timing extraction circuitry, or that of the DiPPM decoder. Thus, amplification was a necessary process.

Amplifier - Filter & Threshold-Crossing Detector

The output sequence of the active amplifier (Appendix3-M) was passed through the low-pass filter of the receiver (Appendix3-G) (with cut off frequency 240 MHz) in order to avoid any external interference higher than the cut off frequency. Next, it has been leaded through the threshold-crossing detector of the optical receiver. With correct adjustment, the threshold-crossing detector (Appendix3-N) gave the final DiPPM outcome of the optical transmitter/receiver, shown in figure 7.7.

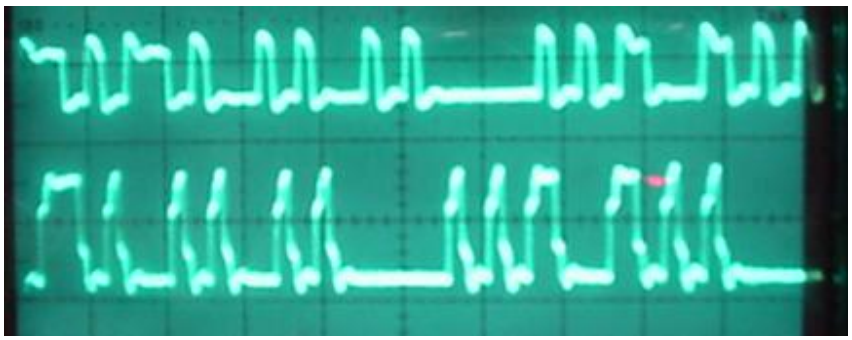


Fig. 7.7: *Output of threshold-crossing detector (top trace), DiPPM from coder (bottom trace).*

The DiPPM signal was transmitted through the optics and interfaced to the rest of the DiPPM system (Timing Extraction – Decoder), in order that the clock be recovered and synchronised with the DiPPM sequence.

7.2 Measurements of the Optical Received DiPPM Sequence

The DiPPM sequence was applied to the Timing Extraction circuit (chapter 6) and the following measurements taken of output:

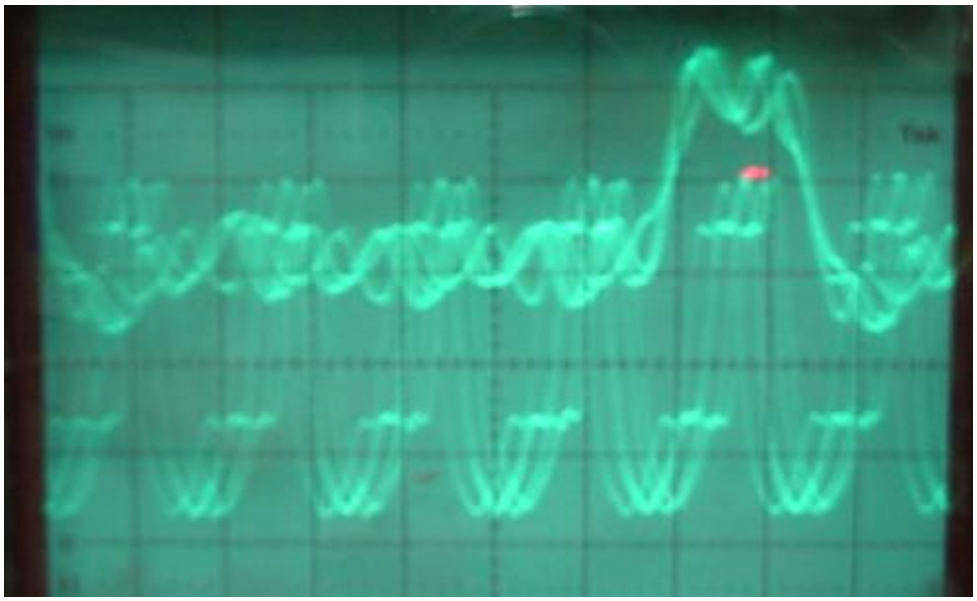


Fig. 7.8: *Asynchronous Outcomes of Timing Extraction: PRBS DiPPM (top trace), Clock (bottom trace).*

From figure 7.8, it appears that the clock was recovered but it was not synchronised. With the right adjustment of the AC coupling (fig.6.1), and at the loop filter (fig. 6.3), synchronisation of the clock to the DiPPM was achieved.

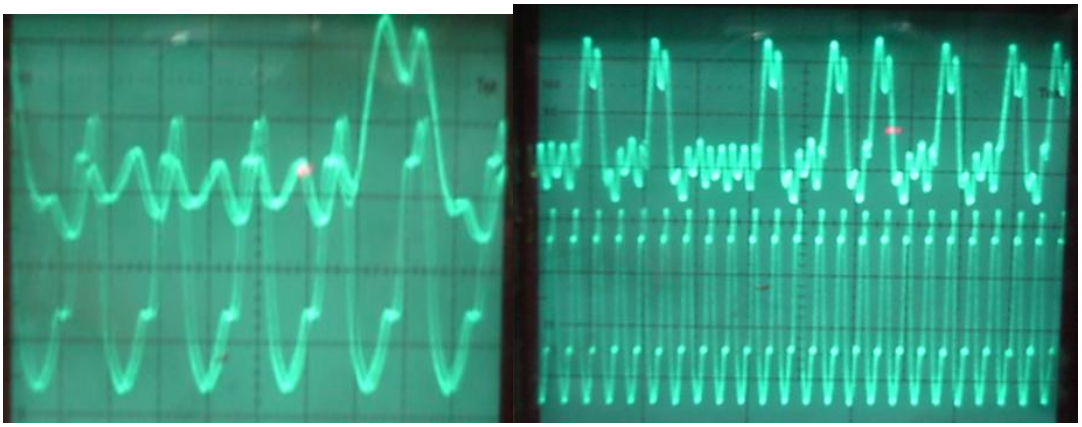


Fig. 7.9: *Synchronous Outcomes of Timing Extraction: PRBS DiPPM (top trace), Clock (bottom trace).*

By connecting the Timing Extraction's DiPPM outcome to the spectral analyser, the PSD of the DiPPM sequence was measured (using the same technique as contained in section 6.2). The power spectral density of the PRBS DiPPM was extracted and synchronised with the clock [33]:

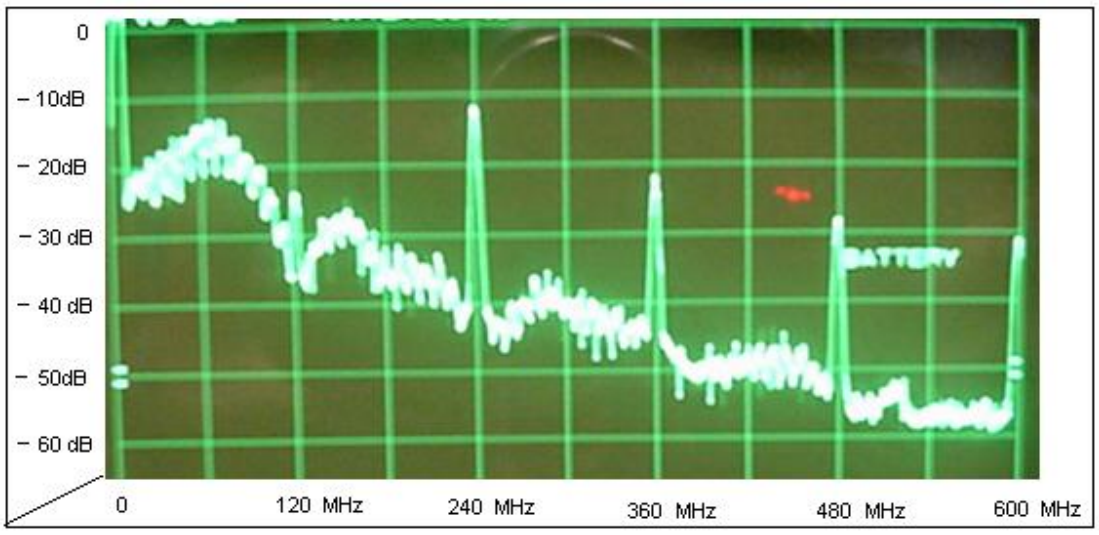


Fig. 7.10: *Optical DiPPM spectrum (synchronised DiPPM-Recovered clock).*

The DiPPM PSD of figure 7.10 does not correspond to the DiPPM PSD results (fig. 4.7, 5.13 and 6.9, [32] as the power of the fourth harmonic is different. This is due to the

width and the shape of the DiPPM pulses differing to those used in the previous analysis (fig. 4.7, 5.13 and 6.9) [32]. As described in section 5.3, a windowing equation (7.4) was constructed in order to simulate the shape and width of the experimental pulses; both PRBS PCM and DiPPM (see section 5.3) [33].

$$W_{opt.} = 15 - \frac{1}{0.9}\cos(2\pi ft) + \frac{1}{0.2}\cos(2\pi ft) - \frac{1}{0.7}\cos(6\pi ft) + \frac{1}{1.42}\cos(8\pi ft)dt \qquad (7.4)$$

Running the simulation code of Appendix-A1 and the equation 5.5 at Appendix-B2 code, both with windowing equation 7.4, the spectrum results are:

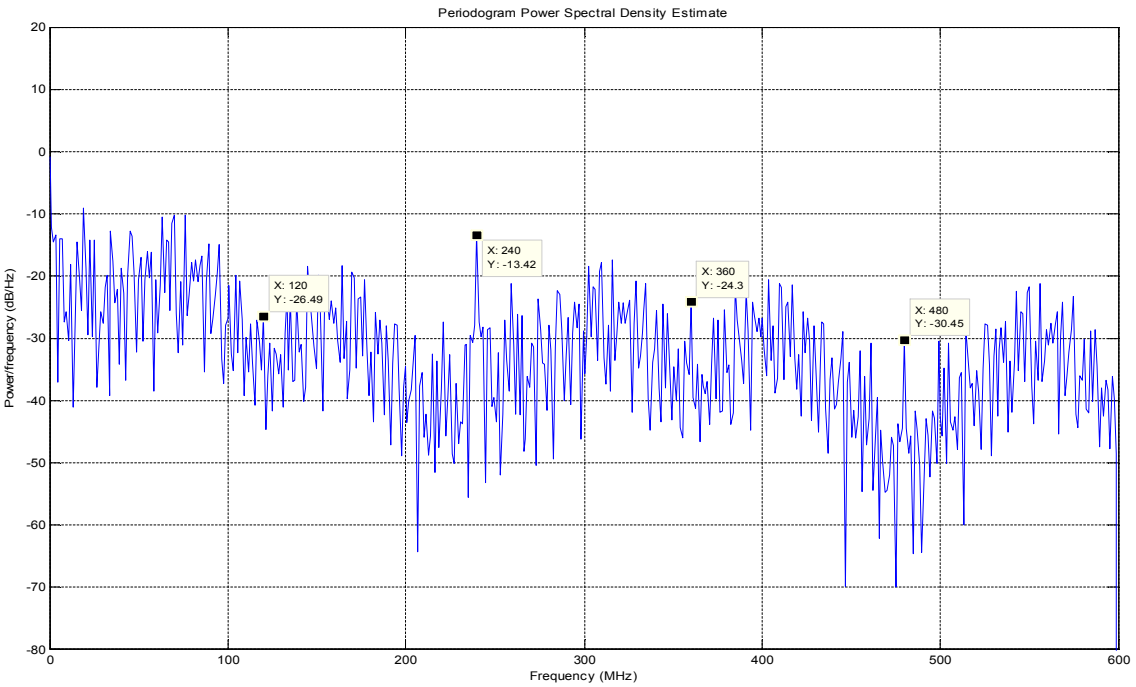


Fig. 7.11: *Optical DiPPM spectrum (simulated).*

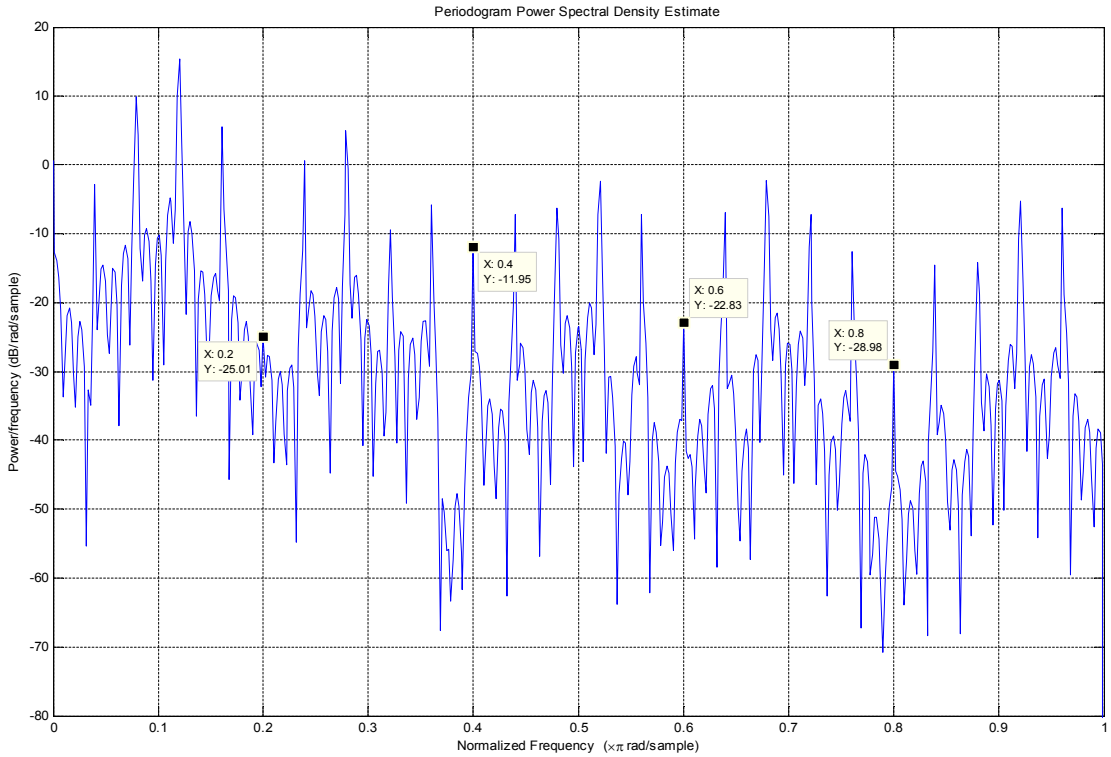


Fig. 7.12: *Optical DiPPM spectrum (from equation 5.5).*

Figures 7.10, 7.11 and 7.12 resemble each other and differences in the value of corresponding powers are negligible.

Complete optical transmitter/receiver system and additional components such as filters, threshold detectors and amplifiers were developed and presented. It has been shown that DiPPM can be transmitted over optics without loss of characteristics.

The complete DiPPM optical system (coder, optical system, timing extraction and decoder) has been presented and its correct operation has been. It has been also mentioned that delays, distortions, internal and external interferences have affected the DiPPM system and associated outputs. Hence, DiPPM should be examined in a more

stable situation where interferences and delays will be reduced or even to be eliminated. The use of FPGA can be a solution to those problems, thus DiPPM coder, decoder and timing extraction has to be programmed in VHDL.

Volume II of II

'Software'

Contents of Volume II

Chapter 8

VHDL: Dicode PPM Coder/Decoder & Clock Recovery

A complete optical DiPPM system was developed and analysed (Part I). Discussions on the outcome of measurements on the PPM scheme have also been included. Part II of this thesis will describe how to implement the format in more detail. The DiPPM system was programmed into a Field-Programmable Gate Array (FPGA). FPGAs were programmed using a source code in a Very high Speed Integrated Circuits-(VHSIC) Hardware Description Language (VHDL), which specifies how the device functions. The reason for utilising the FPGA is that the resultant system will have lower levels of external noise and any internal delays than in the previous implementations.

The basic versions of DiPPM coder and decoder will be presented and discussed in this chapter.

8.1 DiPPM Coder/Decoder Simulation Version

8.1.1 DiPPM Coder

The first VHDL coding version of DiPPM coder appears in Appendix1-C1 and was programmed in Active-HDL (demo). This version simulates the components employed in the experimental DiPPM coder in chapter 4 (fig. 4.1).

Line 9 of Appendix1-C1 DiPPM coder simulation, contains the construction (*entity)* of the coder. The architecture of the DiPPM coder, which is the input and output address of the components that have been used (D flip-flop, NOR, OR/NOR gates), is presented in line 10. The mapping of component input and output pins has been set in lines 12 to 19, while the signals that connect them have been named in line 11.

Running the DiPPM coder simulation with clock and deterministic PCM input waveforms, the simulation output is as follows:

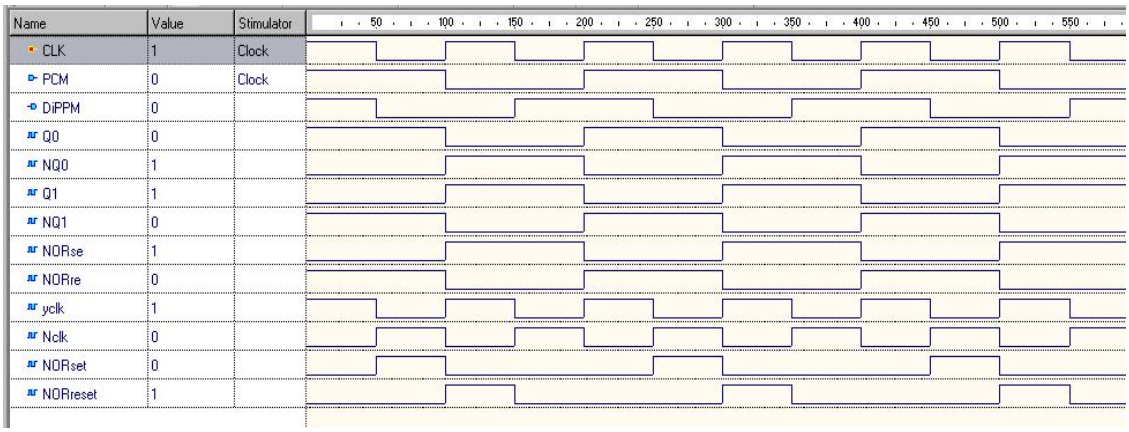


Fig. 8.1: *VHDL: Deterministic DiPPM coder simulation*

Figure 8.1 shows the process of the DiPPM coder VHDL software. The output waveforms correlate with theory. The PCM waveform has been correctly coded to DiPPM format. Hence, the simulation of Appendix1-C1 is correct. Thus, running the code using an PRBS PCM waveform input signal, it is expected to give the correct DiPPM coded outcome:

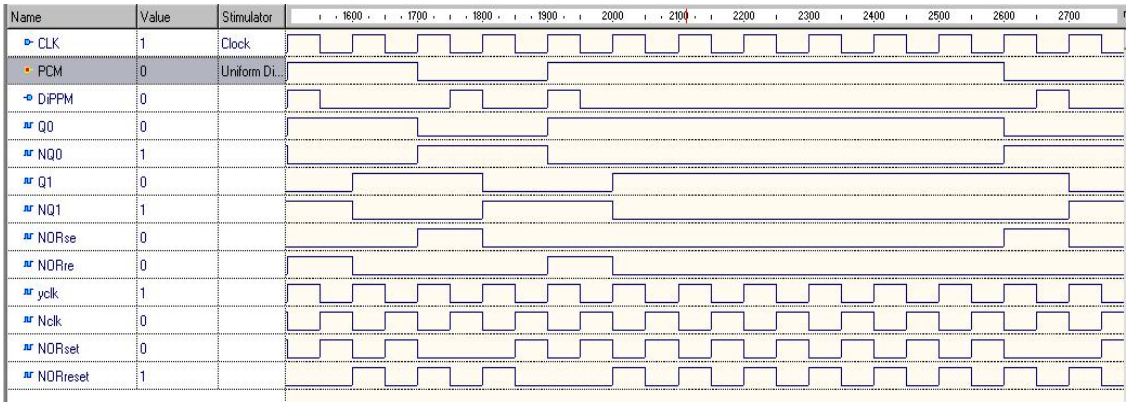


Fig. 8.2: *VHDL: PRBS DiPPM coder simulation*

Figure 8.2 illustrates the correct function of the DiPPM coder (Appendix1-C1) through the correct simulation results. The output waveforms also agree with the theoretical simulation of figure 4.2 from chapter 4.

8.1.2 DiPPM Decoder

The VHDL simulation of the DiPPM decoder appears in Appendix1-C2. In lines 1 to 4, the structure and architecture of the components used in DiPPM decoder simulation are described. The structure of the DiPPM decoder is presented in line 5. The input-output pins of the DiPPM decoder components are contained in line 6 and have been connected (lines 8-13) through signals (line 7) to produce the process of the DiPPM decoder.

As the Active-HDL cannot produce a DiPPM waveform, the DiPPM coder VHDL simulation has been used to produce the DiPPM input waveform of the DiPPM decoder as figure 8.3 shows:

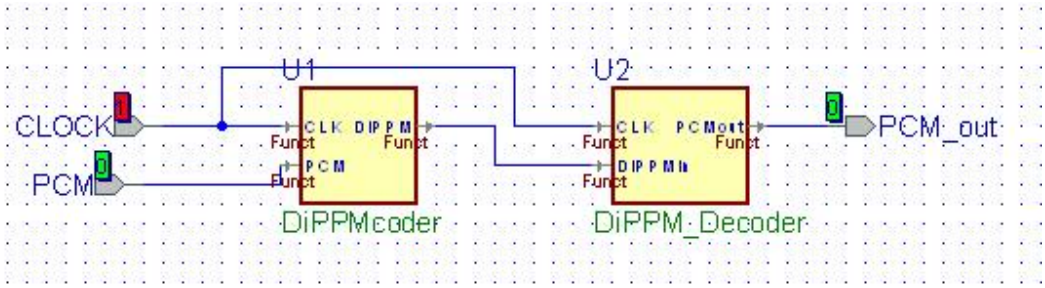


Fig. 8.3: *DiPPM Coder-Decoder*

Simulating the DiPPM Coder-Decoder system, as shown in the schematic of figure 8.3, with a deterministic PCM input signal, the output waveforms are:

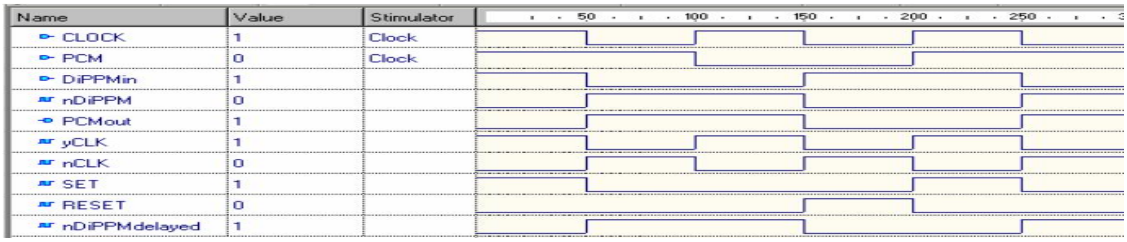


Fig. 8.4: *VHDL: Deterministic DiPPM decoder simulation*

From figure 8.4 it can be ascertained that DiPPM decoder VHDL simulation of Apendix1-C2 is correct, as resulting waveforms correlate with theory. The output waveforms for the same DiPPM system with input PRBS PCM sequence are shown below:

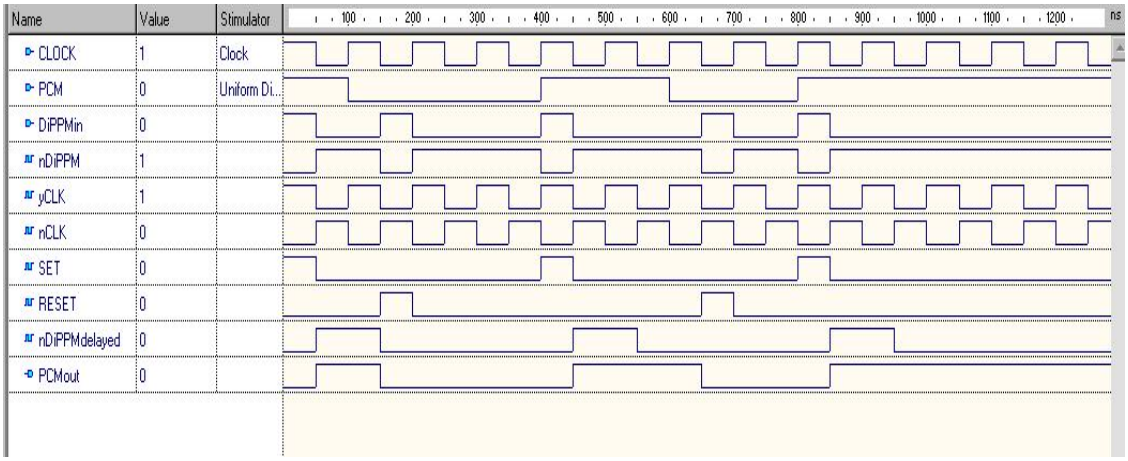


Fig. 8.5: *VHDL: PRBS DiPPM decoder simulation*

The outcomes of DiPPM decoder appear as were expected, when the input signal of the DiPPM system (DiPPM coder-decoder) is a PRBS PCM sequence (fig.8.5). Hence, the DiPPM coder and decoder (chapter 4) can be simulated in VHDL.

8.2 DiPPM Coder/Decoder Upgraded Version

8.2.1 DiPPM Coder

An upgraded version of the DiPPM coder is presented in Appendix1-C3. The DiPPM coding structure (*entity*) is presented in line 3 while architecture is contained in line 4. This version of the DiPPM coder (Appendix1-C3) generates the DiPPM SET pulse when the PCM and clock are '1' and the DiPPM RESET pulse is generated when the PCM and clock are '0'. For the other cases, the system generates no pulses ('0'). Line 4 is adding the SET and RESET sequences and producing the DiPPM waveform.

As the VHDL software requires that the input signals are synchronised, a D flip-flop is used to synchronise the input PCM signal with that of the clock (line 2). Line 5 contains

the input and output of the components while line 6 contains the track (inside signal) that connects them. Finally, at lines 7 and 8 the circuit is shown to be complete. Running the upgraded DiPPM coder VHDL software, the waveforms are as follows:

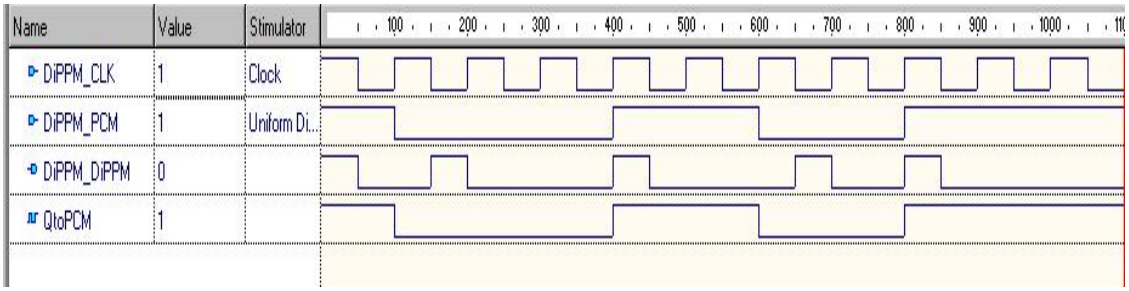


Fig. 8.6: *VHDL: PRBS Upgraded DiPPM coder - D flip-flop included*

The DiPPM coding appears to be correct in figure 8.6 with the *'QtoPCM'* signal output being synchronised with the input signals. Without this synchronisation, the RESET pulse will not appear in the correct position. In the upgraded DiPPM coder a set delay is used to replace the D Flip-flop (Appendix1-C4, line 3).

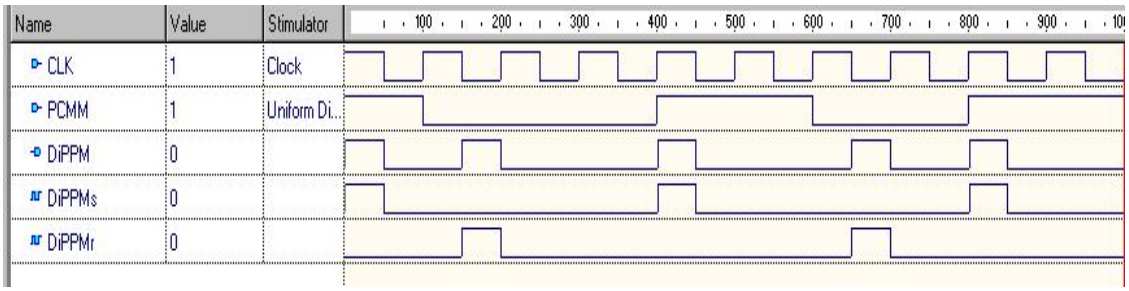


Fig. 8.7: *VHDL: PRBS Upgraded DiPPM coder - delay included*

As can be seen in Appendix1-C4, the DiPPM coder software is composed of the Entity (line 1) and the Architecture (line 2) sections. In the architecture section, the internal

signals have been set, but now in processes form. Architecture also includes the delay (line 3) of 2 ns which is delays the PCM input sequence so the clock signal will appear first. Hence the SET pulse '1' have been set at the beginning of every PCM '1' and the RESET pulse'1' at the beginning of every PCM '1' but when the clock is '0'. When these two sequences (SET-RESET) are added, they produce a DiPPM waveform without the use of D flip-flop for the synchronisation.

8.2.2 DiPPM Decoder

An upgraded version of DiPPM decoder is presented in (Appendix1-C5). Simpler than the DiPPM decoder VHDL simulation version, the new version is composed of two parts; the the entity (line 1) and architecture (line 2). The DiPPM decoder sets '1'at the output when the input DiPPM sequence and clock are '1'. It also resets '0' while the DiPPM is '1' and the clock is '0'. Unfortunately, correct PCM sequence cannot be produced as the DiPPM decoder had to reset '0' a half clock earlier. The delay command has been used once again. In line 3 the software delays the SET by half a clock period; hence the correct PCM will be appears as output.

For the DiPPM decoder simulation the upgraded version of DiPPM coder was used to produce the input DiPPM signal of the decoder:

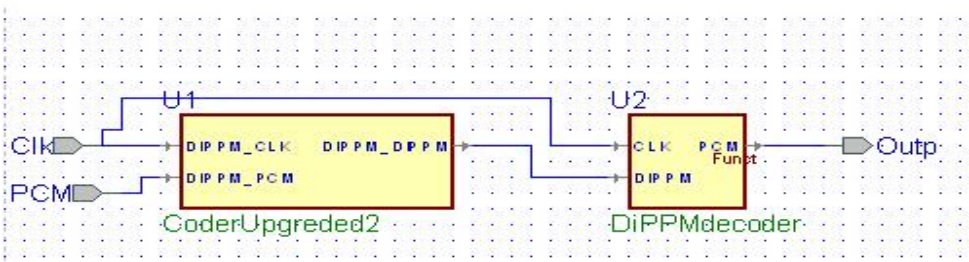


Fig. 8.8: *DiPPM upgraded versions of coder-decoder*

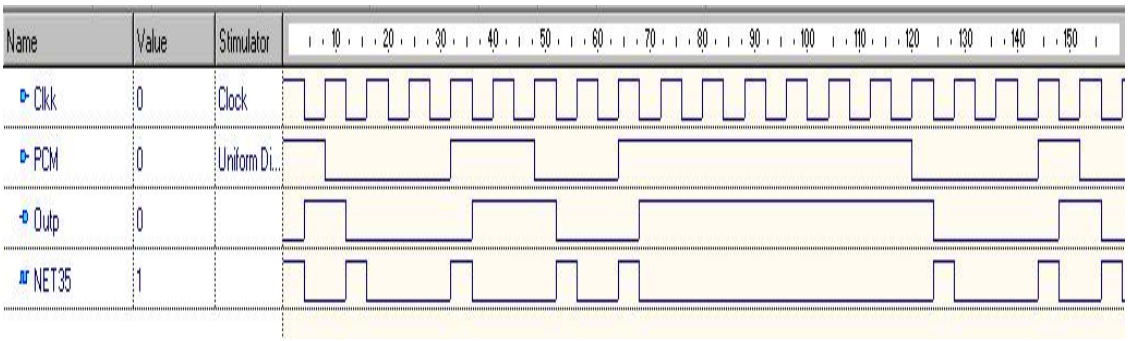


Fig. 8.9: *VHDL: PRBS Upgraded DiPPM decoder - delay included*

8.3 DiPPM Coder/Decoder Real-Time Measurements

For the real-time process of the DiPPM coder-decoder an FPGA was used, programmed by software from ALTERA Company. Unfortunately, the ALTERA software (Quartus) does not accept D flip-flop or any other simulated logic component as presented at the simulation of the DiPPM system (section 8.1). Thus, previous codes required conversion inorder to be compatible with the Quartus software.

Appendix1-C6 contains the DiPPM coder VHDL version for Quartus software; the entity of the coder is at line 1 and the architecture at line 2. The PCM has been moved half a clock forward (line 3) with the addition of the PCM sequence to line 3. Thus, when'10' is 1 and all the other pairs are equal to '0', the SET sequence appears (line 4). As the PCM has been reversed (line 5), the same process has been used (line 6-7) on the sequence of line 5 as that of SET (line 3-4). Adding the SET (line 4) and the RESET (line 8) gives the DiPPM sequence (line 9).

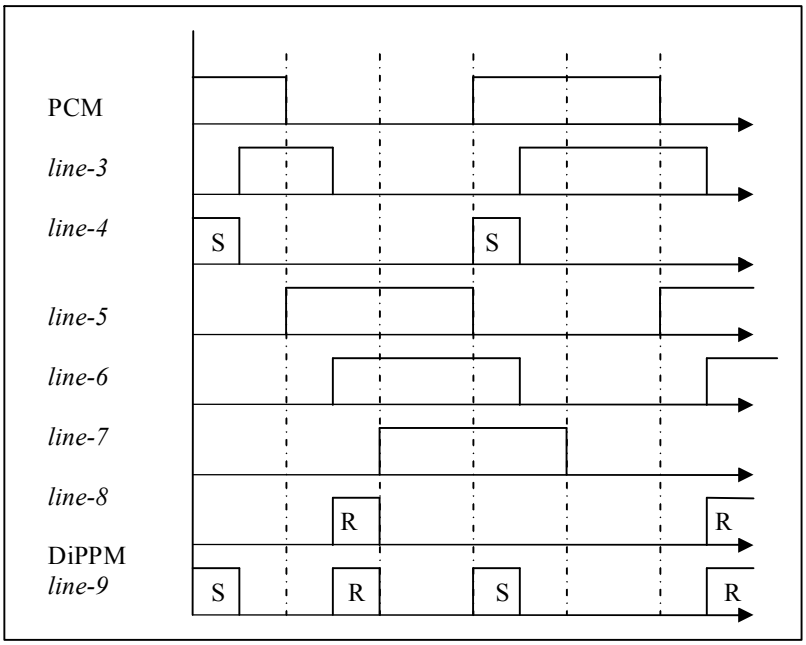


Fig. 8.10: *VHDL: DiPPM coder process in Quartus*

In line 3, *nclk* is set as invert clock (not clk) and is used in line 4 with the input DiPPM sequence. Positive pulse has to be produced when the DiPPM SET pulses appear and the *nclk* is equal to '1'. Thus, positive PCM is achieved with a half clock delay. While a RESET pulse appears the positive PCM false until a SET DiPPM pulse appears again.

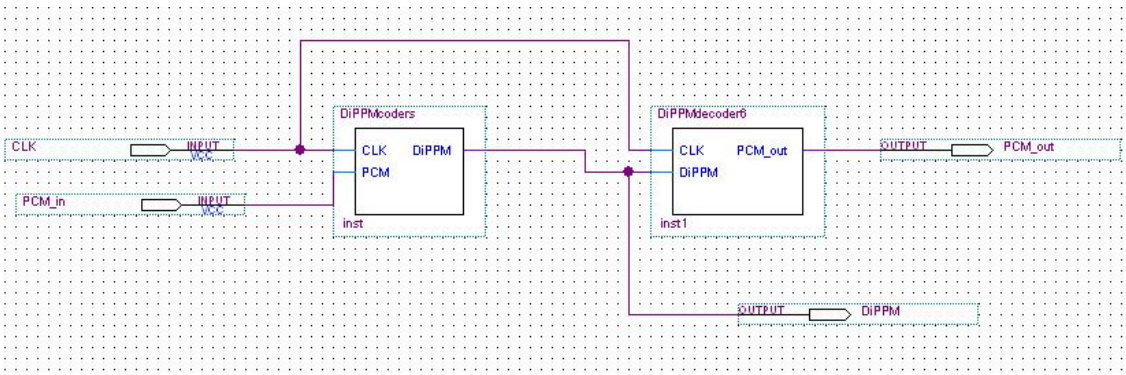


Fig. 8.11: *VHDL: DiPPM coder-decoder in Quartus*

Running the simulation DiPPM system (fig.8.11) in Quartus, the outcomes are:

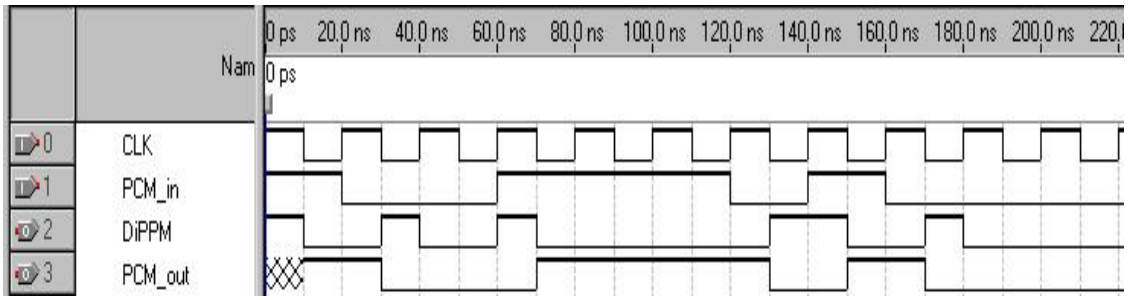


Fig. 8.12: *VHDL: DiPPM coder-decoder outcomes*

Figure 8.12 shows that both (coder-decoder) simulations produce correct outcomes. The PCM output sequence is delayed by half a clock as expected. Hence, the software (Appendix1-C6 and C7) was installed into the FPGA component for further measurements of outcome.

Figure 8.13 shows the PRBS DiPPM sequence produced by VHDL software throughout FPGA, compared with the input PRBS PCM sequence of the system.

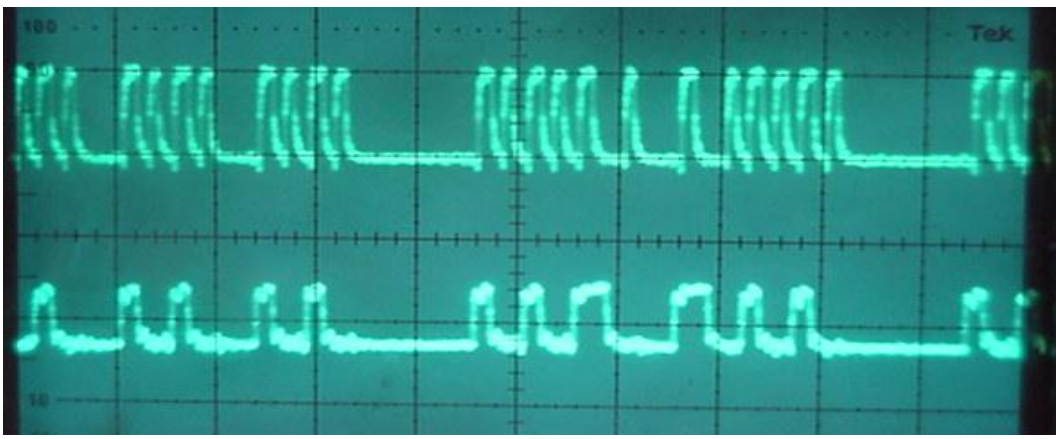


Fig. 8.13: FPGA: *PRBS DiPPM output sequence (top trace), PRBS PCM input sequence (bottom trace)*

The output pin of the DiPPM coder was connected with the input pin of the DiPPM decoder and the measurement taken.

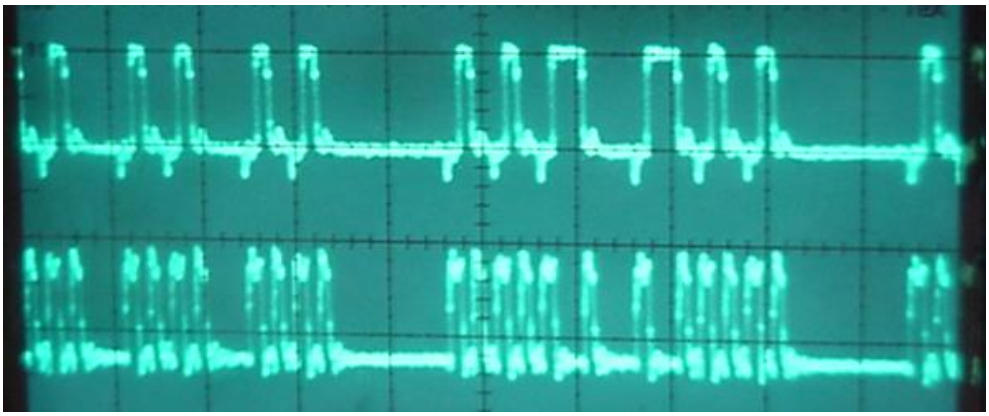


Fig. 8.14: FPGA: *PRBS PCM output sequence (top trace), PRBS DiPPM input sequence (bottom trace)*

Finally, both PCM sequences (input and output) were measured and comparisons made.

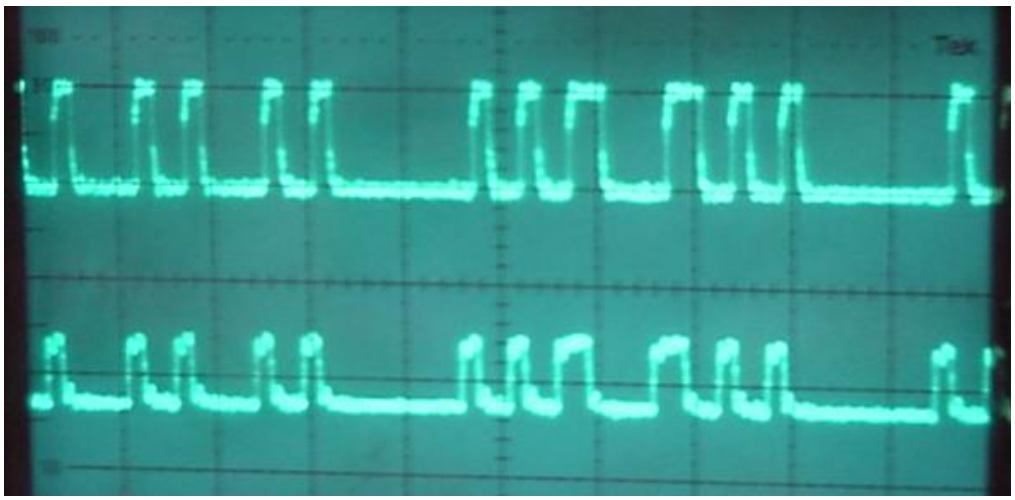


Fig. 8.15: FPGA: *PRBS PCM output sequence (top trace), PRBS PCM input sequence (bottom trace)*

Figures 8.13, 8.14 and 8.15 show that the software (Appendix1-C6 and C7) produces the correct DiPPM through FPGA and correctly decodes it back to PCM form.

For completion of the DiPPM system (coder-Timing Extraction-decoder), software was required to simulate timing extraction of the DiPPM format. But, the use of the VCO in the timing extraction program is necessary. As VCO components need analogue inputs and VHDL software can not achieve such analogue process. Thus another software language had to be used, which does accept analogue elements, but common with VHDL.

8.4 VHDL-AMS: DiPPM Timing Extraction

VHDL-AMS (VHDL-Analogue Mixed Signal) is the software language which accepts digital and analogue signals. It also converts digital input signals to analogue form and vice versa. FPGA components accept VHDL-AMS and VHDL language. With the use of this VHDL-AMS language the PLL circuit can be achieved, hence the DiPPM timing extraction circuit (fig. 6.1) can be simulated.

DiPPM timing extraction software is composed of five sub-programs: the buffer, the phase detector, the loop filter, the VCO which produces the clock in sinewave form and the digitaliser which is an analogue to digital converter (ADC).

8.4.1 Buffer

The DiPPM coded sequence was inserted into the buffer software (Appendix1-D1) to produce two DiPPM sequences exactly the same as the input sequence (fig. 8.16). The DiPPM sequence was connected to the DiPPM decoder software and the output sequence

of the buffer software was connected to the PLL software, producing the synchronised clock waveform.

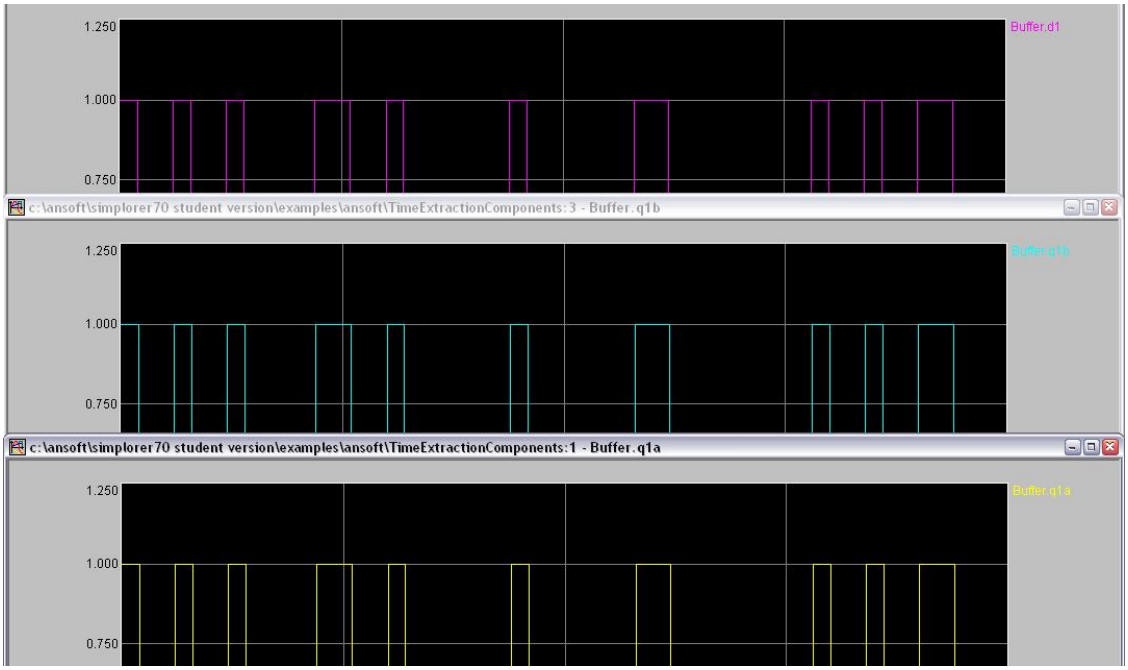


Fig. 8.16*: PRBS DiPPM input (top trace), PRBS DiPPM output inserted to decoder (middle trace), PRBS DiPPM output inserted to PLL circuit (bottom trace).*

8.4.2 Phase Detector

The phase detector (Appendix1-D2) is the first in the row of components which compose the PLL software circuit. The phase detector is programmed in VHDL, as well as the buffer software, as both input and output signals are digital.

The phase detector software compares the phases of two inputs and reproduces the DiPPM to the same respective edges:

```
Q <= '0' when  Vin='0'and VRef='0' else
     '0' when  Vin='0'and VRef='1' else
     '1' when  Vin='1'and VRef='1' else
     '1' when  Vin='1'and VRef='0' else
'0';
```

Fig. 8.17*: Phase detector software process*

Vin is the phase detector input DiPPM sequence received from the buffer. The *VRef* signal is the extracted clock signal received from the VCO. From the comparison of these two waveforms, the edges have been synchronised.

When the phase detector receives the incoming DiPPM sequence and no signal has been received at the *Vref* pin, the output waveform is the same as that of the input. However, when the clock signal is received by the phase detector (pin *VRef*) and the clock is not synchronised with the DiPPM sequence, which has been received at the *Vin* pin, the output sequence of the phase detector gives the difference between the asynchronised input sequences (fig. 8.18).

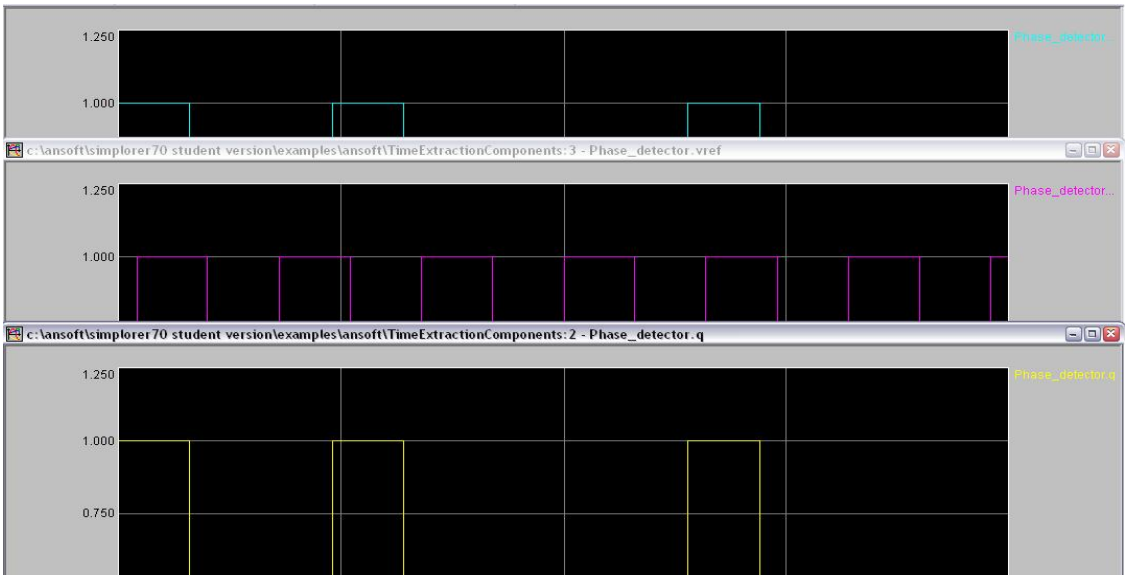


Fig. 8.18*: Phase detector: Output sequence (top trace), input clock VRef (middle trace), Input Vin sequence (bottom trace).*

The phase detector signal, as given in figure 8.18, provides the pulse difference between the DiPPM pulse and the delay of the clock sequence. It also produces the remaining DiPPM sequence and the DiPPM outcome is the same as the DiPPM sequence received by the pin *Vin* of the phase detector, due the following command:

$$'1' \text{ when } Vin='1' \text{and } VRef='1' \text{ else} \qquad (8a)$$

which exists in phase detector's software (fig. 8.17).

If the command changes to:

$$'0' \text{ when } Vin='1' \text{and } VRef='1' \text{ else} \qquad (8b)$$

The phase detector will produce signal edges, as that of the experimental phase detector (figure 6.2), and this will not function correctly. As can be seen in figure 8.19, the DiPPM RESET (01) causes errors with the use of the command 8b. Edges are produced at SETs but the edges at RESET are asynchronous.

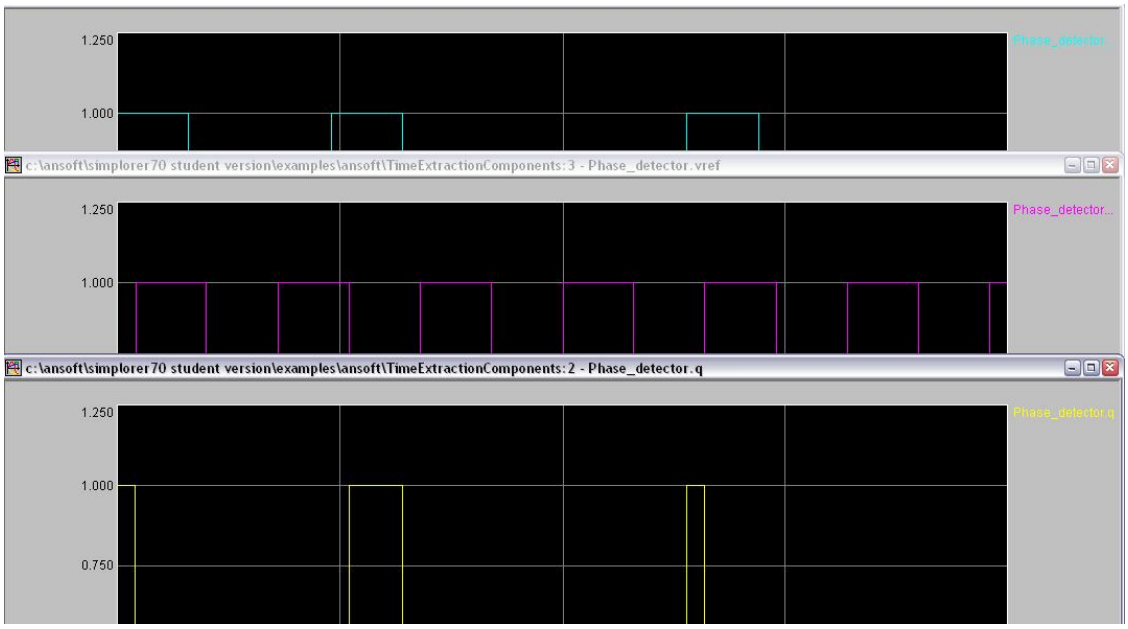


Fig. 8.19*: Phase detector (faulty): Input Vin sequence (top trace), input clock VRef (middle trace), Output sequence (bottom trace).*

If the software given in figure 8.17 changes to the following:

```
Q <= '0' when  Vin='0'and VRef='0' else
    '1' when  Vin='0'and VRef='1' else
    '0' when  Vin='1'and VRef='1' else
    '1' when  Vin='1'and VRef='0' else
    '0';
```

Fig. 8.20*: Phase detector software process (faulty)*

It is anticipated that the signal edges (rising and falling points of clock) will appear when the DiPPM sequence is ‘1’ and the asynchronous clock sequence is ‘0’. Additionally when the DiPPM is ‘0’ and the clock is ‘1’. This is still not a suitable sequence for the PLL as can be seen in figure 8.21.

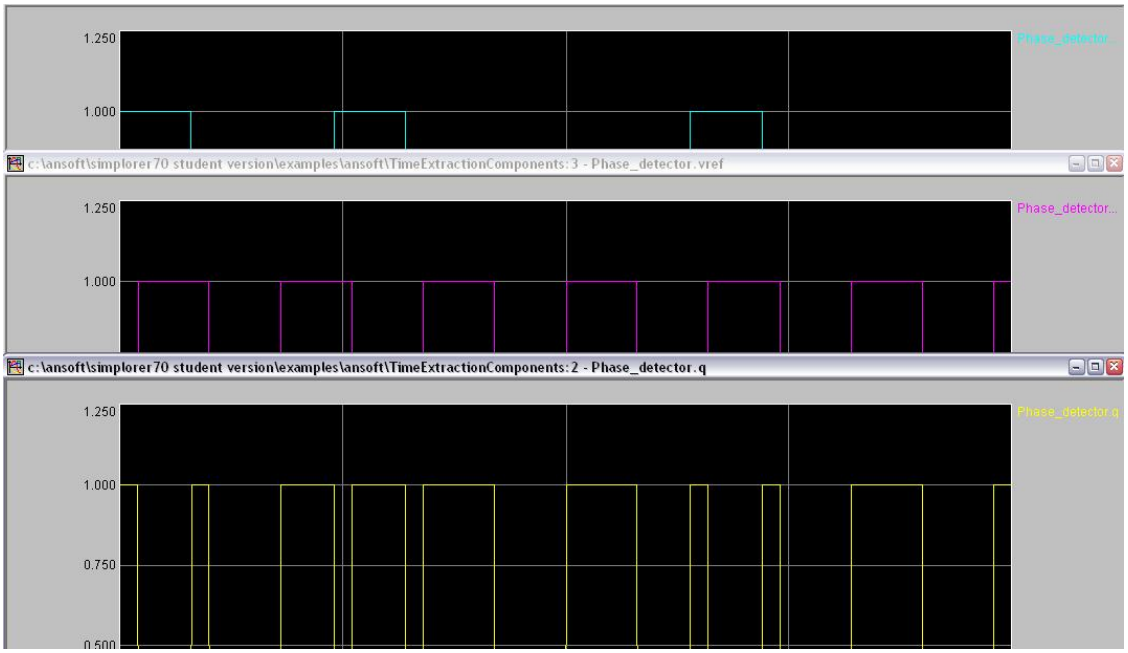


Fig. 8.21*: Phase detector (faulty): Output sequence (top trace), Input Vin sequence (middle trace), input clock VRef (bottom trace).*

The anticipated signal edges appear when a SET exists at the DiPPM sequence (fig. 8.21-first two pulses at the bottom trace). However, when RESET appears in the DiPPM sequence, the outcome of the phase detector is not compatible with the requirements of the PLL. Hence, figure 8.17 demonstrates a simulation of the experimental phase detector; driving the PLL circuit.

8.4.3 Loop Filter

The loop filter could not be simulated by VHDL. VHDL-AMS software language was therefore utilised, as the loop filter is a 'mixed' (digital input-analogue output) component.

As VHDL-AMS language can simulate resistors, capacitors and inductors, a low pass filter was incorporated. VHDL-AMS language can also accept mathematics; thus, the Laplace Transfer Function (LTF) can be used for the loop filter simulation. Hence, each individual component of the filter can be transformed into the s-domain and the respective interaction can be determined.

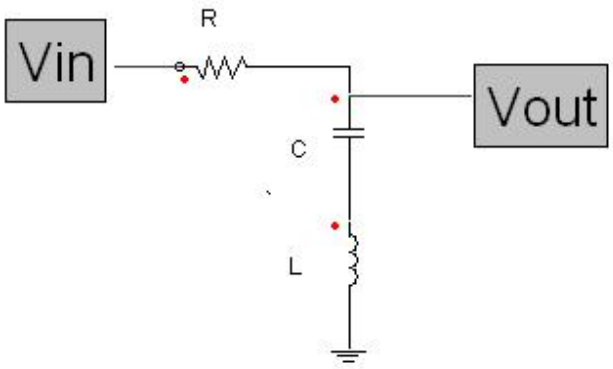


Fig. 8.22*: Loop Filter (VHDL-AMS)*

Figure 8.22 shows the *loop filter* scheme that has been programmed in VHDL-AMS language. The system's transfer function is equal to the s-domain representation of the output signal, divided by the s-domain representation of the input signal.

$$H(s) = \frac{sL + 1/sC}{R + sL + 1/sC} = \frac{sL + 1/sC}{R + sL + 1/sC}\left[\frac{s}{s}\right] = \frac{Ls^2 + 1/C}{Ls^2 + Rs + 1/C} \qquad (8.1)$$

The VHDL-AMS software of the loop filter appears in Appendix1-D3. In addition to the digital library (std_logic_1164.all), two further libraries have been used; 'math_real.all'

which enables the software to accept mathematical equations and 'electrical _systems.all' which enables the acceptance or production of analogue signals. The Entity section of the software is configured so that input and output are set to accept and produce both digital and analogue signals. The numerator (C, L) and the denominator (C, L, R) of the s-domain have been set in the Architecture section. Finally, an equation calls for the *Vout* result of the filter which is equal to *Vin* multiplied with the Laplace Transfer Function of equation 8.1.

Given a PRBS DiPPM sequence input signal the compiled software from Appendix1-D3 outputs a waveform as:

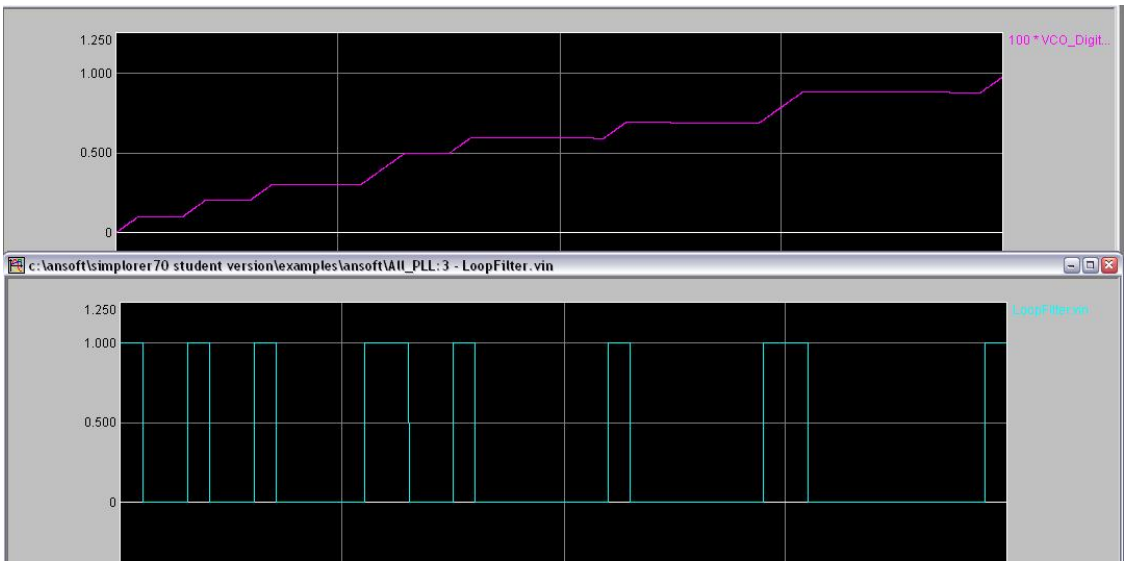


Fig. 8.23*: Loop filter's output (top trace) when its input signal is PRBS DiPPM (bottom trace).*

8.4.4 VCO

VCO is an analogue (analogue input/output) software component designed to receive the *Vout* of filter (analogue) and produce a deterministic sinewave sequence (analogue). The VCO output can be written as:

$$VCO(t) = V_0 \cos\left(2\pi \int_0^t \left(f_c + \kappa VCO\left(U_{in} - U_c\right)\right)dt + \varphi(t) \right) \tag{8.2}$$

where V_0 is the maximum output amplitude, kVCO is the simulated VCO gain sensitivity (Hz/V), f_c and U_c are the frequency and the voltage of the central point of the tuning range and U_{in} is the control voltage. But for VHDL-AMS the equation 8.2 could be written in a simpler form, as:

$$f_{VCO}(t) = f_c + k_v V_{lf}(t) \tag{8.3}$$

where f_c is the VCO centre frequency (Hz), k_v is the VCO gain (HZ/V) and V_{lf} is the output of the loop filter. The VHDL-AMS program of VCO appears at Appendix1-D4. As the loop filter software, the VCO has the same libraries, while using mathematics and analogue signals. In the Entity section the centre frequency and the gain are set and the inputs and outputs of the system could either receive or produce analogue signals. In the Architecture section of the VCO, the φ (phi) is set, whilst the gain and the centre frequency have changed to (rad/s)V and rad/s respectively. In the process of the VCO the

φ (phi) has been initialized and the VCO frequency has been calculated. Figure 8.24 shows the output of the VCO compared with the input DiPPM sequence.

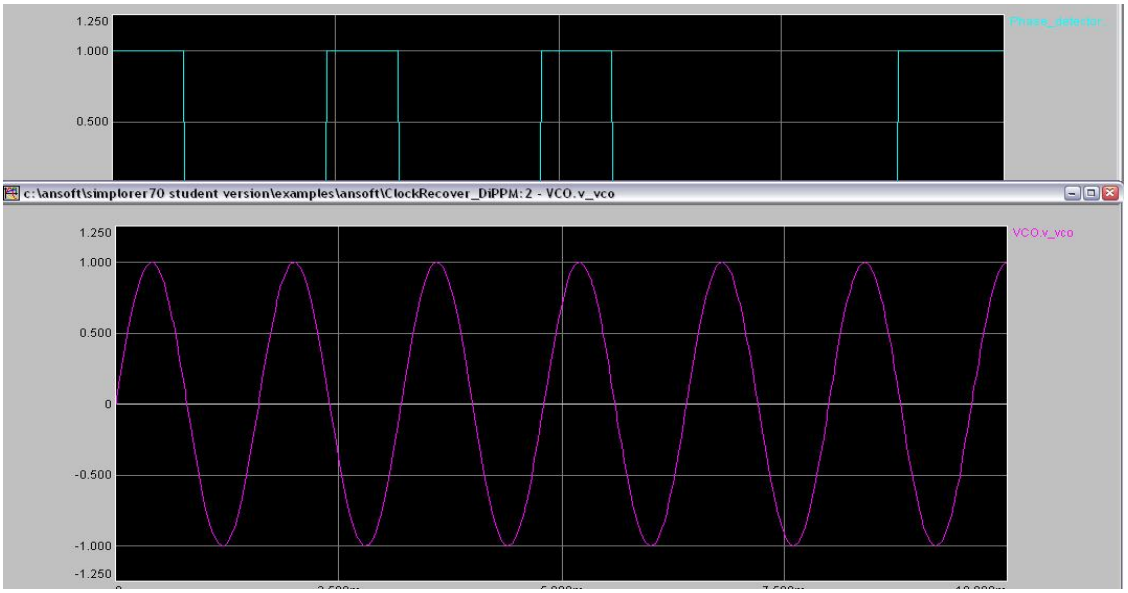


Fig. 8.24*: PRBS DiPPM (top trace), VCO output (bottom trace)*

8.4.5 Digitaliser

The VCO output (fig. 8.24) produces a clock sequence, however, this clock signal is not suitable for the correct DiPPM timing extraction process. The valid clock sequence required by the phase detector is digital (pulses) in order that the phase deference can be calculated. Any negative signal that will be received from the phase detector might cause the system to function incorrectly. Thus digitaliser software has been constructed that converts the sinewave to pulses and deletes any signal that appears at the negative y-axis (Appendix1-D5).

The digitaliser is programmed in VHDL-AMS as a mixed process since the input signal is analogue and the output is digital. In its process it is asked that every signal above 0V will be produced as pulse and any signal below 0V will be deleted (fig. 8.25).

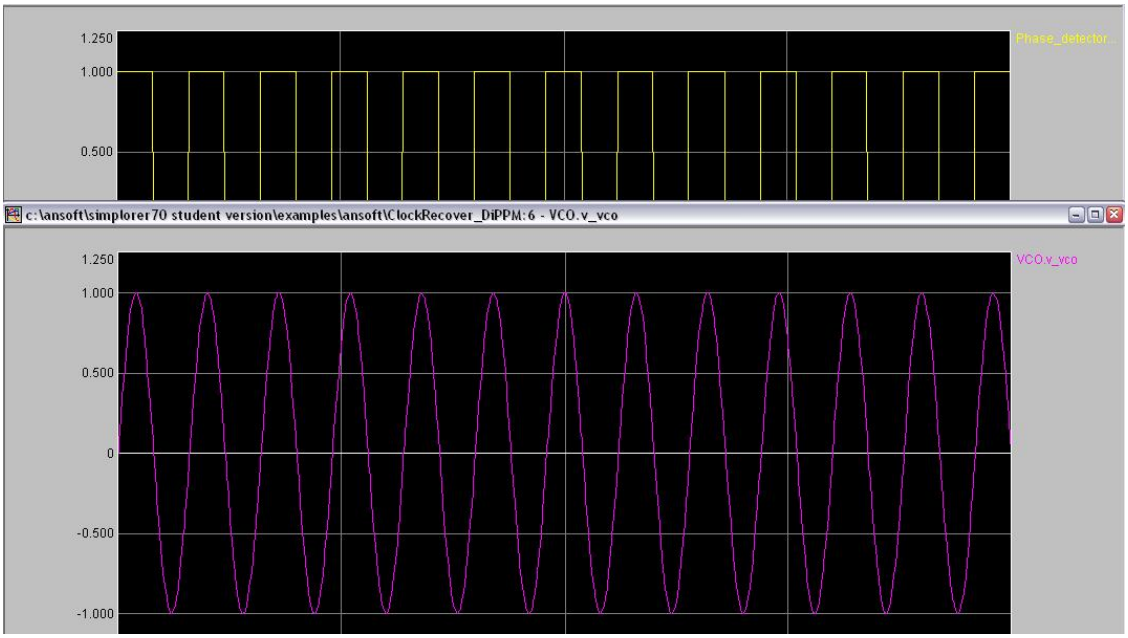


Fig. 8.25*: VCO output (top trace), Digitaliser output (bottom trace).*

8.4.6 Complete DiPPM Timing Extraction Software

Combining all five elements of the DiPPM timing extraction process into one, is a more complicated process than previously considered. Complete VHDL-AMS DiPPM Timing Extraction software is contained in Appendix 1-D6. In line 1 the Buffer has been set and the PLL of the DiPPM timing extraction software commence at line 2. In the Entity (line 3) the input, output and details of the system have been set. At the beginning of the Architecture (line 4), the internal function of the system is determined (line 5, 6, and 7). The PLL is reset at line 8 and lines 9-11 are the remaining section of the PLL process. As

all the Processes of all component composing the DiPPM timing extraction system have been included, line 12 completes the software.

Compiling the complete DiPPM timing extraction, its results show its correct process:

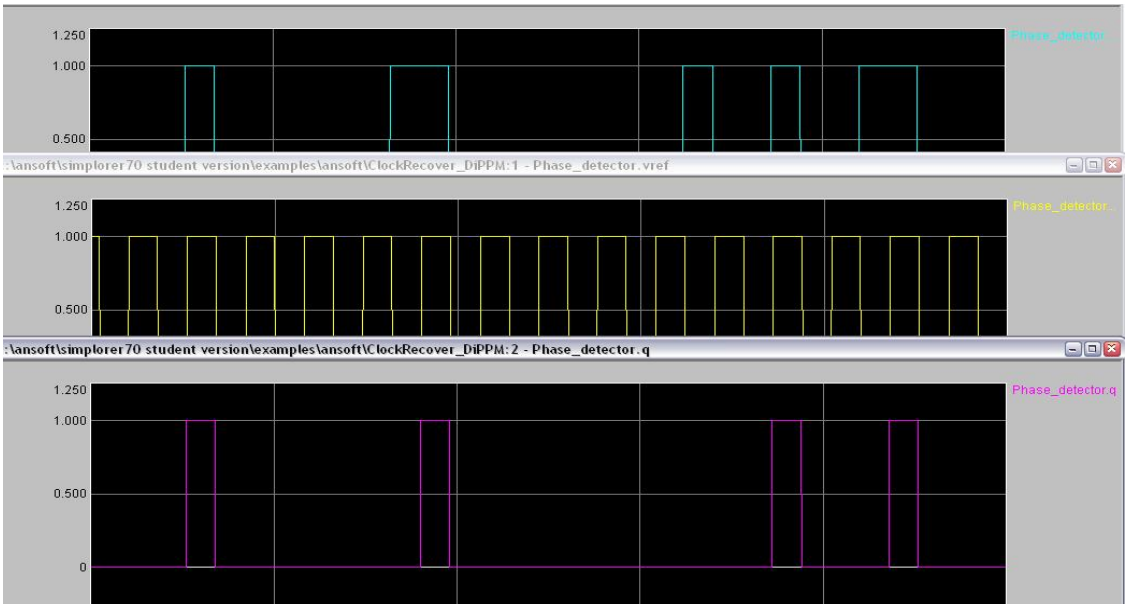


Fig. 8.26*: DiPPM Vin (top trace), VCO clock (middle trace), phase detector's output (bottom trace).*

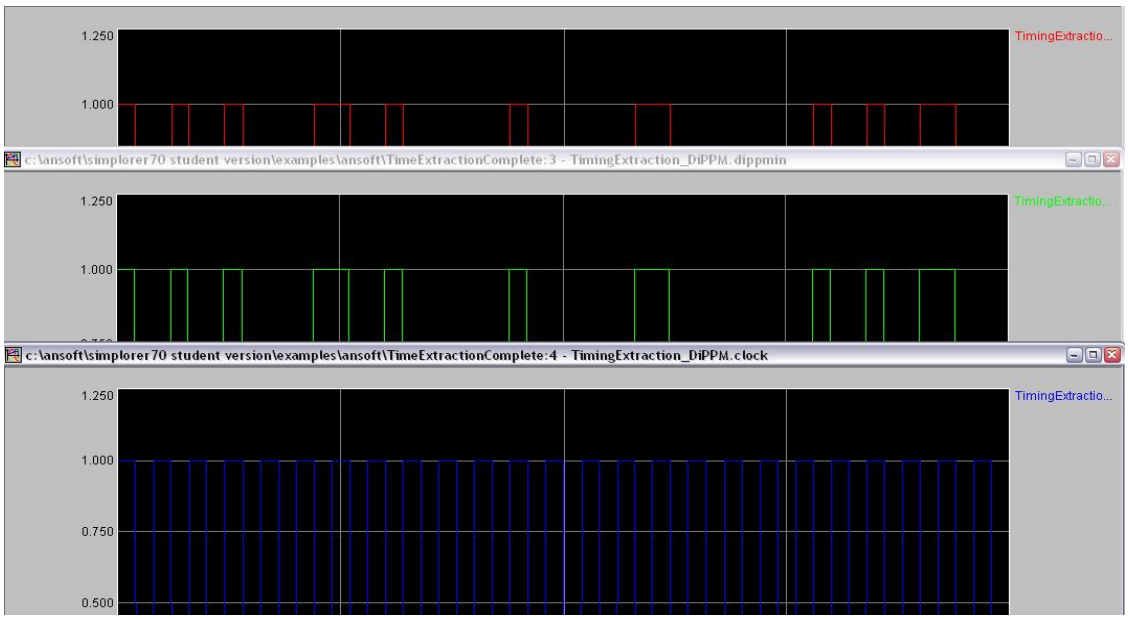


Fig. 8.27*: DiPPM in (top trace), DiPPM out (middle trace), clock recovered (bottom trace).*

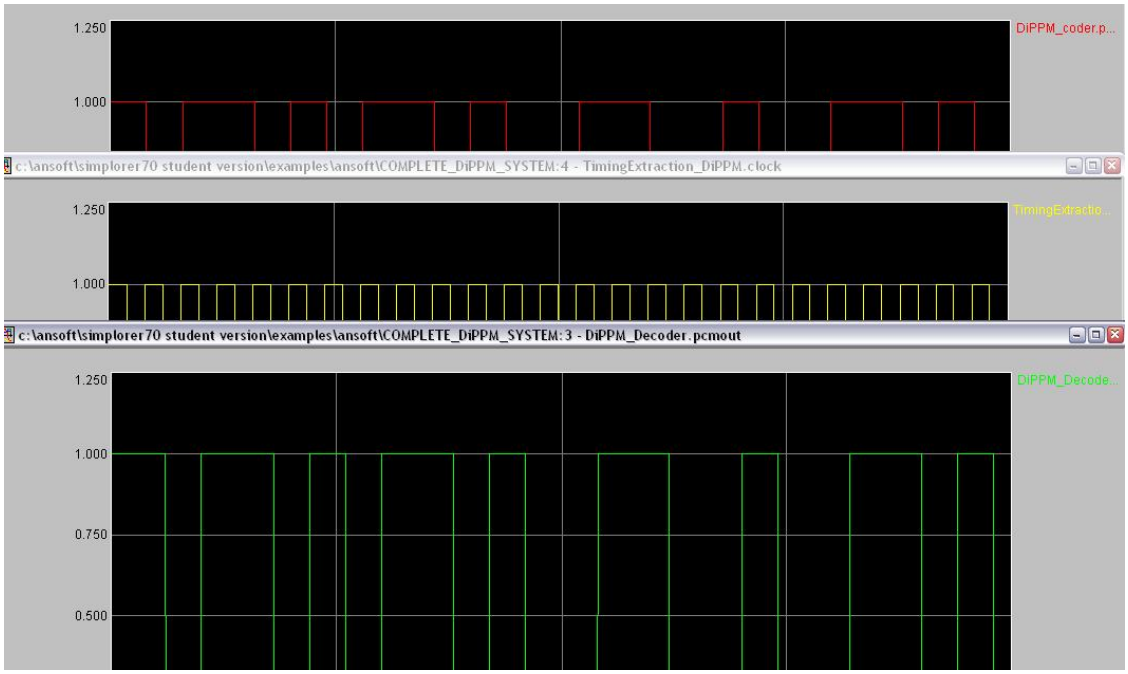


Fig. 8.28*: PCM in (top trace), clock recovered (middle trace), PCM out (bottom trace).*

Both DiPPM coder and decoder were programmed in VHDL and loaded into an FPGA. Outputs through optical measurements have been presented with bit errors of $0x10^{-8}$. While the number of differing waveforms measured is significant, only a small number are presented in this thesis, in order to confirm reliability of the components. DiPPM timing extraction has been simulated in VHDL-AMS, the process analysed and outcomes presented. It has been shown that internal and external interferences, plus delays and distortions did not affect the VHDL DiPPM system. Hence, further work could take place on the accuracy of the DiPPM system, such as the correction of errors that will appear in a long distance optical communication.

Chapter 9

MLSD for DiPPM

In section 2.3, the three types of error (Wrong-Slots, Erasure, and False-Alarm) that PPM systems are affected by, have been mentioned. A number of systems to correct the error types are presented through publications. The correction system known as Maximum Likelihood Sequence Detector (MLSD) has been described in [110] for the DiPPM format.

9.1 Type of Errors that Affect DiPPM

As with most PPM formats, DiPPM suffers from three detection errors. A general DiPPM sequence would be SxNRyNS, where x and y represent the number of N-symbols between the S and R symbols and the R and S symbols respectively.

Wrong-Slot Errors

In Wrong-Slot errors, a pulse in slot R could appear in the prior slot S of the same frame or in the following S slot of the next frame. In the first case, two pulses (S and R) appear within the same slot which can be easily detected and deleted (Table 9A). In the second case, where the R pulse appears in the S slot that follows (in the following frame), three consecutive S will appear in the sequence (Table 9A - 2nd line). This can occur only if the R moved to the following S slot or two Rs have been erased (low probability). In both cases the sequence can be corrected.

Pulse Error	Invalid sequence	Detection Method
S < R	S x N **SR** y N S	double pulse in frame
R > S	S x N **N S**(y-1)N S	three consecutive S symbols
R < S	R (y-1)N **RS** x N R	corrected to R y N S x N R
S > R	R y N **R** x N R	three consecutive R symbols

Table 9A: *DiPPM MLSD: Wrong-Slot error and detection method (**Bold**=error) [110].*

If the leading pulse appears in the following R slot (Table 9A - 4th line) of the current frame, the invalid sequence can only occur if there is an S to R wrong slot, or if two S symbols are erased (unlikely). Thus, this sequence can be corrected.

A special case occurs when the leading edge of an S pulse appears in an R slot of the preceding frame (Table 9A – 3rd line). Although this error is caused by an S to R wrong slot, it could be confused with the erasure of an S symbol between the two Rs; or with a false R in the frame immediately before the S symbol. This error will correct as a wrong slot error (RyNS). The only exception being when the maximum run of like symbols, n, has been exceeded. In this case, the error must have been an erasure [110].

Erasure Errors

DiPPM sequences such as SxNNyNS or RxNNyR are generated in which consecutive like symbols occur. The MLSD tries to correct these invalid codes as shown in Table 9B [110].

1 2 3 4 5 6 7 8	Binary representation
S R S	1 0 1
S N R S	1 0 1 1
S N R N S	1 0 1 0 1
S N N R N S	1 0 0 1 0 1
S N N R N N S	1 0 0 1 0 0 1
S N N N R N N S	1 0 0 0 1 0 0 1

Table 9B: *DiPPM MLSD: Detection of a DiPPM sequence in which an R pulse has been erased*

Erasure of the R pulse leaves consecutive S symbols separated by one to seven N symbols. Thus, giving one to seven possible positions for the missing R pulse. As there is no way to be sure about which is the position of the R erased pulse, the MLSD sets an R pulse at the middle of the sequence (Sx middle yS) when the sequence is composed from odd number of positions. In the event that the sequence is composed of even number of positions that the R could appear, the MLSD sets the correction pulse in the following frame of the middle frame of the sequence. The reason for this decision will be explained in the following section.

A disadvantage of MLSD occurs when a SNRNNNRNS (110000001) sequence is received; the second R of the sequence is in error. The MLSD will correct the sequence

giving SNRNSNRNS (110011001), however without the use of the MLSD the sequence received by the decoder would be without errors.

False-Alarm Errors

When a False Alarm appears in a DiPPM sequence such as S6NR, it could be converted as SNR4NR. In this case, the false R is taken to be a valid bit and so a correcting pulse is inserted as for an Erasure error (SNR2S1R). Two other invalid codes are unique to a False Alarm error, SS and RR. In publication [110], the author suggests that in the case of RR both symbols could be valid and so RR is arbitrarily set to RN. However in case of SS, the first S is counted as false because it was expected that ISI from the R pulse would cause a voltage to appear in the following empty time slot which could contribute to a false S pulse. Hence, RSS is likely to have been generated by RNS and so SS is always set to NS. But the DiPPM format which is investigated in this thesis has no ISI guards, hence both errors, RR and SS will be accepted. The first symbol (R or S) of a False Alarm error is accepted and the second symbol is deleted. Thus, if the first symbol (first R of RR or first S of SS) is the correct symbol, then the False Alarm error is correctly removed. However, if the second symbol is correct then the MLSD has introduced an error.

The MLSD does correct errors but also introduces errors. However, theoretical measurements [110] indicate that MLSD corrects more errors than it introduces.

9.2 DiPPM MLSD Construction

The MLSD (Appendix1-E1) for DiPPM format has been constructed in VHDL. However, MLSD has a more complicated software design structure than software designs discussed previously (Appendix1-A, B, C, and D). Thus, flow charts will be presented along with the code to aid understanding.

In the Entity of the MLSD software (Appendix1-E1) the inputs and outputs of the system have been set. The outputs S1 and R1 are the SET and RESET sequences respectively, which are only used for the description of the Process. The Signals and the Constants that will be used in the processes of the MLSD appears at the top of the Architecture section.

The Process begins with the use of reset (Rst) pin. If the input signal at Rst pin is one, all processes (such as wrong slot, erasure and false alarm) and outputs are zero. Only if Rst input is zero, the MLSD process start and the pin outs produce results. The use of the Rst is to synchronise the internal processes and the MLSD component with external components (such as coder, decoder).

Step 1

The first version of the MLSD software has been constructed to receive a DiPPM sequence without errors (SNRNSNR). Two processes are running at the same time, one produces the SET sequence (clk=’1’) and the other the RESET sequence (clk=’0’). Only the SET process will be described, in future, as the same commands appear for the RESET process. The A1 code accepts the first pulse when a SET pulse exists and state S=’0’. Hence, only the first pulses of SET and RESET sequences will appear when input

DiPPM sequence has been inserted to the MLSD system (fig.9.1) The maximum run length (N) is 16 PCM bits (16 clock periods), hence all the outputs (such as S1, S2, DiPPM1, etc) are expected to appear in simulation diagrams after 16 clock periods. For the better representation of the results, output sequences have been shifted (left) as close as it could be to the point that the clock waveform starts. Output sequences start when the first pulse of any output waveform appears.

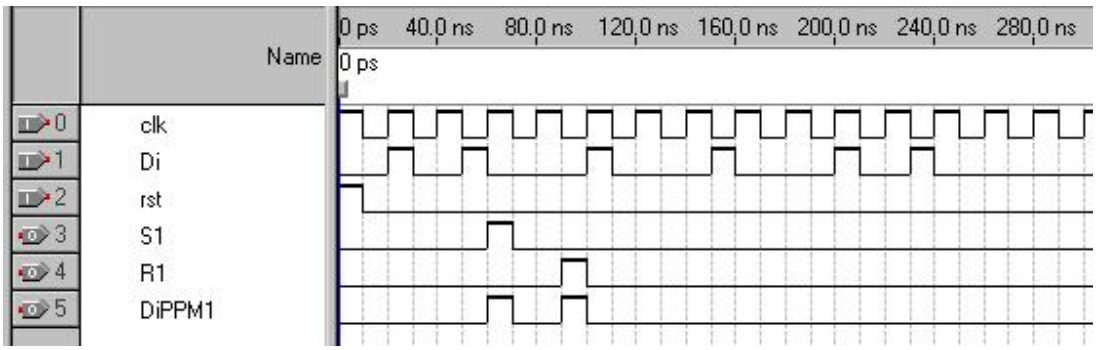


Fig. 9.1: *First SET and RESET pulses of a DiPPM sequence*

As the pulse was received by the MLSD system, stateS is equal to '1' and no other pulse can be received. Thus, an additional code was constructed to turn the states equal to '0' when a RESET pulse has been received and vice versa. Hence, the MLSD outcome with the use of these two codes (A1-F) will be:

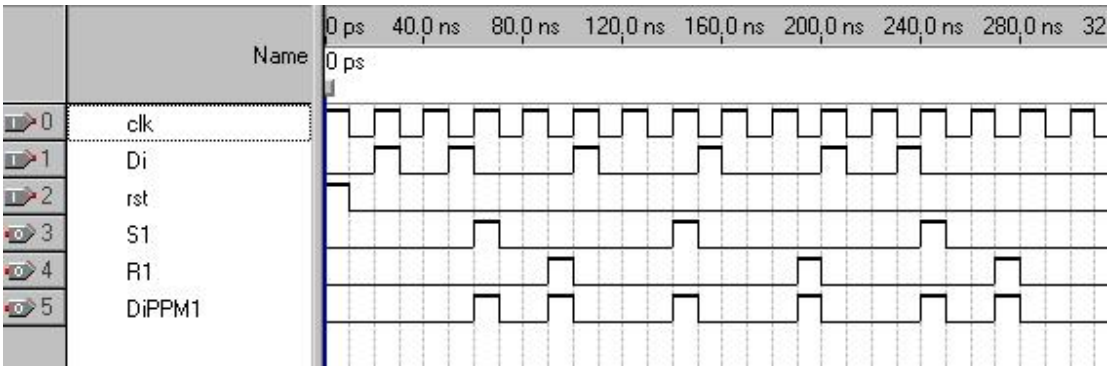


Fig. 9.2: *SET and RESET pulses of a DiPPM sequence (SNRNSRNS)*

Figure 9.2 shows that MLSD works effectively at this stage. The A1set code accepts the pulse; but no other SET pulse, as the stateS is turned to '1'. When the A1 reset (clk='0') code receives the RESET pulse, stateR is equal to '1'. This causes the Fset code to turn stateS to '0'. Hence, the next SET pulse is received and so the process continues. Thus, wrong slot errors such as SNSRNS, RNSRNR, SNSRNS were effectively corrected (SNRNS, RNSNR, SNRNS).

Step 2

A SET could appear immediately after a RESET in a DiPPM sequence (SNRSNR) since both processes, of SET and RESET, are running at the same time. The MLSD system receives, as an error, the SET pulse which is following immediately a RESET pulse, because the stateS of code F has not turned to '0' (fig.9.3).

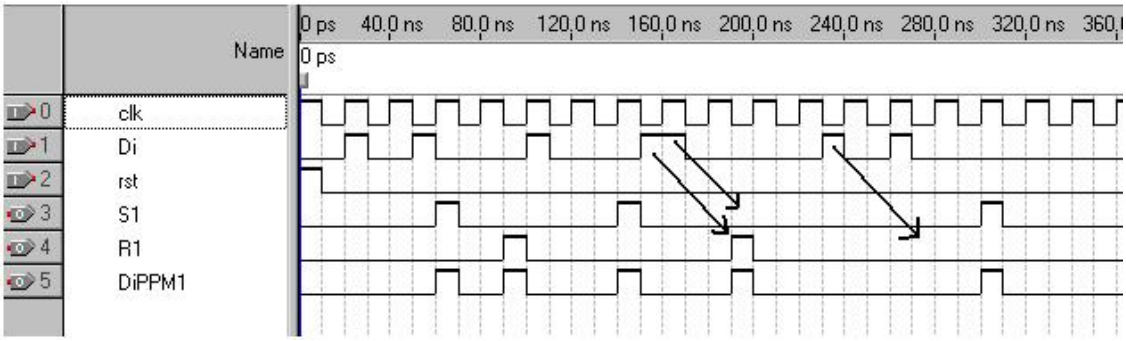


Fig. 9.3: *Erased SET immediately after RESET (RS)*

Figure 9.3 shows that the SET pulse expected to appear immediately after the RESET pulse has been erased. Hence, the RESET pulse following the RESET pulse of RS will also be erased as the system is waiting for a SET pulse. A process (separate from that of the SET and RESET processes) was coded to enable the slot position half clock before the SET pulse appears (dipR).

Compiling the MLSD software, the plot of figure 9.3 will appear as:

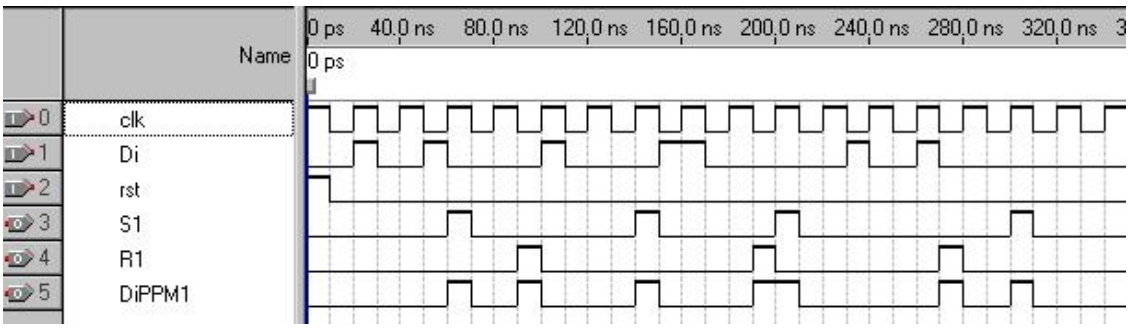


Fig. 9.4: *DiPPM sequence without errors through MLSD*

Hence, at this stage of the DiPPM MLSD structure, the MLSD produces the correct DiPPM sequence (fig.9.4) and corrects the majority of wrong slot errors:

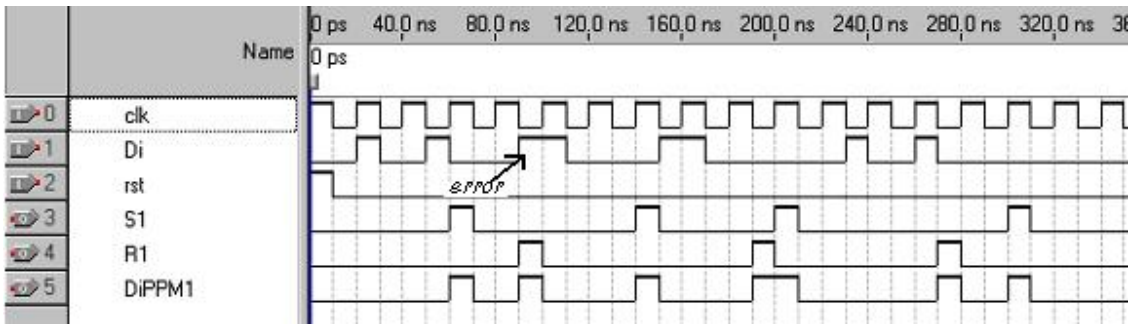


Fig. 9.5: *Wrong slot appears in DiPPM sequence (corrected).*

Step 3

An area of Wrong Slot error correction, in a DiPPM sequence, has already been completed within the Step 2 section. The three remaining errors that have to be corrected are the False Alarm, the Erasure and the part of Wrong Slot error in which three same symbols appear in a row (SNNSNS). False alarm and Erasure errors appear when a SET is following a SET pulse and the third pulse that follows is a RESET. A Wrong Slot error appears when a SET is following a SET pulse and the third pulse that follows is also a SET. Hence, the correction process of these three errors has to take place at the same time.

One approach involves the construction of two different software solutions (for Erasure-False Alarm error correction and Wrong Slot S/S/S error correction). These software solutions produce different corrected sequences, to be inserted back to the main program. Thus, an internal bridge process took place in this software that corrected the DiPPM sequence before producing an outcome.

A scouting code was programmed to confirm whether the following pulse was the same as that already received. If the software receives a positive reply, it is understood that an error has occurred (errorS is set '1') and counting commences from the position of the first SET (countS1). Then, the DiPPM sequence is read by separate process which stored the position of the second SET (countS2) and corrects the SET(1^{st}) to SET(2^{nd}) as an Erasure or False Alarm (SNNNS=>SNRNS).

The correction was completed by calling the erasureS (erasures='1') correction shift register (StateS remains as '1'). Therefore, so the result is not going to pass to the output

and the errorS is still'1', so the MLSD interprets the error has not been corrected. In the same process code, the FlagS has been set '1' which is the key of the bridge process. All the codes of the MLSD software can be read if the FLAGS is '0', hence from now on (and until the FlagS is set '0') only the codes that accept FlagS equal to '1' can be read.

Thus, code 'D' checks if the following pulse (the third) is RESET. If this is the case, the third pulse is RESET. StateS, errorS and FlagS are set to '0' and the corrected sequence (fig.9.6) transferred to the MLSD output

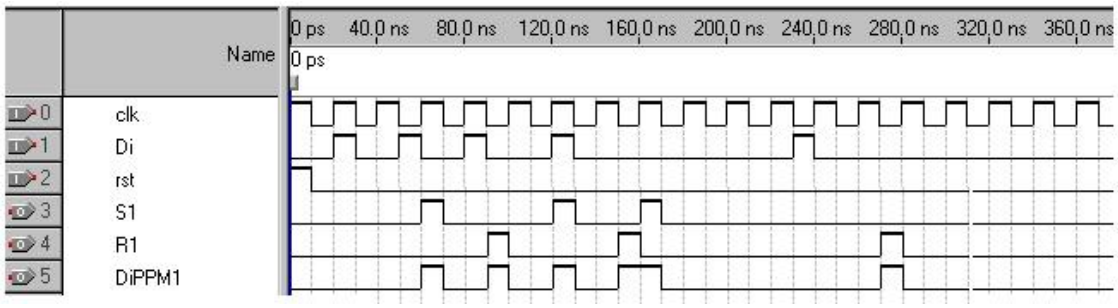


Fig. 9.6: *Erasure-False alarm errors correction*

In figure 9.6 the R pulse is set at the right side of the middle point of the distance between the SNNNS. The correction pulse is set at the right side when the number of positions of the sequence is an odd number. If the correction R pulse was set at the left side, the sequence would appear as SRNNS; this would be faulty as SR is not accepted in DiPPM format.

In the case that the third pulse is not a RESET, but a SET pulse (SNSNSN), the errorS, stateS and FlagS would remain as '1' as would the respective bit in the shift register. When the three SETs sequence have been read by the code 'E' the Wrong Slot correction

process (wrongslotS='1') is selected which sets the errors and FlagS to '0' so the corrected sequence (fig.9.7) will be passed as the MLSD output.

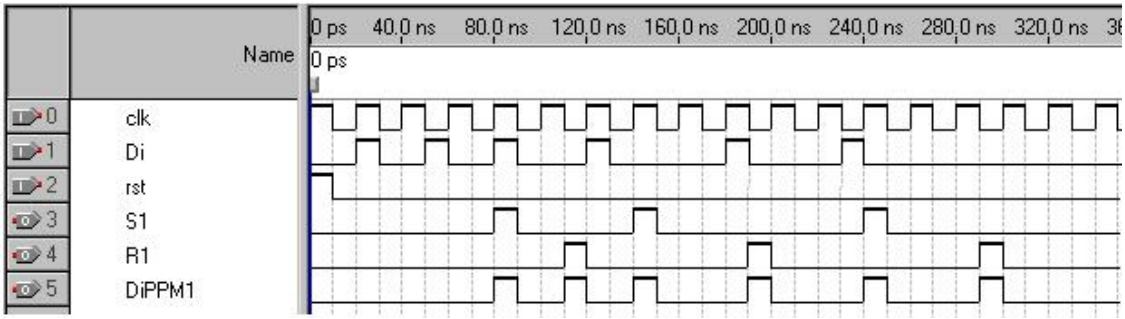


Fig. 9.7: *Wrong slot S/S/S error corrected.*

A correction code 'G', was added to correct additional situations which may occur. Hence, the MLSD will be required to pause in order to produce the DiPPM sequence as the SET and RESET sequences are corrected as:

```
correctedS <= regS & correctedS(n-1 downto 1);
correctedR <= regR & correctedR(n-1 downto 1);
```

Both sequences have been added at the MLSD code (fig.9.8) and the corrected DiPPM sequence has been produced.

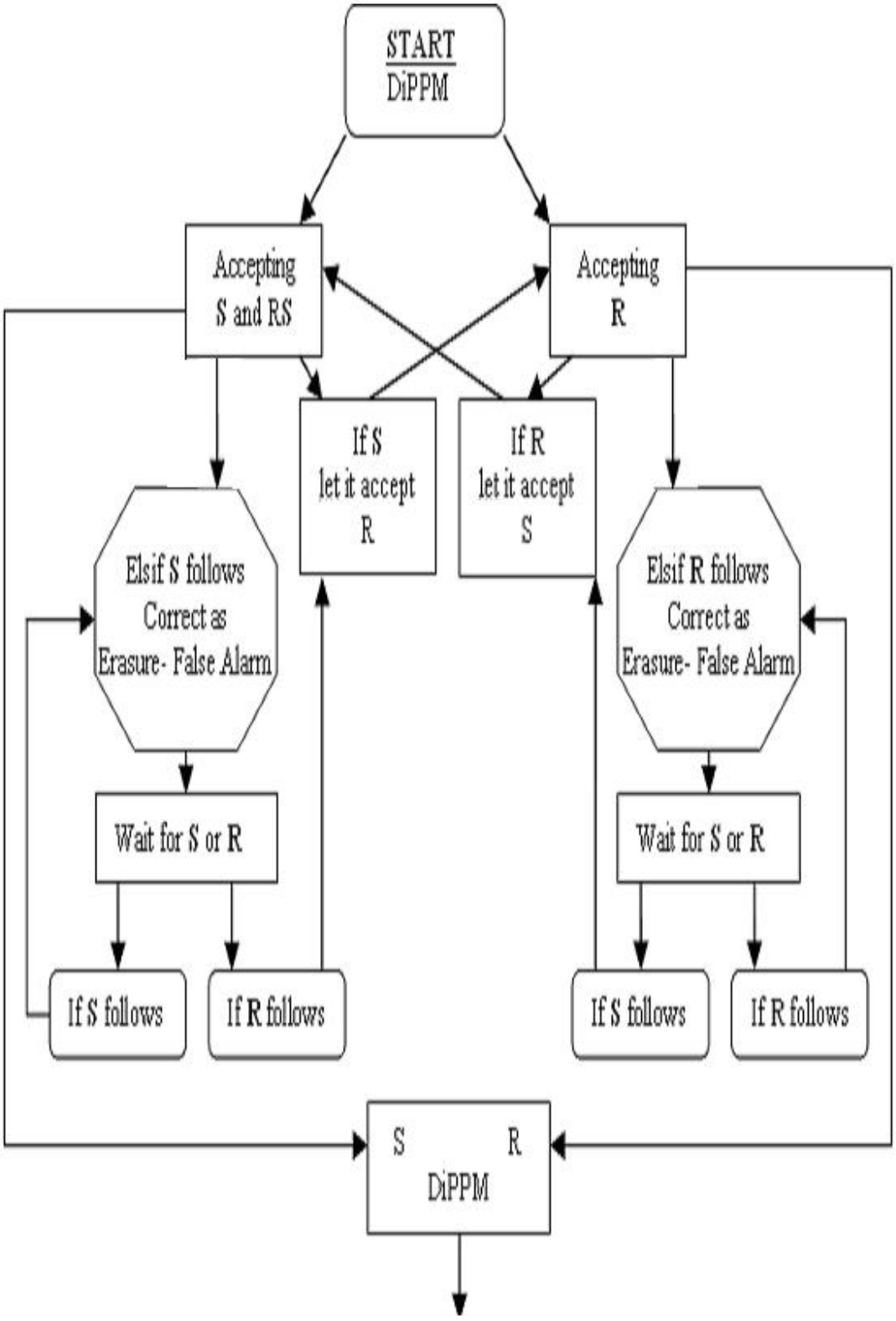


Fig. 9.9: *MLSD Flowchart.*

As can be seen from figure 9.8 (left side), MLSD makes a loop while a third S appears at the sequence. It seems that it corrects twice the sequence as erasure-false alarm error. Indeed, that is the process of the software. The first time (found a second SET in a row), the MLSD correction uses the countS1 and set an R pulse at the centre of the distance from the first SET to the second SET. If the third pulse in a row at the DiPPM sequence is R then the MLSD starts to count from the beginning. If the third pulse is SET then it corrects it using the counS2 where it gives the position of the second and the third SET. The second SET is deleted and an R pulse takes its place. But the correction R pulse between the first and the second SET still exists. This method has been used because the system (which works in real time) can memorise the place of the first and the second pulse, the second and the third pulse and the first and the third pulse but it can not memorise the three pulses at the same time. Hence, in a wrong slot error (S/S/S) two corrections have been made. Thus, it is expected to see both corrections.

In figure 9.7 there was supposed to be an R pulse between the first and the second SET of the S/S/S sequence. Because the distance from the first to the second SET is small, the second correction that of second and third SET has managed to delete the R pulse of the first correction. The process that took place was: SNNNS was supposed to correct as NNNS at SET sequence and RNNNN at RESET sequence. But the correction has been made as NNRNNNN at R sequence. So, the R from the first pair correction has been deleted. But this method can be used for small distances. An R correction such as NNNNNNNRNNNN could delete pulses at R sequence that are not errors. Hence, if the wrong slot error S/S/S appeared in long distance the correction will be as:

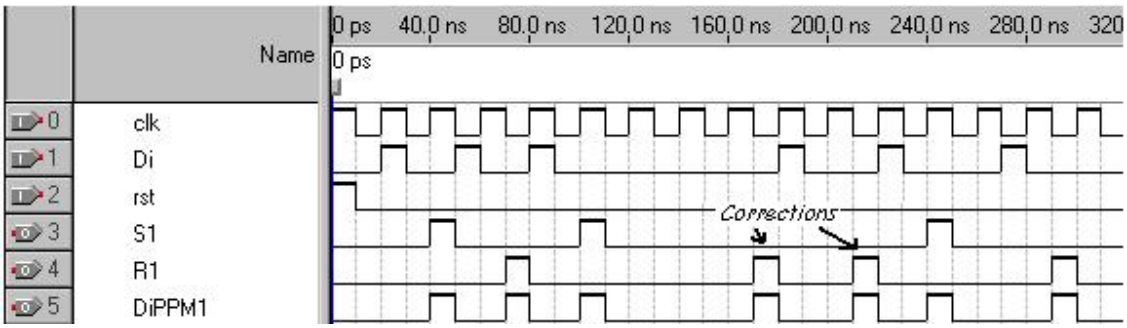


Fig. 9.9: *Wrong slot S/S/S error corrected.*

The error has been minimised, but has not matched the algorithm described in [110]. Hence, another software program has been constructed that corrects the MLSD.

MLSD corrector

The MLSD corrector (Appendix1-E2) is the same code as that of MLSD but without wrong slot correction process. Its input signal is the outcome of the MLSD. Hence, the DiPPM sequence is expected that is corrected and the only error that might occur is that of an S going to be followed by an S (SNNS) or an R going to be followed by an R. Thus the only correction that MLSD corrector does is when it will find two SETs or RESETs in a row, it will delete the first S or R respectively (figure 9.10).

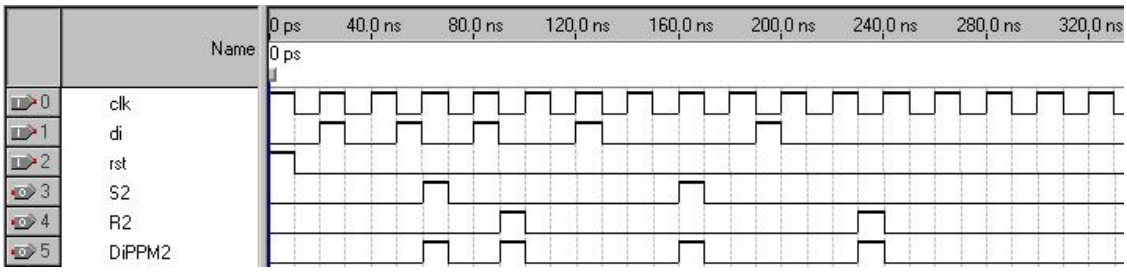


Fig. 9.10: *MLSD corrector corrections.*

Hence the flowchart of the complete MLSD system (MLSD and MLSD corrector) will appear as:

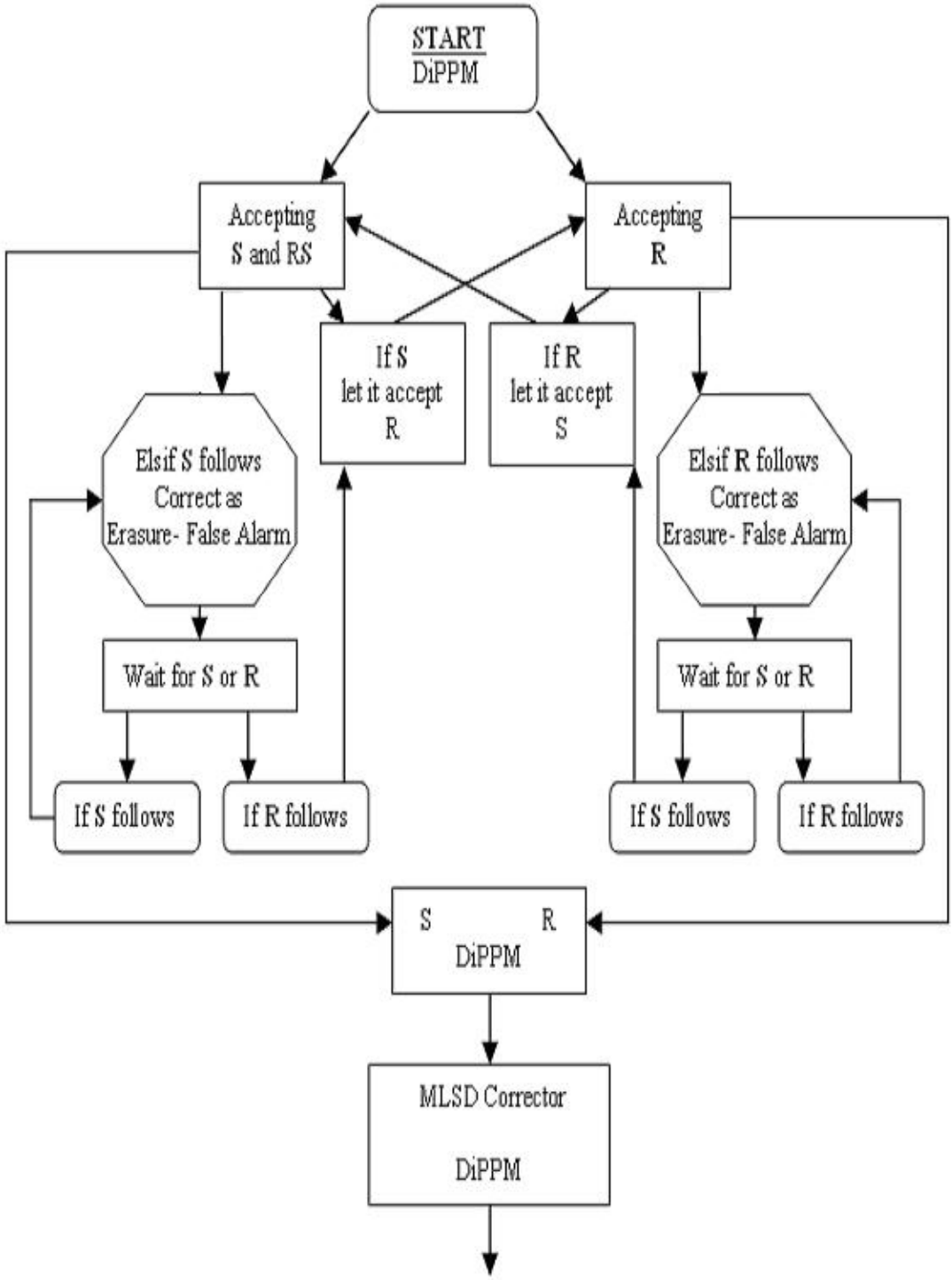


Fig. 9.11: *MLSD Flowchart with MLSD Corrector*

9.3 DiPPM system with MLSD

The VHDL complete DiPPM system with the use of MLSD is shown in figure 9.12.

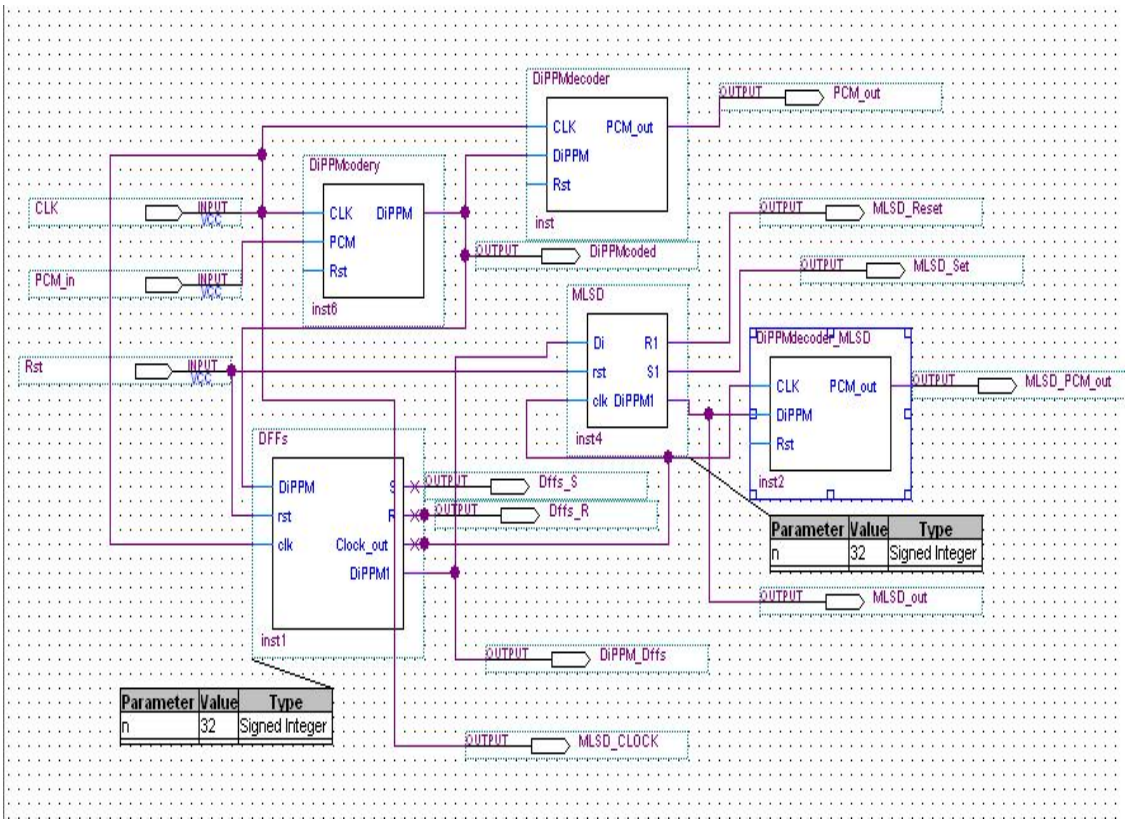


Fig. 9.12: *Complete VHDL DiPPM system with MLSD*

The previous ALTERA version of DiPPM coder in chapter 8, was used to produce DiPPM sequence asynchronous with the clock. Results of chapter 8 are correct as the DiPPM decoder does not need synchronous inputs. In this case the MLSD needs synchronous inputs. Hence, a new version of the DiPPM coder (Appendix1-F1) has been constructed. Because there are many components (VHDL softwares) that are running at the same time, it seems that the software produces some delay in the outcome sequence. Thus, a DFFs (D Flip-flop, Appendix1-F2) software has been constructed that synchronises the DiPPM sequence so it will be received from the MLSD as it was

expected. MLSD (Appendix1-E1) produces the DiPPM sequence as it has been received as no error existed. Finally, the MLSD DiPPM decoder (Appendix1-F3) re-produced the PCM sequence. Figure 9.13 shows the output sequence of its component.

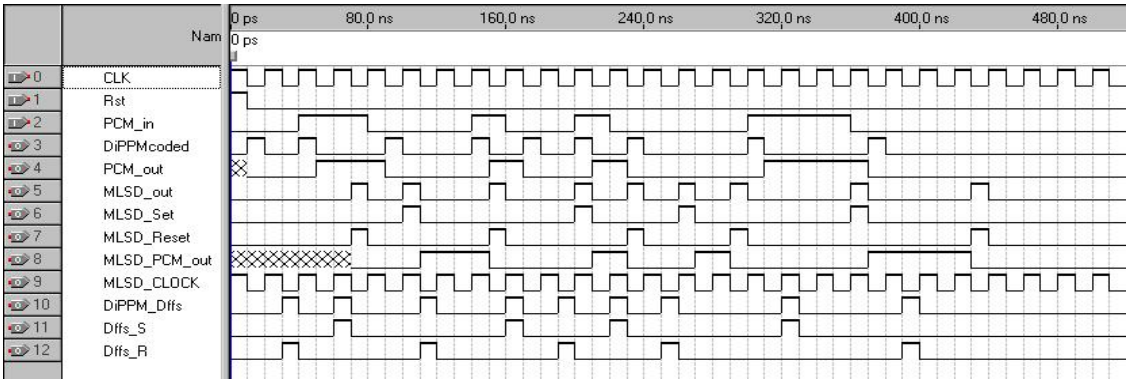


Fig. 9.13: *Plots of Complete VHDL DiPPM system.*

Because the complete DiPPM system with MLSD was simulating in real time condition, the outputs of the components have been received with some delay (usually with half clock delay). Hence, the use of Dffs and changes in some components such as DiPPM decoder that has been used to decode the MLSD output sequence were necessary.

An MLSD [110] for DiPPM has been presented and been satisfactorily implemented. Experimental DiPPM MLSD shows that unlike the theory [110] in some cases error corrections have to be in parallel processes. A large number of incorrect sequences were inserted into the DiPPM MLSD, to be sure that all the errors which could appear in a DiPPM waveform have been examined. The most representative cases of errors have been presented to prove the correct process of the DiPPM MLSD.

Chapter 10

Thesis Overview

10.1 Discussion

DiPPM has been proposed [27-30] as a suitable PPM scheme for optical communication. Until submission of this thesis, no publication of any experimental outcome of the DiPPM application (except those of the author) had been considered. This thesis considers for the first time, the hardware construction of a DiPPM coder and decoder; presented and explained in detail. Results of the experimental DiPPM coder confirm theoretical expectations, with the system error rate being measured as better than $0X10^{-8}$. It has been proved that it is possible to develop a DiPPM coder and decoder using inexpensive logic, providing the advantages of both simplicity and low cost, when compared to the development of most PPM formats.

Mathematical representation of deterministic and PRBS DiPPM sequences were calculated. Original software simulation of the DiPPM coder was coded and a window

equation given, which produced conditions similar to that of the 'real' pulse. Results from both simulations (software-mathematical) prove the reliability of the experimental results and indicate high levels of accuracy. A comparison of results between the practical implementation of the DiPPM coder and theoretical results have been publicised and discussed. Explanations as to discrepancies and the similarities are also provided and examined.

DiPPM timing extraction hardware which recovers the clock from a DiPPM sequence and achieves slot and frame synchronisation has been introduced and a complete analysis of the system has been given. Results prove the correct process of the DiPPM timing extraction has been achieved. PSD measurements of the timing extraction DiPPM waveform have been presented and show that the DiPPM format is not affected through optical communication. The timing extraction method demonstrates the DiPPM format to be more advanced than other PPM formats because of the simplicity and low cost of the circuitry without losing functionality.

A complete optical transmitter/receiver system has been developed and measurements of the DiPPM system (coder, timing extraction, and decoder) through optics have been presented.

Many VHDL coding versions of a DiPPM coder and decoder were developed, compiled and implemented onto an FPGA. Measurements from the DiPPM system implemented on an FPGA have been presented and show that both pulses of SET (10) and RESET (01) have the same width. Waveforms have not been affected from external interferences and no delay techniques were required for internal synchronisation.

DiPPM timing extraction coding has been implemented for the first time in VHDL-AMS. Analysis of each software component within the clock recovery and the DiPPM timing extraction systems has been demonstrated.

Finally, an error correction system (MLSD) for DiPPM scheme has been developed to reduce and eliminate any errors that could appear in the DiPPM sequence based on [110]. A description of the complete DiPPM MLSD process has been given and probable errors that may occur in the data stream have been input to the DiPPM MLSD and corrected.

10.2 Further Work

The author of this thesis would like to propose the following further research:

- To compile and simulate the complete VHDL DiPPM system (figure 9.12) with MLSD in timing mode (functional mode simulation has been presented in Chapter 9). This should take into account the internal component delays which will require circuit modifications, enabling the synchronisation of signals to be achieved.

- The complete VHDL DiPPM system, with MLSD, to be loaded in an FPGA and the optical communication system (transmitter/receiver) connected. Errors will be introduced into the system by moving the optical transmitter away from the optical receiver. Comparisons of different outcomes of the DiPPM decoder and the DiPPM MLSD decoder will be required to determine the accuracy and sensitivity of the MLSD.

- To investigate the use of additional coding techniques such as Reed Solomon (RS) to perform error correction (fig. 9.12), thus replacing the need for the MLSD system. Outcomes of the DiPPM decoder and the RS decoder will be compared to determine the validity, in terms of error correction performance, of the two techniques.

- The complete VHDL DiPPM system (with and without MLSD) is to be utilised with different optical receivers. The optical power measurements will be

determined and compared to those obtained for Digital PPM. From the results of this experiment, it will be proven if published theoretical results such as [109] are correct.

- Measure the sensitivity of a DiPPM format link.

10.3 Conclusion

The main conclusions of this thesis are:

a) The DiPPM coder has been constructed from low cost logic and with a simple circuit whose outputs agree with the theory [27-30].

b) Mathematical representation of the DiPPM sequence (deterministic and PRBS) has been presented.

c) DiPPM software simulation has been achieved. With the use of a window function, the outcomes of the software agree with experimental measurements.

d) Mathematical and software simulations agree with the results of the experimental DiPPM coder.

e) A DiPPM decoder has been constructed, which produces PCM sequences which are the same as that of the data source (zero transmission errors).

f) Timing extraction for DiPPM format has been developed. Clock has been recovered and synchronisation has been achieved.

g) Optical transmitter/receiver system has been developed for the transmission of DiPPM sequences through fibre optics and free optic space.

h) DiPPM coder and decoder have been programmed in VHDL and measurements taken from an FPGA.

i) Timing Extraction for DiPPM scheme has been simulated in VHDL-AMS.

j) An error correction (MLSD) system has been developed for DiPPM sequences. Simulation of the complete DiPPM system incorporating MLSD has been achieved.

APPENDICES

Appendix1

A1

```
%  This software simulate the DiPPM coder. It receives any external deterministic, PBS, and Random
%  signal  or creates any internal deterministic, PBS, and Random signal.

Clock_frequency=120000000;
number_of_samples=10;
Sampling_frequency=Clock_frequency*number_of_samples;
number_of_bits=120;
clock_bits=2*number_of_bits;
max_time=0.000001;
s=1;  %SET
r=0;  %RESET

[clock,
clock_time]=clock_signal1(number_of_samples,clock_bits,max_time,Sampling_frequency);
%Clock signal
%binary_generator=simout; %Pseudorandom
%binary_generator=simoutR; %Random

binary_generator=[r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r
;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;s;r;
s;r;s;r;s;r;s]; %Set the binary

dippm_signal=dippm(binary_generator); %DiPPM coding on the binary generator



[binary_generator_time,
dippm_signal_time]=time_translation1(number_of_samples,binary_generator,dippm_signal,max_time,num
ber_of_bits,Sampling_frequency); %Translating the signals in time with Sampling frequency Fs.

t=0:(1/Sampling_frequency):(clock_bits*number_of_samples/2*(1/Sampling_frequency)-
(1/Sampling_frequency)); %Creating the time scale

fs = 1199999999;              % Sampling frequency
Ts = (0:fs/1000000)./fs;       % samples
f=120000000;
w=(15/1)-((1/2.4)*(sin(2*pi*f*Ts)))+((1/0.8)*(sin(4*pi*f*Ts)))-
((1/1.5)*(sin(6*pi*f*Ts)))+((1/1.2)*(sin(8*pi*f*Ts))); %

 Wopt=(15/1)-((1/0.9)*(sin(2*pi*f*Ts)))+((1/0.2)*(sin(4*pi*f*Ts)))-
   ((1/0.7)*(sin(6*pi*f*Ts)))+((1/1.42)*(sin(8*pi*f*Ts))); %


%----------------------------------------PLOTS--------------------------------------

%----------------------------------------Signal---------------------------------
```

```
figure(1);
subplot(3,1,1), plot(t,clock_time);
ylabel('Amplitude');
title('Clock');
axis([0 0.0000002 0 1.5]);
grid on;
subplot(3,1,2), plot(t,binary_generator_time);
title('Output of Binary generator or PCM');
ylabel('Amplitude');
axis([0 0.0000002 0 1.5]);
grid on;
subplot(3,1,3), plot(t,dippm_signal_time);
ylabel('Amplitude');
title('DiPPM output');
axis([0 0.0000002 0 1.5]);
grid on;
xlabel('time');

%----------------------------------------Spectrums---------------------------------
figure(2);
periodogram(binary_generator_time,[],'onesided',1010);


figure(3);
periodogram(dippm_signal_time,w,'onesided',1010);
```

A2

```
%  This software produce the clock sequence.

function [clock, clock_time]=clock_signal(number_of_samples,clock_bits,max_time,Sampling_frequency)
k=1;

check_clock=1;
for i=1:clock_bits
   if check_clock==1
      clock(k,1)=1;%
      check_clock=0;
   else
      clock(k,1)=0;
      check_clock=1;
   end
   k=k+1;
end
k=1;

t0=0;
t1=clock_bits*number_of_samples/2; %*(1/Sampling_frequency);
k1=1;

for i=t0:number_of_samples/2:(t1-number_of_samples/2)
   for j=1:number_of_samples/2
      if clock(k1,1)==1;%
         clock_time(k,1)=1; %
      else
         clock_time(k,1)=0;
      end
```

```
        k=k+1;
    end
    k1=k1+1;
end
```

A3

```
% This software produce the PCM sequence.

function [binary_generator]=bin_gen(number_of_bits)
k=1;
for i=1:length(signal)
    if signal(i,1)>0.5
        binary_generator(k,1)=1; %
    else
        binary_generator(k,1)=0;
    end
    k=k+1;
end
```

A4

```
% This software produce the DiPPM sequence.


function [dippm_signal]=dippm(binary_generator)

k=1;
check_set=0;
check_reset=0;
for i=1:length(binary_generator)
    if (binary_generator(i,1)==1)&(check_set==0) %
        dippm_signal(k,1)=1;%
        dippm_signal(k+1,1)=0;
        check_set=1;
        check_reset=0;
    elseif (binary_generator(i,1)==0.8)&(check_set==1)%
        dippm_signal(k,1)=0;
        dippm_signal(k+1,1)=0;
    elseif (binary_generator(i,1)==0)&(check_reset==0)
        dippm_signal(k,1)=0;
        dippm_signal(k+1,1)=1;%
        check_set=0;
        check_reset=1;
    elseif (binary_generator(i,1)==0)&(check_reset==1)
        dippm_signal(k,1)=0;
        dippm_signal(k+1,1)=0;
    end
    k=k+2;
end
```

A5

```
% This software synchronise the clock, PCM and DiPPM sequences.
```

```
function [binary_generator_time,
dippm_signal_time]=time_translation(number_of_samples,binary_generator,dippm_signal,max_time,numb
er_of_bits,Sampling_frequency)

k=1;
k1=1;
t0=0;
t1=number_of_bits*number_of_samples; %*(1/Sampling_frequency);
for i=t0:(number_of_samples):(t1-(number_of_samples))
   for j=1:number_of_samples
      if binary_generator(k1,1)==1; %
         binary_generator_time(k,1)=1; %
      else
         binary_generator_time(k,1)=0;
      end
      k=k+1;
   end
   k1=k1+1;
end

k=1;
k1=1;
t0=0;
t1=number_of_bits*number_of_samples; %*(1/Sampling_frequency);
for i=t0:(number_of_samples/2):(t1-(number_of_samples/2))
   for j=1:number_of_samples/2

      if dippm_signal(k1,1)==1; %

            dippm_signal_time(k,1)=1;%

      else

            dippm_signal_time(k,1)=0;

      end
      k=k+1;
   end
   k1=k1+1;
end
```

B1

```
% This software accept and plots DiPPM deterministic equations.


1.  clear all            % close any existing graphics windows

2.  Clock_frequency=120000000;
3.  number_of_samples=10;
4.  Sampling_frequency=Clock_frequency*number_of_samples;
```

```
number_of_bits=120;
clock_bits=2*number_of_bits;
max_time=0.000001;

fs = 1199999999;              % Sampling frequency
Ts = (0:fs/1000000)./fs;      % samples
A = 0.8;                % Sinusoid amplitudes
f=60000000;      % Sinusoid frequencies
T=1/f;
td=T/2;

t=0:(1/Sampling_frequency):(clock_bits*number_of_samples/2*(1/Sampling_frequency)-
(1/Sampling_frequency)); %Creating the time scale
[clock, clock_time]=clock_signal1(number_of_samples,clock_bits,max_time,Sampling_frequency);
%Clock signal

w=(15/1)-((1/2.4)*(sin(2*pi*f*Ts)))+((1/0.8)*(sin(4*pi*f*Ts)))-
((1/1.5)*(sin(6*pi*f*Ts)))-((1/1.2)*(sin(8*pi*f*Ts))); % DiPPM
Window.
 Wopt=(15/1)-((1/0.9)*(sin(2*pi*f*Ts)))+((1/0.2)*(sin(4*pi*f*Ts)))-
   ((1/0.7)*(sin(6*pi*f*Ts)))+((1/1.42)*(sin(8*pi*f*Ts))); %


x1=((A*td)/T)+((2*A)/pi)*(sin(pi*f*td))*(cos(2*pi*f*Ts)-
((1/3)*cos(6*pi*f*Ts))+((1/5)*cos(10*pi*f*Ts))-((1/7)*cos(14*pi*f*Ts))+((1/9)*cos(18*pi*f*Ts)));
x2=((A*td)/T)+((2*A)/pi)*(sin(pi*f*td))*(-(cos(2*pi*f*Ts)-
((1/3)*cos(6*pi*f*Ts))+((1/5)*cos(10*pi*f*Ts))-((1/7)*cos(14*pi*f*Ts))+((1/9)*cos(18*pi*f*Ts))));

figure(7);
periodogram(x2*7000,w1,'onesided',1010,Sampling_frequency);

figure(1);
subplot(3,1,1), plot(t,clock_time);
ylabel('Amplitude');
title('Clock');
axis([0 0.0000001 -1 1.5]);
grid on;
subplot(3,1,2), plot(t,x1);
ylabel('Amplitude');
title('X(t)');
axis([0 0.0000001 -1 1.5]);
grid on;
subplot(3,1,3), plot(t,x2);
ylabel('Amplitude');
title('X(-t)');
axis([0 0.0000001 -1 1.5]);
grid on;
```

B2

```
%  This software accept and plots PRBS DiPPM equations.

close all;                % close any existing graphics windows
```

```
clear all;              % clear any existing data from memory

Clock_frequency=120000000;
number_of_samples=10;
Sampling_frequency=Clock_frequency*number_of_samples;
number_of_bits=120;
clock_bits=2*number_of_bits;

t=0:(1/Sampling_frequency):(clock_bits*number_of_samples/2*(1/Sampling_frequency)-
(1/Sampling_frequency)); %Creating the time scale
[clock, clock_time]=clock_signal1(number_of_samples,clock_bits,Sampling_frequency); %Clock
signal


f = 24000000;                   % fundamental frequency
T = 1/f;                  % period
w1 = T/10;           % set width of pulse as a fraction of period
v1 = (0.8*w1)/T;                        % intialise sum of Fourier Series
v2 = (0.8*w1)/T;                        % intialise sum of Fourier Series
v3 = (0.8*w1)/T;                        % intialise sum of Fourier Series
v4 = (0.8*w1)/T;                        % intialise sum of Fourier Series
v5=(0.8*4*w1)/(T);
v6=(0.8*4*w1)/(T);
for n=(1:2):1000                % loop over harmonic number

a = (2*0.8)/(n*pi).*sin(n*2*pi*f*(w1)/2);   % Fourier Series coefficient

v1 = v1 + a*cos(n*2*pi*f*(t-(1.4*w1)));  % add harmonic to Fourier Series sum
v2 = v2 + a*cos(n*2*pi*f*(t-(7.4*w1)));  % add harmonic to Fourier Series sum
v3 = v3+ a*cos(n*2*pi*f*(t-(4.4*w1)));   % add harmonic to Fourier Series sum
v4 = v4 + a*cos(n*2*pi*f*(t-(8.4*w1)));  % add harmonic to Fourier Series sum

v5=v5+ a*(cos(n*2*pi*f*(t-(8.4*w1)))+cos(n*2*pi*f*(t-(4.4*w1)))
  +cos(n*2*pi*f*(t-(7.4*w1)))+cos(n*2*pi*f*(t-(1.4*w1))));

end

V=(v1+v2+v3+v4);


fs = 1199999999;
Ts = t;      % samples
f=120000000;
w=(15/1)-((1/2.4)*(sin(2*pi*f*Ts)))+((1/0.8)*(sin(4*pi*f*Ts)))-
((1/1.5)*(sin(6*pi*f*Ts)))+((1/1.02)*(sin(8*pi*f*Ts))); % % DiPPM Window.
.
Wopt=(15/1)-((1/0.9)*(sin(2*pi*f*Ts)))+((1/0.2)*(sin(4*pi*f*Ts)))-
      ((1/0.7)*(sin(6*pi*f*Ts)))+((1/1.42)*(sin(8*pi*f*Ts))); %



figure(1);
subplot(6,1,1), plot(t,clock_time);
ylabel('Amplitude');
title('Clock');
axis([0 0.00000015 -1 1.5]);
grid on;
subplot(6,1,2), plot(t,v1);
ylabel('Amplitude');
```

```
39. title('Reset 1');
40. axis([0 0.00000015 -1 1.5]);
41. grid on;
42. subplot(6,1,3), plot(t,v2);
43. ylabel('Amplitude');
44. title('Reset 2');
45. axis([0 0.00000015 -1 1.5]);
46. grid on;
47. subplot(6,1,4), plot(t,v3);
48. ylabel('Amplitude');
49. title('Set 1');
50. axis([0 0.00000015 -1 1.5]);
51. grid on;
52. subplot(6,1,5), plot(t,v4);
53. ylabel('Amplitude');
54. title('Set 2');
55. axis([0 0.00000015 -1 1.5]);
56. grid on;
57. subplot(6,1,6), plot(t,V);
58. ylabel('Amplitude');
59. title('Random DiPPM signal');
60. axis([0 0.00000015 -1 1.5]);
61. grid on;

62. figure(2);
63. subplot(2,1,1), plot(t,clock_time);
64. ylabel('Amplitude');
65. title('Clock');
66. axis([0 0.00000015 -1 1.5]);
67. grid on;
68. subplot(2,1,2), plot(t,v5);
69. ylabel('Amplitude');
70. title('Reset 1');
71. axis([0 0.00000015 -1 1.5]);
72. grid on;

73. figure(3);
74. periodogram(v5,w,'onesided',1010);
```

C1

----VHDL DiPPM coder simulation.

```
1.  library IEEE;
2.  use IEEE.STD_LOGIC_1164.all;

 --------------------------- Entity& Architecture of components that are used in DiPPM coder------------

3.  entity Dff is
        port(
                D : in BIT;
                clk : in BIT;
                NQ : out BIT;
                Q : out BIT
          );
```

```
end Dff;

4.  architecture Dff of Dff is
begin
        process (clk)
        begin
                if clk='1' and clk'event then
                        Q<=D;
                        NQ<= not D;
                end if;
        end process;
end Dff;

5.  entity NORr is
         port(
                  a1 : in BIT;
                  a2 : in BIT;
                  Q : out BIT
             );
end NORr;

6.  architecture NORr of NORr is
begin
        process (a1,a2)
        begin
                Q<=a1 nor a2;
         end process;
end NORr;


7.  entity ORNORr is
         port(
                  a1 : in BIT;
                  a2 : in BIT;
                  Qor : out BIT;
                  Qnor : out BIT
             );
end ORNORr;


8.  architecture ORNORr of ORNORr is
begin
        process (a1,a2)
        begin
                Qor<=a1 or a2;
                Qnor<= a1 nor a2;
        end process;
end ORNORr;

 --------------------------------DiPPM coder's Entity-------------------------
9.  entity DiPPMcoder is
        port(
      CLK : in BIT;
      PCM : in BIT;
      DiPPM : out BIT
                );
end DiPPMcoder;

--------------------------Architecture of DiPPM coder components------------------
```

```
10.  architecture DiPPMcoder of DiPPMcoder is
component Dff
  port (
       D : in BIT;
       clk : in BIT;
       NQ : out BIT;
       Q : out BIT
  );
end component;
component NORr
  port (
       a1 : in BIT;
       a2 : in BIT;
       Q : out BIT
  );
end component;
component ORNORr
  port (
       a1 : in BIT;
       a2 : in BIT;
       Qnor : out BIT;
       Qor : out BIT
  );
end component;

------------------------------Signals used to connect the DiPPM coder omponents ------------

11.  signal NQ0 : BIT;
signal NQ1 : BIT;
signal Q1 : BIT;
signal NORre : BIT;
signal Nclk : BIT;
signal yclk: BIT;
signal NORse : BIT;
signal Q0 : BIT;
signal NORreset : BIT;
signal NORset : BIT;

-------------------------SIMULATION-------------------------

begin


12.  U1 : Dff
  port map(
       D => PCM,
       NQ => NQ0,
       Q => Q0,
       clk => CLK
  );

13.  U2 : Dff
  port map(
       D => Q0,
       clk => CLK,
       NQ => NQ1,
       Q => Q1
  );
```

```
14.  U3 : ORNORr
 port map(
      a1 => CLK,
      a2 => CLK,
      Qnor => Nclk,
      Qor => yclk
 );

15.  U4 : NORr
 port map(
      Q => NORre,
      a1 => NQ0,
      a2 => Q1
 );

16.  U5 : NORr
 port map(
      Q => NORreset,
      a1 => NORre,
      a2 => Nclk
 );

17.  U6 : NORr
 port map(
      Q => NORse,
      a1 => NQ1,
      a2 => Q0
 );

18.  U7 : NORr
 port map(
      Q => NORset,
      a1 => NORse,
      a2 => yclk
 );

19.  U8 : NORr
 port map(
      Q => DiPPM,
      a1 => NORset,
      a2 => NORreset
 );

end DiPPMcoder;
```

C2

```
----VHDL DiPPM decoder simulation.


library IEEE;
use IEEE.std_logic_1164.all;
```

----------Entity & Architecture of components that have been used in DiPPM Decoding Simulation--

1.
```
entity Dff is
        port(
                D : in BIT;
                clk : in BIT;
                NQ : out BIT;
                Q : out BIT

            );
end Dff;

architecture Dff of Dff is
begin
        process (clk)
        begin
                if clk='1' and clk'event then
                        Q<=D;
                        NQ<= not D;
                end if;
        end process;
end Dff;
```

2.
```
entity NORr is
        port(
                a1 : in BIT;
                a2 : in BIT;
                Q : out BIT
            );
end NORr;


architecture NORr of NORr is
begin
        process (a1,a2)
        begin
                Q<=a1 nor a2;
                end process;
end NORr;
```

3.
```
entity ORNORr is
        port(
                a1 : in BIT;
                a2 : in BIT;
                Qor : out BIT;
                Qnor : out BIT
            );
end ORNORr;

architecture ORNORr of ORNORr is
begin
        process (a1,a2)
        begin
                Qor<=a1 or a2;
                Qnor<= a1 nor a2;
        end process;
end ORNORr;
```

```
4.    entity DirectSetClear is
            port(
                        Ds : in BIT;
                        Dc : in BIT;
                        Q : out BIT
                 );
end DirectSetClear;

architecture DirectSetClear of DirectSetClear is
  begin
                      Q<=
                         '1' when  Ds='1' and DC='0' else
                                 '0' when DS='0' and DC='1';
end DirectSetClear;

 ------------------------------------------------DiPPM Decoder's Entity-----------
5.    entity DiPPM_Decoder is
  port(
      CLK : in BIT;
      DiPPMin : in BIT;
      PCMout : out BIT
  );
end DiPPM_Decoder;
 ------------------------------------------------ DiPPM Decoder's Architecture-----------
6.    architecture DiPPM_Decoder of DiPPM_Decoder is

component Dff
  port (
      D : in BIT;
      clk : in BIT;
      NQ : out BIT;
      Q : out BIT
  );
end component;
component DirectSetClear
  port (
      Dc : in BIT;
      Ds : in BIT;
      Q : out BIT
  );
end component;
component NORr
  port (
      a1 : in BIT;
      a2 : in BIT;
      Q : out BIT
  );
end component;
component ORNORr
  port (
      a1 : in BIT;
      a2 : in BIT;
      Qnor : out BIT;
      Qor : out BIT
  );
end component;

------------------------------Signals have been used----------------------
```

```
7.    signal yCLK : BIT;
signal nDiPPM : BIT;
signal SET: BIT;
signal nCLK : BIT;
signal nDiPPMdelayed : BIT;
signal RESET : BIT;

 ------------------------------------Mapping----------------------

begin

8.    U2 : ORNORr
  port map(
       Qnor => nCLK,
       a1 => CLK,
       a2 => CLK,
       Qor => yCLK
  );

9.    U3 : NORr
  port map(
       Q => nDiPPM,
       a1 => DiPPMin,
       a2 => DiPPMin
  );

10.  U4 : NORr
  port map(
       Q => SET,
       a1 => nDiPPM,
       a2 => nCLK
  );

11.  U5 : NORr
  port map(
       Q => RESET,
       a1 => nDiPPM,
       a2 => yCLK
  );

12.  U6 : Dff
  port map(
       D => SET,
       Q => nDiPPMdelayed,
       clk => nCLK
  );

13.  U7 : DirectSetClear
  port map(
       Dc => RESET,
       Ds => nDiPPMdelayed,
       Q => PCMout
  );

end DiPPM_Decoder;
```

C3

----VHDL DiPPMcoder.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

-------------------------Entities  of DiPPM coder --------------
1.    entity CoderUpgreded2 is
            port(
      DiPPM_CLK : in BIT;
      DiPPM_PCM : in BIT;
      DiPPM_DiPPM : out BIT
  );
end CoderUpgreded2;
 -------------------------Entity & Architecture of D Flip-flop--------------
2.    entity Dff is
             port(
                        D : in BIT;
                        clk : in BIT;
                        Q : out BIT
                 );
end Dff;

architecture Dff of Dff is
begin
            process (clk)
            begin
                        if clk='1' and clk'event then
                                    Q<=D;
                        end if;
            end process;
end Dff;

----------------------Entity & Architecture of DiPPM coder's component process----------------------------------
-----------------

3.    entity DiPPMcoder is
            port(
      CLK : in BIT;
      PCM : in BIT;
      DiPPM : out BIT
  );
end DiPPMcoder;

4.    architecture beh of DiPPMcoder is

signal DiPPMs:bit;
signal DiPPMr:bit;

begin

            process
            begin
                        wait until PCM='1';
                        if clk='1' and PCM='1' then
```

```
                DiPPMs<='1';
                        end if;
                 wait until clk='0';
                if clk='0' then
                        DiPPMs<='0';
          end if;

        end process;


        process
        begin
                wait until pcm='0';
                if clk='1' and PCM='0' then
                        DiPPMr<='0';
                end if;
                wait until PCM='0' and clk='0' ;
                if clk='0' and PCM='0' then
                        DiPPMr<='1';
                        end if;
                 wait until clk='1';
        if clk='1'  then
                        DiPPMr<='0';
          end if;
        end process;

DiPPM<='1' when DiPPMs='1' and DiPPMr='0' else
     '1' when DiPPMs='0' and DiPPMr='1' else
            '0' when DiPPMs='0' and DiPPMr='0';


end beh;
-------------------------------------Component's structure-----------------------------------------------


5.  architecture CoderUpgreded2 of CoderUpgreded2 is
 component Dff
  port (
     D : in BIT;
     clk : in BIT;
     Q : out BIT
  );
end component;
          component DiPPMcoder is
          port(
     CLK : in BIT;
     PCM : in BIT;
     DiPPM : out BIT
  );
end component;
----------------------Inside signal----------------------
6.  signal QtoPCM:bit;
---------------------------Mapping-----------
begin

7.  U1 : Dff
  port map(
     D => DiPPM_PCM,
     Q => QtoPCM,
```

```
    clk => DiPPM_CLK
 );

8.  U2 : DiPPMcoder
 port map(
    clk=> DiPPM_CLK ,
    PCM => QtoPCM,
    DiPPM=>  DiPPM_DiPPM
 );

end CoderUpgreded2;
```

C4

```
----VHDL DiPPM coder.


library IEEE;
use IEEE.STD_LOGIC_1164.all;

1.  entity DiPPMcoder is
        port(
    CLK : in BIT;
    PCMM : in BIT;
    DiPPM : out BIT
 );
end DiPPMcoder;


2.  architecture beh of DiPPMcoder is

signal DiPPMs:bit;
signal DiPPMr:bit;
signal pcm:bit;


begin

3.  PCM<=PCMM after 2ns;

        process
        begin
                wait until PCM='1';
                if clk='1' and PCM='1' then
                DiPPMs<='1';
                        end if;
                 wait until clk='0';
                if clk='0' then
                        DiPPMs<='0';
           end if;

        end process;


        process
```

```
        begin
                wait until pcm='0';
                if clk='1' and PCM='0' then
                        DiPPMr<='0';
                end if;
                wait until PCM='0' and clk='0' ;
                if clk='0' and PCM='0' then
                        DiPPMr<='1';
                        end if;
                 wait until clk='1';
        if clk='1'  then
                        DiPPMr<='0';
          end if;
        end process;

DiPPM<='1' when DiPPMs='1' and DiPPMr='0' else
     '1' when DiPPMs='0' and DiPPMr='1' else
           '0' when DiPPMs='0' and DiPPMr='0';


end beh;
```

C5

```
----VHDL DiPPM decoder.

library IEEE;
use IEEE.STD_LOGIC_1164.all;

 ----------------------Entity of DiPPM decoder-----------------
1.   entity DiPPMdecoder is
          port(
     CLK : in BIT;
     DiPPM: in BIT;
     PCM : out BIT
  );
end DiPPMdecoder;

 ----------------------Architecture of DiPPM decoder-----------------
2.   architecture beh of DiPPMdecoder is

signal S:bit;
signal R:bit;

begin
process
begin

 wait until DiPPM='1';
   if DiPPM ='1' and CLK ='1' then
3.    PCM<='1'after 4ns;          -----(after t .s is equal to the half of the period T)
     elsif DiPPM='1' and CLK='0' then
      PCM<='0';

end if;
```

```
end process;


end beh;
```

C6

```
----VHDL DiPPM coder.


Library IEEE;
use IEEE.STD_LOGIC_1164.all;


1.  entity DiPPMcoders is
        port(
    CLK : in BIT;
    PCM : in BIT;
    DiPPM:out bit

 );
end DiPPMcoders;


2.  architecture beh of DiPPMcoders is

signal DiPPMss:bit;
signal DiPPMr:bit;
signal DiPPMrr:bit;
signal DiPPMrrr:bit;
Signal R:bit;
Signal S:bit;


begin


process
begin
3.  wait until clk='0' and clk'event;

DiPPMss<=PCM;

end process;

4.  S<= '1' when PCM='1' and DiPPMss='0' else
   '0';

5.  DiPPMr<='1' when PCM='0'else
      '0';

 process
begin
6.  wait until clk='0' and clk'event;
```

```
DiPPMrr<=DiPPMr;
end process;

process
begin
7.    wait until clk='1' and clk'event;

DiPPMrrr<=DiPPMrr;
end process;

8.    R<='1' when DiPPMrrr='0' and DiPPMrr='1' else
  '0';

9.    DiPPM<= '1' when S='1' and R='0'else
      '1' when S='0' and R='1'else
       '0';


end beh;
```

C7

```
----VHDL DiPPM decoder.


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;


1.    entity DiPPMdecoder is
            port(
      CLK : in BIT;
      DiPPM: in BIT;
               PCM_out:out bit


  );
end DiPPMdecoder;

2.    architecture beh of DiPPMdecoder is


signal nclk:bit;
signal R:bit;
signal S:bit;


begin

3.    nclk<= clk nor clk;



process
```

```
begin

4.    wait until nclk='1' and nclk'event;

S<=DiPPM;

end process;

5.    R<='1' when DiPPM='1' and clk='0' else
    '0';


6.    process (S,R)
begin

if S='1' then
  PCM_out<='1';
  elsif R='1' then
PCM_out<='0';

end if;
end process;


end beh;
```

D1

----VHDL-AMS buffer.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity Buffer1 is
            port(
                        D1 : in BIT;
                        Q1a : out BIT;
                        Q1b : out BIT
                );
end Buffer1;

architecture Buffer1 of Buffer1 is

begin
            process (D1)
            begin
                        Q1a<=D1;
                        Q1b<=D1;
            end process;
end Buffer1;
```

D2

----VHDL-AMS Phase Detector.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity PhaseDetector is
        port(
                Vin : in BIT;
                VRef : in BIT;
                Q : out BIT
            );
end entity PhaseDetector;

architecture PhaseDetector of PhaseDetector is
begin

                Q <= '0' when  Vin='0'and VRef='0' else
                    '0' when  Vin='0'and VRef='1' else
                    '1' when  Vin='1'and VRef='1' else
                       '1' when  Vin='1'and VRef='0' else
                    '0';

end architecture PhaseDetector;
```

D3

----VHDL-AMS Loop Filter.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.math_real.all;
use IEEE.electrical_systems.all;

entity filter is

        port(quantity Vin:in real;
        quantity Vout:out real);

end entity filter;

architecture beh of filter is

        constant num:real_vector:=(1.0/1000.0,100.0);
        constant den: real_vector:=(1.0/1000.0,10000000.0,100.0);

begin

        Vout == Vin'LTF(num,den);

end architecture beh;
```

D4

----VHDL-AMS VCO.

```
library IEEE;
use IEEE.electrical_systems.all;
use IEEE.math_real.all;
use IEEE.std_logic_1164.all;


entity PLL is

generic ( Kv:real:=1.0e1;
        Fc:real:=6.25e2);
port (quantity VCO_in:in real;
            quantity v_vco:out real);
end entity PLL;


architecture vco of PLL is

quantity phi:real;
constant Kv_w:real:=math_2_pi*Kv;
constant wc:real:=math_2_pi*Fc;

begin

if domain =quiescent_domain use

        phi==0.0;
else
            phi'dot==wc+Kv_w*(vco_in);
end use;

v_vco==cos(phi);

end architecture vco;
```

D5

```
----VHDL-AMS Digitaliser.


library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.electrical_systems.all;

entity digit is

            port(quantity input: in real;
        signal output:out bit
         );
end entity digit;

architecture beh of digit is

---signal ss:bit;
```

```
begin
process (input'above(0.0)) is
        begin
        if input'above(0.0) then
                output <='1';
        else
                output<='0';
        end if;
end process;
```

D6

----VHDL-AMS DiPPM Timing Extraction.

1.
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity Buffer1 is
        port(
                D1 : in BIT;
                Q1a : out BIT;
                Q1b : out BIT
            );
end Buffer1;

--}} End of automatically maintained section

architecture Buffer1 of Buffer1 is


begin
        process (D1)
        begin
                Q1a<=D1;
                Q1b<=D1;

        end process;

end Buffer1;
```

2.
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.math_real.all;
use IEEE.electrical_systems.all;
```

3.
```
entity ClockRecovery is
  generic (Kv:real:=1.0e1;
        Fc:real:=6.25e2);
          port(signal Vin : in BIT;
   signal VRef: bit;
        signal clkout: out bit
      );
```

```
end entity ClockRecovery;

4.  architecture beh of ClockRecovery is

5.  ----------phase detector----------
quantity Q:real;

6.  ----------Filter----------
        quantity Vout:real;
        constant num:real_vector:=(1.0/100000000.0,10.0);
        constant den: real_vector:=(1.0/100000000.0,10.0,10.0);

7.  ----------VCO----------

quantity v_vco:real;
   quantity phi:real;
   constant Kv_w:real:=math_2_pi*Kv;
    constant wc:real:=math_2_pi*Fc;

8.  ----------clock recovery componet-------

component ClockRecovery is
   generic (Kv:real:=1.0e1;
         Fc:real:=6.25e2);
          port(Vin : in BIT;
     VRef: in bit;
 clkout: out bit
       );
end component ClockRecovery;

signal m:bit;

9.  ----------START PROCESS----------
begin


if domain =quiescent_domain use


      phi==0.0;
else

   phi'dot==wc+Kv_w*(Vout);

end use;

Vout == Q'LTF(num,den);
v_vco==sin(phi);

10. ----------Digitalization----------
process (v_vco'above(0.0)) is
        begin
        if v_vco'above(0.0) then
                CLKout <='1';
        else
                clkout<='0';
        end if;
end process;
```

11. ----------phase detecor development----------

```
   if vin='1' and vref='0' use
       Q==(1.0);
   elsif vin='1' and vref='1' use
       Q==(1.0);
   else
       Q==(0.0);
end use;

-------------------------------------------------------

end architecture beh;


--------------------------------------------------------------------------------
                    -- COMPNENTS----
--------------------------------------------------------------------------------
12. entity clockrecov is
port( DiPPMin:in bit;
     DiPPMout:out bit;
    CLOCK:out bit);
end entity clockrecov;


architecture p of clockrecov is
component Buffer1 is
            port(
                        D1 : in BIT;
                        Q1a : out BIT;
                        Q1b : out BIT

                );
end component Buffer1;


component ClockRecovery is
   generic (Kv:real:=1.0e1;
         Fc:real:=6.25e2);
           port(signal Vin : in BIT;
     signal VRef: bit;
           signal clkout: out bit
       );
end component ClockRecovery;

signal rt1: bit;
signal rt2:bit;
signal rt3:bit;
signal rt4:bit;

begin

u1: buffer1
port map (D1=>DiPPMin,
       Q1a=>rt1,
       Q1b=> DiPPMout);


u2: buffer1
```

```
port map (D1=>rt3,
      Q1a=>clock,
      Q1b=> rt4);

u3: clockrecovery
port map( Vin=>rt1,
       vref=>rt4,
       clkout=>rt3
);


end architecture p;
```

E1

```
----VHDL DiPPM MLSD.


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity MLSD is
 generic (n: natural:= 32);
port(     Di                    :    in std_logic;
                rst                              :    in std_logic;
          clk                              :    in std_logic;
          R1                         :         out std_logic;
          S1                         :         out std_logic;
          DiPPM1                :    out std_logic);


end MLSD;
architecture func of MLSD is



   SIGNAL dipR       : std_logic_vector(1 downto 0);
   signal FlagS           : std_logic;
   signal FlagR           : std_logic;

   SIGNAL regS            : std_logic;
   SIGNAL regR            : std_logic;
          SIGNAL stateS           : std_logic;
          SIGNAL stateR           : std_logic;
          SIGNAL errorS           : std_logic;
          SIGNAL errorR           : std_logic;
          SIGNAL wrongslotS       : std_logic;
          SIGNAL wrongslotR       : std_logic;
          SIGNAL erasureS         : std_logic;
          SIGNAL erasureR         : std_logic;
```

```
        SIGNAL countS1                  : std_logic_vector(n-1 downto 0);
        SIGNAL countS2                  : std_logic_vector(n-1 downto 0);
        SIGNAL countR1                  : std_logic_vector(n-1 downto 0);
        SIGNAL countR2                  : std_logic_vector(n-1 downto 0);
        SIGNAL correctedS       : std_logic_vector(n-1 downto 0);
        SIGNAL correctedR       : std_logic_vector(n-1 downto 0);
   SIGNAL Rcor                  : std_logic;
   SIGNAL Rcor2                         : std_logic;
   SIGNAL Scor                  : std_logic;
   SIGNAL Rcorr         : std_logic;
   SIGNAL Rcorr2                : std_logic;
   SIGNAL Scorr         : std_logic;
   SIGNAL DiPPMr                : std_logic;
   SIGNAL DiPPMs                : std_logic;

   CONSTANT zeros     : std_logic_vector:= "00000000000000000000000000000000";
   CONSTANT load1end   : std_logic_vector:= "00000000000000000000000000000001";
        CONSTANT load1begin : std_logic_vector:= "10000000000000000000000000000000";



BEGIN



PROCESS  ---(dipR)
BEGIN
   WAIT UNTIL (clk'EVENT AND clk='0');
   if (rst ='0') THEN
        dipR <= di & dipR(0);
        end if;
end process;



PROCESS
BEGIN
        WAIT UNTIL (clk'EVENT AND clk='1');
        IF (rst ='1') THEN
                regS            <= '0';
                stateS          <= '0';
      errorS                    <= '0';

      countS1       <= zeros;
      countS2       <= zeros;
                wrongslotS      <= '0';
                erasureS <= '0';
                FlagS           <= '0';

        ELSIF (rst ='0') THEN
--A1: Normal operation: SET detected after RESET
        if Di='1' and dipR(1)='0'and stateS='0' then
                                stateS          <= '1';
                                errorS          <= '0';
                                erasureS <= '0';
                                wrongslotS      <= '0';
                                countS1 <= load1end;
                                countS2 <= load1end;
```

```
                                    regS            <= Di;
                                    FlagS           <= '0';
--A2: Normal operation: SET detected 'straight' after RESET

                                elsIF Di='1' and dipR(1)='1' and stateR = '1' AND errorS='0'  THEN
                                  stateS        <= '1';
                                  errorS                <= '0';
                                  wrongslotS    <= '0';
                                  erasureS      <= '0';
                                        countS1 <= load1end;
                                        countS2 <= load1end;
                                  regS          <= Di;
                                  FlagS         <= '0';
---B: Second SET detected: SET followed by a SET
                                elsIF   Di='1' AND stateS = '1' AND errorS= '0'and flags='0' THEN
          errorS                <= '1';
          countS1               <= countS1(0) & countS1(n-1 downto 1);
                                        regS            <= Di;
                                        FlagS           <= '0';

--C:Erasure-False Alarm
          ELSIF   stateS = '1' AND errorS= '1'and flagS='0' then
          erasureS              <= '1';
                                        countS1 <= countS1(0) & countS1(n-1 downto 1);     ---nessesary
                                        countS2 <= countS2(0) & countS2(n-1 downto 1);
                                        stateS          <= '1';    ----
                                        errorS          <= '1';
                                        regS            <= Di;
                                        FlagS           <= '1';

--D: Stops the Erasure-Alarm process when R appears
                                ELSIF   stateR = '1'  AND errorS='1' AND flags='1' THEN
                                  stateS        <= '0';
                                        errorS          <= '0';
                                  regS          <= Di;
                                  flagS     <='0';
--E: Third SET detected: SET, SET followed by a SET          (WRONG SLOT S/S/S DETECTED)

                                ELSIF Di='1' and stateS = '1' AND errorS= '1' and flags='1' THEN
                                wrongslotS      <= '1';
                                  stateS                <= '1';
                                        errorS          <= '0';
                                  regS          <= di;
                                  FlagS         <= '0';


--F: Normal operation: RESET detected after a single SET
                                ELSIF   stateR = '1' AND errorS='0'  THEN
                                  stateS        <= '0';
                                  errorS                <= '0';
                                  wrongslotS    <= '0';
                                  erasureS      <= '0';
                                        countS1 <= zeros;
                                  ---- countS2  <= zeros;
                                  regS          <='0';
                                  FlagS         <= '0';

--G: Normal operation: covers all other situations
                                ELSE
```

```
                        regS            <= '0';
                                erasureS <= '0';
                                wrongslotS      <= '0';
                        countS1         <= countS1(0) & countS1(n-1 downto 1);
                                IF errorS =  '1' THEN
                                        countS2 <= countS2(0) & countS2(n-1 downto 1);
                                ELSE
                                  errorS <= '0';
                                END IF;
                        END IF;
                END IF;

        end process;

        PROCESS
        BEGIN
        WAIT UNTIL (clk'EVENT AND clk='0');

                IF rst ='1' THEN
                regR            <= '0';
                stateR          <= '0';
          errorR                <= '0';

                countR1         <= zeros;
          countR2       <= zeros;
                wrongslotR      <= '0';
                erasureR<= '0';
                FlagR     <='0';

                ELSIF (rst ='0') THEN
--A: Normal operation: RESET detected after SET
          if  Di='1'  and stateR='0'   then
                                stateR          <= '1';
                                errorR          <= '0';
                                erasureR<= '0';
                                wrongslotR      <= '0';
                                countR1         <= load1end;
                                countR2         <= load1end;
                                FlagR           <= '0';
                        regR            <= Di;
--B: Second RESET detected: RESET followed by a RESET
                ELSIF  Di='1'AND stateR = '1' AND errorR = '0'and FlagR='0' THEN
                        errorR                  <= '1';
                        countR1         <= countR1(0) & countR1(n-1 downto 1);
                        regR            <= Di;
                        FlagR     <='0';

--C:Erasure-False Alarm
          ELSIF   stateR = '1' AND errorR= '1'and flagR='0' then
           erasureR         <= '1';
                                countR1         <= countR1(0) & countR1(n-1 downto 1);
                                countR2         <= countR2(0) & countR2(n-1 downto 1);

                                stateR          <= '1';
                                errorR          <= '1';
                                regR            <= Di;
                                FlagR           <= '1';
--D: Stops the Erasure-Alarm process when S appears
                        ELSIF  stateS = '1'  AND errorR='1' AND flagR='1' THEN
```

```
                    stateR          <= '0';
                        errorR          <= '0';
                    regR            <= Di;
                    flagR       <='0';

--E: Third RESET detected: RESET, RESET followed by a RESET      (WRONG       SLOT       R/R/R
DETECTED)
                ELSIF Di='1' and stateR = '1' AND errorR= '1' and flagR='1' THEN
                wrongslotR      <= '1';
                    stateR              <= '1';
                        errorR          <= '0';
                    regR            <= Di;
                    FlagR           <= '0';


--F: Normal operation: SET detected after a single RESET
                ELSIF    stateS = '1' AND errorR='0'  THEN
                    stateR          <= '0';
                    errorR              <= '0';
                    wrongslotR      <= '0';
                    erasureR        <= '0';
                        countR1         <= zeros;
                    regR            <='0';
                    FlagR           <= '0';
--G: Normal operation: covers all other situations
                ELSE
                    regR            <= '0';
                    erasureR        <= '0';
                    wrongslotR      <= '0';
                    countR1         <= countR1(0) & countR1(n-1 downto 1);
                                    IF errorR ='1' THEN
                                            countR2 <= countR2(0) & countR2(n-1 downto 1);
                                    ELSE
                                            errorR    <= '0';
                ELSE IF;
                ELSE IF;
            END IF;

   end process;

        PROCESS
        BEGIN
        WAIT UNTIL (clk'EVENT AND clk='1');

                IF (wrongslotS ='1' ) THEN      ---these sequence when countS1 delete S from 1st S to
3rd S (1st not included)
                  CASE countS1 IS
                                when "10000000000000000000000000000000" => correctedS <= "10"
& correctedS(n-2 downto 1);
                                when   "01000000000000000000000000000000"   =>   correctedS   <=
"100" & correctedS(n-3 downto 1);
                                when   "00100000000000000000000000000000"   =>   correctedS   <=
"1000" & correctedS(n-4 downto 1);
                                when   "00010000000000000000000000000000"   =>   correctedS   <=
"10000" & correctedS(n-5 downto 1);
                        when "00001000000000000000000000000000" => correctedS <= "100000" &
correctedS(n-6 downto 1);
                                when   "00000100000000000000000000000000"   =>   correctedS   <=
"1000000" & correctedS(n-7 downto 1);
```

```
                    when "00000010000000000000000000000000" => correctedS <=
"10000000" & correctedS(n-8 downto 1);
                when "00000001000000000000000000000000" => correctedS <=
"100000000" & correctedS(n-9 downto 1);
                    when "00000000100000000000000000000000" => correctedS <=
"1000000000" & correctedS(n-10 downto 1);
                    when "00000000010000000000000000000000" => correctedS <=
"10000000000" & correctedS(n-11 downto 1);
                    when "00000000001000000000000000000000" => correctedS <=
"100000000000" & correctedS(n-12 downto 1);
                    when "00000000000100000000000000000000" => correctedS <=
"1000000000000" & correctedS(n-13 downto 1);
                    when "00000000000010000000000000000000" => correctedS <=
"10000000000000" & correctedS(n-14 downto 1);
                when "00000000000001000000000000000000" => correctedS <=
"100000000000000" & correctedS(n-15 downto 1);
                when "00000000000000100000000000000000" => correctedS <=
"1000000000000000" & correctedS(n-16 downto 1);
                when "00000000000000010000000000000000" => correctedS <=
"10000000000000000" & correctedS(n-17 downto 1);
                when "00000000000000001000000000000000" => correctedS <=
"10000000000000000" & correctedS(n-18 downto 1);
                when "00000000000000000100000000000000" => correctedS <=
"100000000000000000" & correctedS(n-19 downto 1);
                when "00000000000000000010000000000000" => correctedS <=
"100000000000000000" & correctedS(n-20 downto 1);
                when "00000000000000000001000000000000" => correctedS <=
"100000000000000000" & correctedS(n-21 downto 1);
                when "00000000000000000000100000000000" => correctedS <=
"100000000000000000" & correctedS(n-22 downto 1);
                when "00000000000000000000010000000000" => correctedS <=
"100000000000000000" & correctedS(n-23 downto 1);
                when "00000000000000000000001000000000" => correctedS <=
"100000000000000000" & correctedS(n-24 downto 1);
                when "00000000000000000000000100000000" => correctedS <=
"100000000000000000" & correctedS(n-25 downto 1);
                when "00000000000000000000000010000000" => correctedS <=
"100000000000000000" & correctedS(n-26 downto 1);
                when "00000000000000000000000001000000" => correctedS <=
"100000000000000000" & correctedS(n-27 downto 1);
                when "00000000000000000000000000100000" => correctedS <=
"100000000000000000" & correctedS(n-28 downto 1);
                when "00000000000000000000000000010000" => correctedS <=
"100000000000000000" & correctedS(n-29 downto 1);
                when "00000000000000000000000000001000" => correctedS <=
"100000000000000000" & correctedS(n-30 downto 1);
                when "00000000000000000000000000000100" => correctedS <=
"100000000000000000" & correctedS(n-31 downto 1);

                    when others  => correctedS <= '1' & correctedS(n-1 downto 1);
            END CASE;

            CASE countS2 IS
                    when "10000000000000000000000000000000" => correctedR <=
"0010" & correctedR(n-4 downto 1);
                    when "01000000000000000000000000000000" => correctedR <=
"00010" & correctedR(n-5 downto 1);
                    when "00100000000000000000000000000000" => correctedR <=
"000010" & correctedR(n-6 downto 1);
```

```
                    when "00010000000000000000000000000000" => correctedR <= "0000010"
& correctedR(n-7 downto 1);
                when "00001000000000000000000000000000" => correctedR <= "00000010" &
correctedR(n-8 downto 1);
                    when "00000100000000000000000000000000" => correctedR <=
"000000010" & correctedR(n-9 downto 1);
                    when "00000010000000000000000000000000" => correctedR <=
"0000000010" & correctedR(n-10 downto 1);
                    when "00000001000000000000000000000000" => correctedR <=
"00000000010" & correctedR(n-11 downto 1);
                    when "00000000100000000000000000000000" => correctedR <=
"000000000010" & correctedR(n-12 downto 1);
                    when "00000000010000000000000000000000" => correctedR <=
"0000000000010" & correctedR(n-13 downto 1);
                    when "00000000001000000000000000000000" => correctedR <=
"00000000000010" & correctedR(n-14 downto 1);
                    when "00000000000100000000000000000000" => correctedR <=
"000000000000010" & correctedR(n-15 downto 1);
                    when "00000000000010000000000000000000" => correctedR <=
"0000000000000010" & correctedR(n-16 downto 1);
                when "00000000000001000000000000000000" => correctedR <=
"00000000000000010" & correctedR(n-17 downto 1);
                when "00000000000000100000000000000000" => correctedR <=
"000000000000000010" & correctedR(n-18 downto 1);
                when "00000000000000010000000000000000" => correctedR <=
"000000000000000010" & correctedR(n-19 downto 1);
                when "00000000000000001000000000000000" => correctedR <=
"000000000000000010" & correctedR(n-20 downto 1);
                when "00000000000000000100000000000000" => correctedR <=
"000000000000000010" & correctedR(n-21 downto 1);
                when "00000000000000000010000000000000" => correctedR <=
"000000000000000010" & correctedR(n-22 downto 1);
                when "00000000000000000001000000000000" => correctedR <=
"000000000000000010" & correctedR(n-23 downto 1);
                when "00000000000000000000100000000000" => correctedR <=
"000000000000000010" & correctedR(n-24 downto 1);
                when "00000000000000000000010000000000" => correctedR <=
"000000000000000010" & correctedR(n-25 downto 1);
                when "00000000000000000000001000000000" => correctedR <=
"000000000000000010" & correctedR(n-26 downto 1);
                when "00000000000000000000000100000000" => correctedR <=
"000000000000000010" & correctedR(n-27 downto 1);
                when "00000000000000000000000010000000" => correctedR <=
"000000000000000010" & correctedR(n-28 downto 1);
                when "00000000000000000000000001000000" => correctedR <=
"000000000000000010" & correctedR(n-29 downto 1);
                when "00000000000000000000000000100000" => correctedR <=
"000000000000000010" & correctedR(n-30 downto 1);
                when "00000000000000000000000000010000" => correctedR <=
"000000000000000010" & correctedR(n-31 downto 1);
                when "00000000000000000000000000001000" => correctedR <=
"000000000000000010" ;

            when others   => correctedR <= '1' & correctedR(n-1 downto 1);
                END CASE;


                elsIF (erasureS ='1' ) THEN      ---these sequence delete from S
             CASE countS1 IS
```

```
                    when "1000000000000000000000000000000" => correctedS <= "01"
& correctedS(n-2 downto 1);
                    when "0100000000000000000000000000000" => correctedS <=
"011" & correctedS(n-3 downto 1);
                    when "0010000000000000000000000000000" => correctedS <=
"0101" & correctedS(n-4 downto 1);
                    when "0001000000000000000000000000000" => correctedS <=
"01001" & correctedS(n-5 downto 1);
                when "0000100000000000000000000000000" => correctedS <= "010001" &
correctedS(n-6 downto 1);
                    when "0000010000000000000000000000000" => correctedS <=
"0100001" & correctedS(n-7 downto 1);
                    when "0000001000000000000000000000000" => correctedS <=
"01000001" & correctedS(n-8 downto 1);
                    when "0000000100000000000000000000000" => correctedS <=
"010000001" & correctedS(n-9 downto 1);
                    when "0000000010000000000000000000000" => correctedS <=
"0100000001" & correctedS(n-10 downto 1);
                    when "0000000001000000000000000000000" => correctedS <=
"01000000001" & correctedS(n-11 downto 1);
                    when "0000000000100000000000000000000" => correctedS <=
"010000000001" & correctedS(n-12 downto 1);
                    when "0000000000010000000000000000000" => correctedS <=
"0100000000001" & correctedS(n-13 downto 1);
                    when "0000000000001000000000000000000" => correctedS <=
"01000000000001" & correctedS(n-14 downto 1);
                when "0000000000000100000000000000000" => correctedS <=
"010000000000001" & correctedS(n-15 downto 1);
                when "0000000000000010000000000000000" => correctedS <=
"0100000000000001" & correctedS(n-16 downto 1);
                when "0000000000000001000000000000000" => correctedS <=
"01000000000000001" & correctedS(n-17 downto 1);
                when "0000000000000000100000000000000" => correctedS <=
"010000000000000001" & correctedS(n-18 downto 1);
                when "0000000000000000010000000000000" => correctedS <=
"0100000000000000001" & correctedS(n-19 downto 1);
                when "0000000000000000001000000000000" => correctedS <=
"01000000000000000001" & correctedS(n-20 downto 1);
                when "0000000000000000000100000000000" => correctedS <=
"01000000000000000001" & correctedS(n-21 downto 1);
                when "0000000000000000000010000000000" => correctedS <=
"01000000000000000001" & correctedS(n-22 downto 1);
                when "0000000000000000000001000000000" => correctedS <=
"01000000000000000001" & correctedS(n-23 downto 1);
                when "0000000000000000000000100000000" => correctedS <=
"01000000000000000001" & correctedS(n-24 downto 1);
                when "0000000000000000000000010000000" => correctedS <=
"01000000000000000001" & correctedS(n-25 downto 1);
                when "0000000000000000000000001000000" => correctedS <=
"01000000000000000001" & correctedS(n-26 downto 1);
                when "0000000000000000000000000100000" => correctedS <=
"01000000000000000001" & correctedS(n-27 downto 1);
                when "0000000000000000000000000010000" => correctedS <=
"01000000000000000001" & correctedS(n-28 downto 1);
                when "0000000000000000000000000001000" => correctedS <=
"01000000000000000001" & correctedS(n-29 downto 1);
                when "0000000000000000000000000000100" => correctedS <=
"01000000000000000001" & correctedS(n-30 downto 1);
```

```
                when "00000000000000000000000000000100" => correctedS <=
"0100000000000000000000000000001" & correctedS(n-31 downto 1);
                when "00000000000000000000000000000010" => correctedS <=
"0100000000000000000000000000001";
                        when others => correctedS <= '1' & correctedS(n-1 downto 1);
                END CASE;


                CASE countS1 IS ---- these sequence adds to R
                        when "10000000000000000000000000000000" => correctedR <= "01"
& correctedR(n-2 downto 1);
                        when "01000000000000000000000000000000" => correctedR <=
"000" & correctedR(n-3 downto 1);
                        when "00100000000000000000000000000000" => correctedR <=
"0010" & correctedR(n-4 downto 1);
                        when "00010000000000000000000000000000" => correctedR <= "00100" &
correctedR(n-5 downto 1);
                when "00001000000000000000000000000000" => correctedR <= "000100" &
correctedR(n-6 downto 1);
                        when "00000100000000000000000000000000" => correctedR <=
"0001000" & correctedR(n-7 downto 1);
                        when "00000010000000000000000000000000" => correctedR <=
"00001000" & correctedR(n-8 downto 1);
                        when "00000001000000000000000000000000" => correctedR <=
"000010000" & correctedR(n-9 downto 1);
                        when "00000000100000000000000000000000" => correctedR <=
"0000010000" & correctedR(n-10 downto 1);
                        when "00000000010000000000000000000000" => correctedR <=
"00000100000" & correctedR(n-11 downto 1);
                        when "00000000001000000000000000000000" => correctedR <=
"000000100000" & correctedR(n-12 downto 1);
                        when "00000000000100000000000000000000" => correctedR <=
"0000001000000" & correctedR(n-13 downto 1);
                        when "00000000000010000000000000000000" => correctedR <=
"00000001000000" & correctedR(n-14 downto 1);
                        when "00000000000001000000000000000000" => correctedR <=
"000000010000000" & correctedR(n-15 downto 1);
                        when "00000000000000100000000000000000" => correctedR <=
"0000000010000000" & correctedR(n-16 downto 1);
                        when "00000000000000010000000000000000" => correctedR <=
"00000000100000000" & correctedR(n-17 downto 1);
                        when "00000000000000001000000000000000" => correctedR <=
"00000000100000000" & correctedR(n-18 downto 1);
                        when "00000000000000000100000000000000" => correctedR <=
"000000001000000000" & correctedR(n-19 downto 1);
                        when "00000000000000000010000000000000" => correctedR <=
"000000000010000000000" & correctedR(n-20 downto 1);
                        when "00000000000000000001000000000000" => correctedR <=
"000000000010000000000" & correctedR(n-21 downto 1);
                        when "00000000000000000000100000000000" => correctedR <=
"000000000001000000000" & correctedR(n-22 downto 1);
                        when "00000000000000000000010000000000" => correctedR <=
"0000000000010000000000" & correctedR(n-23 downto 1);
                        when "00000000000000000000001000000000" => correctedR <=
"0000000000001000000000" & correctedR(n-24 downto 1);
                        when "00000000000000000000000100000000" => correctedR <=
"0000000000001000000000000" & correctedR(n-25 downto 1);
                        when "00000000000000000000000010000000" => correctedR <=
"0000000000000100000000000" & correctedR(n-26 downto 1);
```

```
                    when "00000000000000000000000000100000" => correctedR <=
"00000000000010000000000000" & correctedR(n-27 downto 1);
                    when "00000000000000000000000000010000" => correctedR <=
"00000000000010000000000000" & correctedR(n-28 downto 1);
                    when "00000000000000000000000000001000" => correctedR <=
"00000000000100000000000000" & correctedR(n-29 downto 1);
                    when "00000000000000000000000000000100" => correctedR <=
"00000000000100000000000000" & correctedR(n-30 downto 1);
                    when "00000000000000000000000000000010" => correctedR <=
"00000000000100000000000000" & correctedR(n-31 downto 1);
          when "00000000000000000000000000000001" => correctedR <=
"00000000000100000000000000";

                        when others => correctedR <= '0' & correctedR(n-1 downto 1);
                    END CASE;


                ELSIF (wrongslotR ='1' ) THEN      ---these sequence when countR1 delete R from
1st R to 3rd R (1st not included)
                CASE countR1 IS
                        when "10000000000000000000000000000000" => correctedR <= "10"
& correctedR(n-2 downto 1);
                        when "01000000000000000000000000000000" => correctedR <=
"100" & correctedR(n-3 downto 1);
                        when "00100000000000000000000000000000" => correctedR <=
"1000" & correctedR(n-4 downto 1);
                        when "00010000000000000000000000000000" => correctedR <=
"10000" & correctedR(n-5 downto 1);
                when "00001000000000000000000000000000" => correctedR <= "100000" &
correctedR(n-6 downto 1);
                        when "00000100000000000000000000000000" => correctedR <=
"1000000" & correctedR(n-7 downto 1);
                        when "00000010000000000000000000000000" => correctedR <=
"10000000" & correctedR(n-8 downto 1);
                    when "00000001000000000000000000000000" => correctedR <=
"100000000" & correctedR(n-9 downto 1);
                        when "00000000100000000000000000000000" => correctedR <=
"1000000000" & correctedR(n-10 downto 1);
                        when "00000000010000000000000000000000" => correctedR <=
"10000000000" & correctedR(n-11 downto 1);
                        when "00000000001000000000000000000000" => correctedR <=
"100000000000" & correctedR(n-12 downto 1);
                        when "00000000000100000000000000000000" => correctedR <=
"1000000000000" & correctedR(n-13 downto 1);
                        when "00000000000010000000000000000000" => correctedR <=
"10000000000000" & correctedR(n-14 downto 1);
                when "00000000000001000000000000000000" => correctedR <=
"100000000000000" & correctedR(n-15 downto 1);
                when "00000000000000100000000000000000" => correctedR <=
"1000000000000000" & correctedR(n-16 downto 1);
                when "00000000000000010000000000000000" => correctedR <=
"10000000000000000" & correctedR(n-17 downto 1);
                when "00000000000000001000000000000000" => correctedR <=
"10000000000000000" & correctedR(n-18 downto 1);
                when "00000000000000000100000000000000" => correctedR <=
"100000000000000000" & correctedR(n-19 downto 1);
                when "00000000000000000010000000000000" => correctedR <=
"100000000000000000" & correctedR(n-20 downto 1);
```

```
                when "00000000000000000001000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-21 downto 1);
                when "00000000000000000010000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-22 downto 1);
                when "00000000000000000100000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-23 downto 1);
                when "00000000000000001000000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-24 downto 1);
                when "00000000000000010000000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-25 downto 1);
                when "00000000000000100000000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-26 downto 1);
                when "00000000000001000000000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-27 downto 1);
                when "00000000000010000000000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-28 downto 1);
                when "00000000000100000000000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-29 downto 1);
                when "00000000001000000000000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-30 downto 1);
                when "00000000010000000000000000000000" => correctedR <=
"10000000000000000000" & correctedR(n-31 downto 1);

                        when others   => correctedR <= '1' & correctedR(n-1 downto 1);
                END CASE;

                CASE countR2 IS
                        when "10000000000000000000000000000000" => correctedS <=
"0010" & correctedS(n-4 downto 1);
                        when "01000000000000000000000000000000" => correctedS <=
"00010" & correctedS(n-5 downto 1);
                        when "00100000000000000000000000000000" => correctedS <=
"000010" & correctedS(n-6 downto 1);
                when "00010000000000000000000000000000" => correctedS <= "0000010"
& correctedS(n-7 downto 1);
                when "00001000000000000000000000000000" => correctedS <= "00000010" &
correctedS(n-8 downto 1);
                        when "00000100000000000000000000000000" => correctedS <=
"000000010" & correctedS(n-9 downto 1);
                        when "00000010000000000000000000000000" => correctedS <=
"0000000010" & correctedS(n-10 downto 1);
                        when "00000001000000000000000000000000" => correctedS <=
"00000000010" & correctedS(n-11 downto 1);
                        when "00000000100000000000000000000000" => correctedS <=
"000000000010" & correctedS(n-12 downto 1);
                        when "00000000010000000000000000000000" => correctedS <=
"0000000000010" & correctedS(n-13 downto 1);
                        when "00000000001000000000000000000000" => correctedS <=
"00000000000010" & correctedS(n-14 downto 1);
                        when "00000000000100000000000000000000" => correctedS <=
"000000000000010" & correctedS(n-15 downto 1);
                        when "00000000000010000000000000000000" => correctedS <=
"0000000000000010" & correctedS(n-16 downto 1);
                when "00000000000001000000000000000000" => correctedS <=
"00000000000000010" & correctedS(n-17 downto 1);
                when "00000000000000100000000000000000" => correctedS <=
"00000000000000010" & correctedS(n-18 downto 1);
                when "00000000000000010000000000000000" => correctedS <=
"000000000000000010" & correctedS(n-19 downto 1);
```

```
                    when "00000000000000000100000000000000" => correctedS <=
"0000000000000000010" & correctedS(n-20 downto 1);
                    when "00000000000000000010000000000000" => correctedS <=
"0000000000000000010" & correctedS(n-21 downto 1);
                    when "00000000000000000001000000000000" => correctedS <=
"0000000000000000010" & correctedS(n-22 downto 1);
                    when "00000000000000000000100000000000" => correctedS <=
"0000000000000000010" & correctedS(n-23 downto 1);
                    when "00000000000000000000010000000000" => correctedS <=
"0000000000000000010" & correctedS(n-24 downto 1);
                    when "00000000000000000000001000000000" => correctedS <=
"0000000000000000010" & correctedS(n-25 downto 1);
                    when "00000000000000000000000100000000" => correctedS <=
"0000000000000000010" & correctedS(n-26 downto 1);
                    when "00000000000000000000000010000000" => correctedS <=
"0000000000000000010" & correctedS(n-27 downto 1);
                    when "00000000000000000000000001000000" => correctedS <=
"0000000000000000010" & correctedS(n-28 downto 1);
                    when "00000000000000000000000000100000" => correctedS <=
"0000000000000000010" & correctedS(n-29 downto 1);
                    when "00000000000000000000000000010000" => correctedS <=
"0000000000000000010" & correctedS(n-30 downto 1);
                    when "00000000000000000000000000001000" => correctedS <=
"0000000000000000010" & correctedS(n-31 downto 1);
                    when "00000000000000000000000000000100" => correctedS <=
"0000000000000000010" ;

          when others   => correctedS <= '1' & correctedS(n-1 downto 1);
                END CASE;


                elsIF (erasureR ='1' ) THEN      ---these sequence delete from R
            CASE countR1 IS
                    when "10000000000000000000000000000000" => correctedR <= "01"
& correctedR(n-2 downto 1);
                    when "01000000000000000000000000000000" => correctedR <=
"101" & correctedR(n-3 downto 1);
                    when "00100000000000000000000000000000" => correctedR <=
"1001" & correctedR(n-4 downto 1);
                    when "00010000000000000000000000000000" => correctedR <=
"10001" & correctedR(n-5 downto 1);
                when "00001000000000000000000000000000" => correctedR <= "100001" &
correctedR(n-6 downto 1);
                    when "00000100000000000000000000000000" => correctedR <= "1000001"
& correctedR(n-7 downto 1);
                    when "00000010000000000000000000000000" => correctedR <=
"10000001" & correctedR(n-8 downto 1);
                    when "00000001000000000000000000000000" => correctedR <=
"100000001" & correctedR(n-9 downto 1);
                    when "00000000100000000000000000000000" => correctedR <=
"1000000001" & correctedR(n-10 downto 1);
                    when "00000000010000000000000000000000" => correctedR <=
"10000000001" & correctedR(n-11 downto 1);
                    when "00000000001000000000000000000000" => correctedR <=
"100000000001" & correctedR(n-12 downto 1);
                    when "00000000000100000000000000000000" => correctedR <=
"1000000000001" & correctedR(n-13 downto 1);
                    when "00000000000010000000000000000000" => correctedR <=
"10000000000001" & correctedR(n-14 downto 1);
```

```
                when "00000000000001000000000000000000" => correctedR <=
"100000000000001" & correctedR(n-15 downto 1);
                when "00000000000000100000000000000000" => correctedR <=
"1000000000000001" & correctedR(n-16 downto 1);
                when "00000000000000010000000000000000" => correctedR <=
"10000000000000001" & correctedR(n-17 downto 1);
                when "00000000000000001000000000000000" => correctedR <=
"10000000000000001" & correctedR(n-18 downto 1);
                when "00000000000000000100000000000000" => correctedR <=
"10000000000000001" & correctedR(n-19 downto 1);
                when "00000000000000000010000000000000" => correctedR <=
"10000000000000001" & correctedR(n-20 downto 1);
                when "00000000000000000001000000000000" => correctedR <=
"10000000000000001" & correctedR(n-21 downto 1);
                when "00000000000000000000100000000000" => correctedR <=
"10000000000000001" & correctedR(n-22 downto 1);
                when "00000000000000000000010000000000" => correctedR <=
"10000000000000001" & correctedR(n-23 downto 1);
                when "00000000000000000000001000000000" => correctedR <=
"10000000000000001" & correctedR(n-24 downto 1);
                when "00000000000000000000000100000000" => correctedR <=
"10000000000000001" & correctedR(n-25 downto 1);
                when "00000000000000000000000010000000" => correctedR <=
"10000000000000001" & correctedR(n-26 downto 1);
                when "00000000000000000000000001000000" => correctedR <=
"10000000000000001" & correctedR(n-27 downto 1);
                when "00000000000000000000000000100000" => correctedR <=
"10000000000000001" & correctedR(n-28 downto 1);
                when "00000000000000000000000000010000" => correctedR <=
"10000000000000001" & correctedR(n-29 downto 1);
                when "00000000000000000000000000001000" => correctedR <=
"10000000000000001" & correctedR(n-30 downto 1);
                when "00000000000000000000000000000100" => correctedR <=
"10000000000000001" & correctedR(n-31 downto 1);
                when "00000000000000000000000000000010" => correctedR <=
"10000000000000000000000000000001";
                        when others  => correctedR <= '1' & correctedR(n-1 downto 1);
                END CASE;



                CASE countR1 IS  ----  these sequence adds to S
                        when "10000000000000000000000000000000" => correctedS <= "00"
& correctedS(n-2 downto 1);
                        when "01000000000000000000000000000000" => correctedS <=
"000" & correctedS(n-3 downto 1);
                        when "00100000000000000000000000000000" => correctedS <=
"1010" & correctedS(n-4 downto 1);
                    when "00010000000000000000000000000000" => correctedS <= "10100" &
correctedS(n-5 downto 1);
                when "00001000000000000000000000000000" => correctedS <= "100100" &
correctedS(n-6 downto 1);
                        when "00000100000000000000000000000000" => correctedS <=
"1010000" & correctedS(n-7 downto 1);
                        when "00000010000000000000000000000000" => correctedS <=
"100010000" & correctedS(n-9 downto 1);
                    when "00000001000000000000000000000000" => correctedS <=
"100010000" & correctedS(n-9 downto 1);
```

```
                                    when "00000000100000000000000000000000" => correctedS <=
"1000010000" & correctedS(n-10 downto 1);
                                when "00000000010000000000000000000000" => correctedS <=
"10000100000" & correctedS(n-11 downto 1);
                                    when "00000000001000000000000000000000" => correctedS <=
"100000100000" & correctedS(n-12 downto 1);
                                    when "00000000000100000000000000000000" => correctedS <=
"1000001000000" & correctedS(n-13 downto 1);
                                    when "00000000000010000000000000000000" => correctedS <=
"10000001000000" & correctedS(n-14 downto 1);
                                when "00000000000001000000000000000000" => correctedS <=
"100000010000000" & correctedS(n-15 downto 1);
                                when "00000000000000100000000000000000" => correctedS <=
"1000000010000000" & correctedS(n-16 downto 1);
                                when "00000000000000010000000000000000" => correctedS <=
"10000000100000000" & correctedS(n-17 downto 1);
                                when "00000000000000001000000000000000" => correctedS <=
"100000000100000000" & correctedS(n-18 downto 1);
                                when "00000000000000000100000000000000" => correctedS <=
"1000000001000000000" & correctedS(n-19 downto 1);
                                when "00000000000000000010000000000000" => correctedS <=
"1000000000100000000" & correctedS(n-20 downto 1);
                                when "00000000000000000001000000000000" => correctedS <=
"10000000000010000000000" & correctedS(n-21 downto 1);
                                when "00000000000000000000100000000000" => correctedS <=
"10000000000010000000000" & correctedS(n-22 downto 1);
                                when "00000000000000000000010000000000" => correctedS <=
"10000000000100000000000" & correctedS(n-23 downto 1);
                                when "00000000000000000000001000000000" => correctedS <=
"10000000000010000000000" & correctedS(n-24 downto 1);
                                when "00000000000000000000000100000000" => correctedS <=
"10000000000001000000000000" & correctedS(n-25 downto 1);
                                when "00000000000000000000000010000000" => correctedS <=
"10000000000010000000000000" & correctedS(n-26 downto 1);
                                when "00000000000000000000000001000000" => correctedS <=
"10000000000001000000000000" & correctedS(n-27 downto 1);
                                when "00000000000000000000000000100000" => correctedS <=
"10000000000000100000000000000" & correctedS(n-28 downto 1);
                                when "00000000000000000000000000010000" => correctedS <=
"10000000000000010000000000000" & correctedS(n-29 downto 1);
                                when "00000000000000000000000000001000" => correctedS <=
"10000000000000001000000000000" & correctedS(n-30 downto 1);
                                when "00000000000000000000000000000100" => correctedS <=
"10000000000000000100000000000000" & correctedS(n-31 downto 1);
          when "00000000000000000000000000000010" => correctedS <=
"10000000000000000010000000000000";

                                    when others  => correctedS <= '1' & correctedS(n-1 downto 1);
                            END CASE;

                                    correctedR <= '0' & correctedR(n-1 downto 1);


                    else
                correctedS <= regS & correctedS(n-1 downto 1);
                            correctedR <= regR & correctedR(n-1 downto 1);

            END IF;
end process;
```

```
        Scor            <= correctedS(0);
        Rcor            <= correctedR(0);

-----------------------------------

process
begin
wait until clk='0' and clk'event;

Scorr<=Scor;

end process;

DiPPMs<= '1' when Scor='1' and Scorr='0' else
   '0';
S1<= '1' when Scor='1' and Scorr='0' else
   '0';

process
begin
wait until clk='0' and clk'event;

Rcorr<=Rcor;

end process;

 Rcor2<= '1' when Rcor='1' and Rcorr='1' else
   '0';

process
begin
wait until clk='1' and clk'event;

Rcorr2<=Rcor2;

end process;

DiPPMr<= '1' when Rcorr2='0' and Rcorr='1' else
    '0';
R1<= '1' when Rcorr2='0' and Rcorr='1' else
    '0';

DiPPM1<= DiPPMs or DiPPMr;



end func;
```

E2

```
----VHDL DiPPM MLSD corrector.


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
```

```
use ieee.std_logic_unsigned.all;

entity MLSDcorrectorSR is
 generic (n: natural:= 32);
port(	di			:	in std_logic;
		rst			:	in std_logic;
		clk			:	in std_logic;
		R2			:	out std_logic;
		S2			:	out std_logic;
		DiPPM2		:	out std_logic);


end MLSDcorrectorSR;
architecture func of MLSDcorrectorSR is



   SIGNAL dipR	: std_logic_vector(1 downto 0);

   signal FlagS		: std_logic;
   signal FlagR		: std_logic;

   SIGNAL regS		: std_logic;
   SIGNAL regR		: std_logic;
	SIGNAL stateS		: std_logic;
	SIGNAL stateR		: std_logic;
	SIGNAL errorS		: std_logic;
	SIGNAL errorR		: std_logic;
	SIGNAL wrongslotS	: std_logic;
	SIGNAL wrongslotR	: std_logic;
	SIGNAL erasureS		: std_logic;
	SIGNAL erasureR		: std_logic;
	SIGNAL countS1			: std_logic_vector(n-1 downto 0);
	SIGNAL countS2			: std_logic_vector(n-1 downto 0);
	SIGNAL countR1			: std_logic_vector(n-1 downto 0);
	SIGNAL countR2			: std_logic_vector(n-1 downto 0);
	SIGNAL correctedS	: std_logic_vector(n-1 downto 0);
	SIGNAL correctedR	: std_logic_vector(n-1 downto 0);
   SIGNAL Rcor			: std_logic;
   SIGNAL Rcor2			: std_logic;
   SIGNAL Scor			: std_logic;
   SIGNAL Rcorr		: std_logic;
   SIGNAL Rcorr2		: std_logic;
   SIGNAL Scorr		: std_logic;
   SIGNAL DiPPMr		: std_logic;
   SIGNAL DiPPMs		: std_logic;

   CONSTANT zeros	: std_logic_vector:= "00000000000000000000000000000000";
   CONSTANT load1end	: std_logic_vector:= "00000000000000000000000000000001";
	CONSTANT load1begin : std_logic_vector:= "10000000000000000000000000000000";

BEGIN

PROCESS  ---(dipR)
BEGIN
   WAIT UNTIL (clk'EVENT AND clk='0');
   if (rst ='0') THEN
	dipR <= di & dipR(0);
	end if;
```

```
end process;



PROCESS
BEGIN

        WAIT UNTIL (clk'EVENT AND clk='1');
        IF (rst ='1') THEN
                regS            <= '0';
                stateS          <= '0';
          errorS                <= '0';

          countS1       <= zeros;
          countS2       <= zeros;
                wrongslotS      <= '0';
                erasureS <= '0';
                FlagS           <= '0';

        ELSIF (rst ='0') THEN
--A1: Normal operation: SET detected after RESET

       if Di='1' and dipR(1)='0'and stateS='0' then

                                stateS          <= '1';
                                errorS          <= '0';
                                erasureS <= '0';
                                wrongslotS      <= '0';
                                countS1 <= load1end;
                                countS2 <= load1end;
                                regS            <= di;
                                FlagS           <= '0';


--A2: Normal operation: SET detected 'straight' after RESET

                        elsIF Di='1' and dipR(1)='1' and stateR = '1' AND errorS='0'  THEN
                          stateS        <= '1';
                          errorS                <= '0';
                           wrongslotS   <= '0';
                          erasureS      <= '0';
                                countS1 <= load1end;
                                countS2 <= load1end;
                          regS          <= di;
                          FlagS         <= '0';


---B: Second SET detected: SET followed by a SET
                        elsIF   Di='1' AND stateS = '1' AND errorS= '0'and flags='0' THEN
          errorS        <= '1';
          countS1       <= countS1(0) & countS1(n-1 downto 1);
                                regS            <= di;
                                 FlagS          <= '0';

--C:Erasure-False Alarm
        ELSIF   stateS = '1' AND errorS= '1'and flagS='0' then
          erasureS      <= '1';
                                countS1 <= countS1(0) & countS1(n-1 downto 1);     ---nessesary
                                countS2 <= countS2(0) & countS2(n-1 downto 1);
```

```
                                    stateS          <= '0';    ----
                                    errorS          <= '0';
                                    regS            <= di;
                                    FlagS           <= '1';




--F: Normal operation: RESET detected after a single SET
                            ELSIF   stateR = '1' AND errorS='0'  THEN

                                stateS          <= '0';
                                errorS                  <= '0';
                                 wrongslotS     <= '0';
                                erasureS        <= '0';
                                        countS1 <= zeros;
                                countS2         <= zeros;
                                regS            <='0';
                                FlagS           <= '0';




--G: Normal operation: covers all other situations
                            ELSE
                                regS            <= '0';
                                        erasureS <= '0';
                                        wrongslotS      <= '0';
                                countS1         <= countS1(0) & countS1(n-1 downto 1);
                                                IF errorS =  '1' THEN
                                                        countS2 <= countS2(0) & countS2(n-1 downto 1);
                                                ELSE
                                                  errorS <= '0';
                                        END IF;
                            END IF;
                    END IF;

            end process;

            PROCESS
            BEGIN
            WAIT UNTIL (clk'EVENT AND clk='0');
IF rst ='1' THEN
                        regR            <= '0';
                        stateR          <= '0';
                  errorR                <= '0';

                        countR1         <= zeros;
                  countR2         <= zeros;
                        wrongslotR      <= '0';
                        erasureR<= '0';
                        FlagR       <='0';

                        ELSIF (rst ='0') THEN
--A: Normal operation: RESET detected after SET

            if  Di='1'  and stateR='0'   then
                                    stateR          <= '1';
```

```
                        errorR          <= '0';
                        erasureR<= '0';
                        wrongslotR      <= '0';
                        countR1         <= load1end;
                        countR2         <= load1end;
                        FlagR           <= '0';
                regR            <= di;


--B: Second RESET detected: RESET followed by a RESET
            ELSIF  Di='1'AND stateR = '1' AND errorR = '0'and FlagR='0' THEN
                errorR                  <= '1';
                countR1         <= countR1(0) & countR1(n-1 downto 1);
                regR            <= di;
                FlagR      <='0';



--C:Erasure-False Alarm
          ELSIF   stateR = '1' AND errorR= '1'and flagR='0' then
           erasureR         <= '1';
                        countR1         <= countR1(0) & countR1(n-1 downto 1);    ---
nessesary
                        countR2         <= countR2(0) & countR2(n-1 downto 1);
                        stateR          <= '0';    ----
                        errorR          <= '0';
                        regR            <= di;
                        FlagR           <= '1';



--F: Normal operation: SET detected after a single RESET
                ELSIF    stateS = '1' AND errorR='0'  THEN

                   stateR       <= '0';
                   errorR               <= '0';
                   wrongslotR   <= '0';
                   erasureR     <= '0';
                        countR1         <= zeros;
                   regR         <='0';
                   FlagR        <= '0';


--G: Normal operation: covers all other situations
                ELSE
                   regR         <= '0';
                   erasureR     <= '0';
                   wrongslotR   <= '0';
                   countR1      <= countR1(0) & countR1(n-1 downto 1);
                                IF errorR ='1' THEN
                                        countR2 <= countR2(0) & countR2(n-1 downto 1);
                                ELSE
                                        errorR    <= '0';
                END IF;
                END IF;
        END IF;

   end process;
```

```
PROCESS
BEGIN
WAIT UNTIL (clk'EVENT AND clk='1');

        IF (erasureS ='1' ) THEN      ---these sequence delete from S
          CASE countS1 IS
                when "10000000000000000000000000000000" => correctedS <= "01"
& correctedS(n-2 downto 1);
                when "01000000000000000000000000000000" => correctedS <=
"010" & correctedS(n-3 downto 1);
                when "00100000000000000000000000000000" => correctedS <=
"0100" & correctedS(n-4 downto 1);
                when "00010000000000000000000000000000" => correctedS <=
"01000" & correctedS(n-5 downto 1);
            when "00001000000000000000000000000000" => correctedS <= "010000" &
correctedS(n-6 downto 1);
                when "00000100000000000000000000000000" => correctedS <=
"0100000" & correctedS(n-7 downto 1);
                when "00000010000000000000000000000000" => correctedS <=
"01000000" & correctedS(n-8 downto 1);
                when "00000001000000000000000000000000" => correctedS <=
"010000000" & correctedS(n-9 downto 1);
                when "00000000100000000000000000000000" => correctedS <=
"0100000000" & correctedS(n-10 downto 1);
                when "00000000010000000000000000000000" => correctedS <=
"01000000000" & correctedS(n-11 downto 1);
                when "00000000001000000000000000000000" => correctedS <=
"010000000000" & correctedS(n-12 downto 1);
                when "00000000000100000000000000000000" => correctedS <=
"0100000000000" & correctedS(n-13 downto 1);
                when "00000000000010000000000000000000" => correctedS <=
"01000000000000" & correctedS(n-14 downto 1);
            when "00000000000001000000000000000000" => correctedS <=
"010000000000000" & correctedS(n-15 downto 1);
            when "00000000000000100000000000000000" => correctedS <=
"0100000000000000" & correctedS(n-16 downto 1);
            when "00000000000000010000000000000000" => correctedS <=
"01000000000000000" & correctedS(n-17 downto 1);
            when "00000000000000001000000000000000" => correctedS <=
"01000000000000000" & correctedS(n-18 downto 1);
            when "00000000000000000100000000000000" => correctedS <=
"01000000000000000" & correctedS(n-19 downto 1);
            when "00000000000000000010000000000000" => correctedS <=
"01000000000000000" & correctedS(n-20 downto 1);
            when "00000000000000000001000000000000" => correctedS <=
"01000000000000000" & correctedS(n-21 downto 1);
            when "00000000000000000000100000000000" => correctedS <=
"01000000000000000" & correctedS(n-22 downto 1);
            when "00000000000000000000010000000000" => correctedS <=
"01000000000000000" & correctedS(n-23 downto 1);
            when "00000000000000000000001000000000" => correctedS <=
"01000000000000000" & correctedS(n-24 downto 1);
            when "00000000000000000000000100000000" => correctedS <=
"01000000000000000" & correctedS(n-25 downto 1);
            when "00000000000000000000000010000000" => correctedS <=
"01000000000000000" & correctedS(n-26 downto 1);
            when "00000000000000000000000001000000" => correctedS <=
"01000000000000000" & correctedS(n-27 downto 1);
```

```
                when "00000000000000000000000000100000" => correctedS <=
"01000000000000000000000000000" & correctedS(n-28 downto 1);
                when "00000000000000000000000000010000" => correctedS <=
"01000000000000000000000000000" & correctedS(n-29 downto 1);
                when "00000000000000000000000000001000" => correctedS <=
"01000000000000000000000000000" & correctedS(n-30 downto 1);
                when "00000000000000000000000000000100" => correctedS <=
"01000000000000000000000000000" & correctedS(n-31 downto 1);
                when "00000000000000000000000000000010" => correctedS <=
"01000000000000000000000000000";
                        when others  => correctedS <= '1' & correctedS(n-1 downto 1);
            END CASE;

            CASE countS1 IS  ----  these sequence adds to S
                when "10000000000000000000000000000000" => correctedR <= "0"
& correctedR(n-1 downto 1);
                when "01000000000000000000000000000000" => correctedR <= "0"
& correctedR(n-1 downto 1);
                when "00100000000000000000000000000000" => correctedR <= "0"
& correctedR(n-1 downto 1);
              when "00010000000000000000000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                when "00001000000000000000000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                when "00000100000000000000000000000000" => correctedR <= "0"
& correctedR(n-1 downto 1);
                when "00000010000000000000000000000000" => correctedR <= "0"
& correctedR(n-1 downto 1);
              when "00000001000000000000000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                when "00000000100000000000000000000000" => correctedR <= "0"
& correctedR(n-1 downto 1);
              when "00000000010000000000000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                when "00000000001000000000000000000000" => correctedR <= "0"
& correctedR(n-1 downto 1);
                when "00000000000100000000000000000000" => correctedR <= "0"
& correctedR(n-1 downto 1);
                when "00000000000010000000000000000000" => correctedR <= "0"
& correctedR(n-1 downto 1);
              when "00000000000001000000000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
              when "00000000000000100000000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
              when "00000000000000010000000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
              when "00000000000000001000000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
              when "00000000000000000100000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
              when "00000000000000000010000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
              when "00000000000000000001000000000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
```

```
                    when "00000000000000000000000100000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                    when "00000000000000000000000010000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                    when "00000000000000000000000001000000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                    when "00000000000000000000000000100000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                    when "00000000000000000000000000010000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                    when "00000000000000000000000000001000" => correctedR <= "0" &
correctedR(n-1 downto 1);
                    when "00000000000000000000000000000100" => correctedR <= "0" &
correctedR(n-1 downto 1);
                    when "00000000000000000000000000000010" => correctedR <= "0" &
correctedR(n-1 downto 1);
          when "00000000000000000000000000000001" => correctedR <=
"00000000000000000000000000000000";

                        when others   => correctedS <= '0' & correctedS(n-1 downto 1);
                    END CASE;



                    elsIF (erasureR ='1' ) THEN      ---these sequence delete from R

                        CASE countR1 IS  ----  these sequence adds to S
                        when "10000000000000000000000000000000" => correctedS <= "0"
& correctedS(n-1 downto 1);
                        when "01000000000000000000000000000000" => correctedS <= "0"
& correctedS(n-1 downto 1);
                        when "00100000000000000000000000000000" => correctedS <= "0"
& correctedS(n-1 downto 1);
                    when "00010000000000000000000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                when "00001000000000000000000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000100000000000000000000000000" => correctedS <= "0"
& correctedS(n-1 downto 1);
                        when "00000010000000000000000000000000" => correctedS <= "0"
& correctedS(n-1 downto 1);
                    when "00000001000000000000000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000100000000000000000000000" => correctedS <= "0"
& correctedS(n-1 downto 1);
                    when "00000000010000000000000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000001000000000000000000000" => correctedS <= "0"
& correctedS(n-1 downto 1);
                        when "00000000000100000000000000000000" => correctedS <= "0"
& correctedS(n-1 downto 1);
                        when "00000000000010000000000000000000" => correctedS <= "0"
& correctedS(n-1 downto 1);
                    when "00000000000001000000000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                    when "00000000000000100000000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
```

```
                        when "00000000000000001000000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000100000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000010000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000001000000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000100000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000010000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000001000000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000000100000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000000010000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000000001000000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000000000100000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000000000010000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000000000001000" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000000000000100" => correctedS <= "0" &
correctedS(n-1 downto 1);
                        when "00000000000000000000000000000010" => correctedS <= "0" &
correctedS(n-1 downto 1);
          when "00000000000000000000000000000001" => correctedS <=
"00000000000000000000000000000000";

                              when others  => correctedS <= '0' & correctedS(n-1 downto 1);
                        END CASE;



                    CASE countR1 IS
                              when "10000000000000000000000000000000" => correctedR <=
"010" & correctedR(n-3 downto 1);
                              when "01000000000000000000000000000000" => correctedR <=
"0100" & correctedR(n-4 downto 1);
                              when "00100000000000000000000000000000" => correctedR <=
"01000" & correctedR(n-5 downto 1);
                              when "00010000000000000000000000000000" => correctedR <=
"010000" & correctedR(n-6 downto 1);
                    when "00001000000000000000000000000000" => correctedR <= "0100000" &
correctedR(n-7 downto 1);
                        when "00000100000000000000000000000000" => correctedR <=
"01000000" & correctedR(n-8 downto 1);
                              when "00000010000000000000000000000000" => correctedR <=
"010000000" & correctedR(n-9 downto 1);
                              when "00000001000000000000000000000000" => correctedR <=
"0100000000" & correctedR(n-10 downto 1);
                              when "00000000100000000000000000000000" => correctedR <=
"01000000000" & correctedR(n-11 downto 1);
                        when "00000000010000000000000000000000" => correctedR <=
"010000000000" & correctedR(n-12 downto 1);
```

```
                                    when "00000000001000000000000000000000" => correctedR <=
"0100000000000" & correctedR(n-13 downto 1);
                                    when "00000000001000000000000000000000" => correctedR <=
"01000000000000" & correctedR(n-14 downto 1);
                                    when "00000000000100000000000000000000" => correctedR <=
"010000000000000" & correctedR(n-15 downto 1);
                          when      "00000000000001000000000000000000"      =>      correctedR      <=
"0100000000000000" & correctedR(n-16 downto 1);
                          when      "00000000000000100000000000000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-17 downto 1);
                          when      "00000000000000010000000000000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-18 downto 1);
                          when      "00000000000000001000000000000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-19 downto 1);
                          when      "00000000000000000100000000000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-20 downto 1);
                          when      "00000000000000000010000000000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-21 downto 1);
                          when      "00000000000000000001000000000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-22 downto 1);
                          when      "00000000000000000000100000000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-23 downto 1);
                          when      "00000000000000000000010000000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-24 downto 1);
                          when      "00000000000000000000001000000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-25 downto 1);
                          when      "00000000000000000000000100000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-26 downto 1);
                          when      "00000000000000000000000010000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-27 downto 1);
                          when      "00000000000000000000000001000000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-28 downto 1);
                          when      "00000000000000000000000000100000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-29 downto 1);
                          when      "00000000000000000000000000010000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-30 downto 1);
                          when      "00000000000000000000000000001000"      =>      correctedR      <=
"01000000000000000" & correctedR(n-31 downto 1);
                          when      "00000000000000000000000000000100"      =>      correctedR      <=
"01000000000000000";
                                    when others   => correctedR <= '1' & correctedR(n-1 downto 1);
                          END CASE;



                ELSE
              correctedS <= regS & correctedS(n-1 downto 1);
                          correctedR <= regR & correctedR(n-1 downto 1);

        END IF;
end process;
        Scor            <= correctedS(0);
        Rcor            <= correctedR(0);

---------------------------------

process
begin
wait until clk='0' and clk'event;
```

```
Scorr<=Scor;

end process;

DiPPMs<= '1' when Scor='1' and Scorr='0' else
   '0';
S2<= '1' when Scor='1' and Scorr='0' else
   '0';

process
begin
wait until clk='0' and clk'event;

Rcorr<=Rcor;

end process;

 Rcor2<= '1' when Rcor='1' and Rcorr='1' else
   '0';

process
begin
wait until clk='1' and clk'event;

Rcorr2<=Rcor2;

end process;

DiPPMr<= '1' when Rcorr2='0' and Rcorr='1' else
    '0';
R2<= '1' when Rcorr2='0' and Rcorr='1' else
    '0';

DiPPM2<= DiPPMs or DiPPMr;


end func;
```

F1

```
----VHDL DiPPM coder synchronous output.


Library IEEE;
use IEEE.STD_LOGIC_1164.all;


entity DiPPMcodery is
            port(
      CLK : in std_logic;
      PCM : in std_logic;
      Rst : in std_logic;
      DiPPM:out std_logic

  );
end DiPPMcodery;
```

```
--}} End of automatically maintained section

architecture beh of DiPPMcodery is

signal DiPPMsup:std_logic;
signal DiPPMsdown:std_logic;

signal DiPPMrup:std_logic;
signal DiPPMrdown:std_logic;
signal DiPPMr:std_logic;
signal DiPPMrr:std_logic;
signal DiPPMrrr:std_logic;
Signal R:std_logic;
Signal S:std_logic;
signal DiPPMset:std_logic;
signal DiPPMreset:std_logic;


begin
process
begin
wait until clk'event and clk='1';
if PCM='1'and clk='1' then
DiPPMsup<='1';
else
DiPPMsup<='0';
end if;
end process;

process
begin
wait until clk'event and clk='0';
if PCM='1'and clk='0' then
DiPPMsdown<='1';
else
DiPPMsdown<='0';
end if;
end process;

DiPPMset<='1' when DiPPMsup='1' and DiPPMsdown='0' else
        '0';


process
begin
wait until clk'event and clk='0';
if PCM='0'and clk='0' then
DiPPMrup<='1';
else
DiPPMrup<='0';
end if;
end process;

process
begin
wait until clk'event and clk='1';
if DiPPMrup='1'and clk='1' then
DiPPMrdown<='1';
```

```
else
DiPPMrdown<='0';
end if;
end process;

DiPPMreset<='1' when DiPPMrup='1' and DiPPMrdown='0' else
          '0';


DiPPM<= DiPPMset xor DiPPMreset;
--DiPPM<= '1' when S='1' and R='0'else
    --  '1' when S='0' and R='1'else
      -- '0';


end beh;
```

F2

```
----VHDL Synchroniser.


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity DFFs is
 generic (n: natural:= 32);
port(      DiPPM                    :    in std_logic;
                  rst                                    :    in std_logic;
           clk                                  :    in std_logic;
           S         : out std_logic;
      R         : out std_logic;
      Clock_out         : out std_logic;
           DiPPM1                   :    out std_logic);


end DFFs;
architecture func of DFFs is


   SIGNAL regIP1       : std_logic_vector(1 downto 0);
   SIGNAL regIP2       : std_logic_vector(1 downto 0);
   SIGNAL S1       : std_logic;
   SIGNAL R1       : std_logic;
   SIGNAL Sd       : std_logic;
   SIGNAL Rd       : std_logic;

 BEGIN

Clock_out<=clk;

process
 begin
  wait until (clk'event and clk='0');
  if (rst='1') then
```

```
  regIP1 <="00";
  elsif rst='0' then
regIP1(1 downto 0) <= DiPPM & regIP1(1);
  end if;
 end process;

 process
 begin
  wait until (clk'event and clk='1');
  if (rst='1') then
  regIP2 <="00";
  elsif rst='0' then
regIP2(1 downto 0) <= DiPPM & regIP2(1);
  end if;
 end process;

 S1<=regIP1(1);

 S<='1' when S1='1' and clk='1' else
    '0';
  Sd<='1' when S1='1' and clk='1' else
    '0';

 R1<=regIP2(1);

 R<='1' when R1='1' and clk='0' else
     '0';
  Rd<='1' when R1='1' and clk='0' else
     '0';

DiPPM1<='0' when Rd='0' and Sd='0' else
     '1';

 end func;
```

F3

```
----VHDL DiPPM decoder while its MLSD input is asynchronous.

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;


entity DiPPMdecoder_MLSD is
            port(
      CLK : in BIT;
      DiPPM: in BIT;
      Rst: in BIT;
               PCM_out:out bit


 );
end DiPPMdecoder_MLSD;
```

```
architecture beh of DiPPMdecoder_MLSD is


signal nclk:bit;
signal R:bit;
signal S:bit;


begin

nclk<= clk nor clk;


process
begin

wait until nclk='1' and nclk'event;


S<=DiPPM;

end process;

R<='1' when DiPPM='1' and clk='0' else
   '0';


process (S,R)
begin


if S='1' then
  PCM_out<='1';
  elsif R='1' then
PCM_out<='0';
end if;

end process;


end beh;
```

F4

```
----VHDL DiPPM error generator.

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity errorgen is
  generic (n1: natural:= 4;
           n2: natural:= 7);

port(    pcmip       :    in std_logic;
```

```
        clk             :   in std_logic;
        dippm           :       out std_logic;
        errortype       :   in std_logic_vector(2 downto 0));

end errorgen;

architecture func of errorgen is

  SIGNAL setop1    : std_logic;
  SIGNAL resetop1  : std_logic;
  SIGNAL set1      : std_logic;
  SIGNAL reset1    : std_logic;
  SIGNAL countS    : std_logic;
  SIGNAL countR    : std_logic;
  SIGNAL registerS : std_logic_vector (n1-1 downto 0);
  SIGNAL registerR : std_logic_vector (n1 downto 0);
  SIGNAL control   : std_logic_vector (n2-1 downto 0):= "1000000";
  SIGNAL dippmS    : std_logic;
  SIGNAL dippmR    : std_logic;


BEGIN

        PROCESS  -- creating a SET pulse whilst PCM = 1
        BEGIN
                WAIT UNTIL (clk'EVENT AND clk = '1');
                IF (pcmip ='1' and setop1 = '0' ) THEN
                        setop1          <= '1';

                ELSE
      setop1            <= '0';

    END IF;
  END PROCESS;

  PROCESS
        BEGIN
                WAIT UNTIL (clk'EVENT AND clk = '0');
                IF (pcmip = '0' and resetop1 = '0') THEN
                        resetop1        <= '1';
          ELSE
                        resetop1                <= '0';

    END IF;
  END PROCESS;


   PROCESS
        BEGIN
                WAIT UNTIL (clk'EVENT AND clk = '1');
                IF (setop1 = '1' and countS = '0') THEN
                        set1    <= '1';
                        countS  <= '1';
                        countR  <= '0';

                ELSIF (resetop1 = '1' and countR = '0') THEN
                        reset1  <= '1';
                        countR  <= '1';
                        countS  <= '0';
```

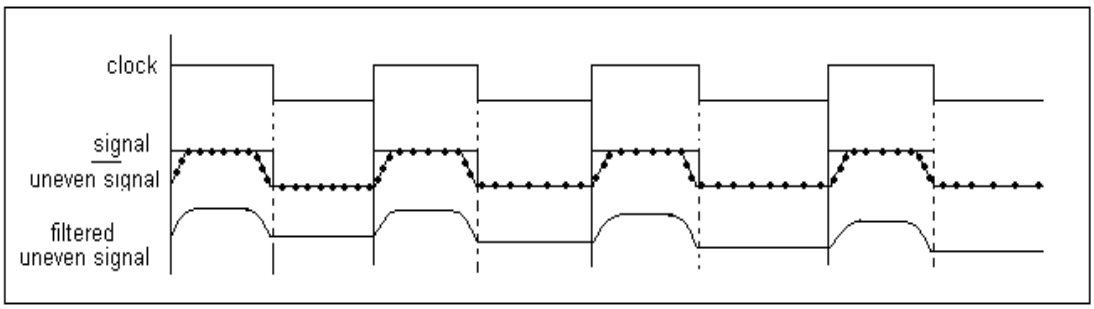
```
            ELSE
                        set1        <= '0';
        reset1      <= '0';
    END IF;

    registerS <= set1 & registerS(n1-1 downto 1);
    registerR <= reset1 & registerR(n1 downto 1);
    control <= control(0) & control(n2-1 downto 1);


    CASE errortype IS
                WHEN "001" =>
                                        IF control(0)= '1' THEN
            registerS(3) <= NOT (registerS(3));
            END IF;                 -- If the control reg size is not a multiple
                                -- set reg then the SET register either has an extra
                                -- set (1) or a SET is removed.
        WHEN "010" =>
                                        IF control(0)= '1' THEN
            registerR(2) <= NOT (registerR(2));
            END IF;                                                 -- If the control reg size is
not a multiple
                                -- reset reg then the RESET register either has an extra
                                -- reset (1) or a RESET is removed.
        WHEN "011" =>
                                        IF control(0)= '1' AND registerS(1) = '1' THEN
              registerS(0) <= '0';
              registerR(0) <= '1';
            END IF;                                             -- If the control reg(0)=1 and the
SET reg(1)=1 then SET is
                                -- changed to a RESET

        WHEN "100" =>
                                        IF control(0)= '1' AND registerR(1) = '1' THEN
              registerR(0) <= '0';
              registerS(0) <= '1';
            END IF;

        WHEN OTHERS => registerS(0) <= registerS(0);
                registerR(0) <= registerR(0);
      END CASE;

  END PROCESS;

 dippmS <= registerS(0);
 dippmR <= registerR(0);

 PROCESS (dippmS,dippmR,clk)
        BEGIN
        Dippm <= (dippmS AND clk) OR (dippmR AND NOT Clk);

  end process;


END func;
```

Appendix2

A

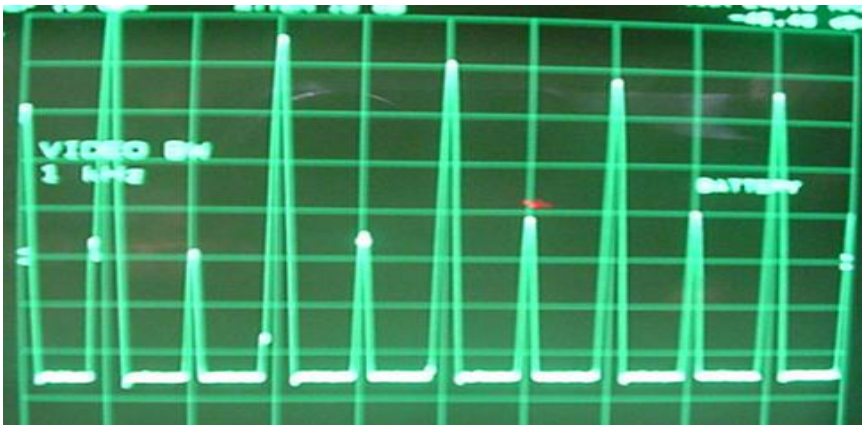
The window function, which is used by the periodogram command, is an equation that simulates the filter of the spectrum analyser in Matlab. This filter defines the shape (width of the pulses) of the sequence. The sequence produced by the signal generator at high frequencies (as 240 MHz in the experiments of this thesis) would not be square (rectangular window) as at low frequencies. The higher the speed, the smaller the pulse width and the greater the effect of finite rise and fall times on the pulses.

Tests were carried out using a PCM deterministic signal (101010) running at 120 MHz. The window response was obtained so that the Matlab simulation and the measured spectrum gave accurate correlation. The window was required because of the finite rise and fall times and the reduced mark:space ratio of the signal (fig. Appendix2-A1).

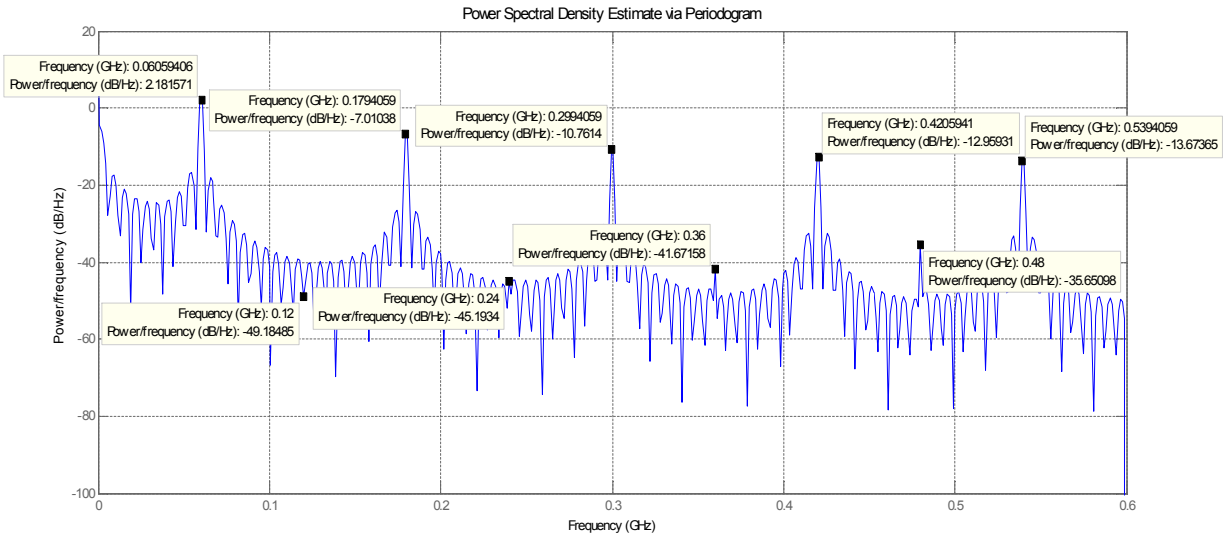


Appendix2-A1: (Up) clock, (middle) signal and uneven signal, (down) filtered uneven signal

Figure Appendix2-A2 shows the measured spectrum of the deterministic PCM signal. Figure Appendix2-A3 shows the Matlab result. The window expression is given in equation Appendix2-1A.



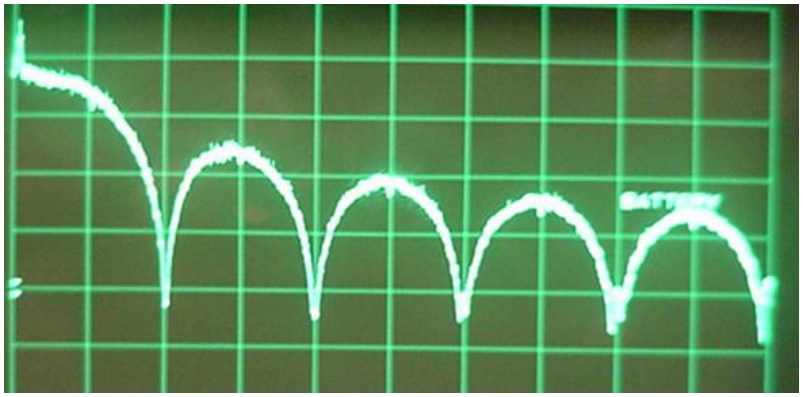
Appendix2-A2: *Deterministic PCM PSD from pulse generator.*



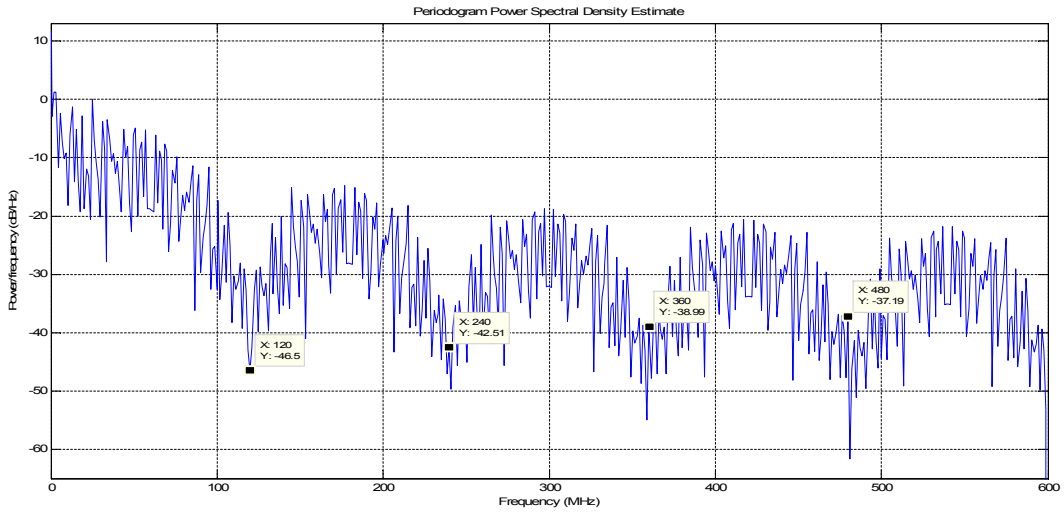
Appendix2-A3: *Deterministic PCM PSD from simulation.*

$$W = \frac{15}{1} * \frac{1}{38}\sin(2\pi ft) + \frac{1}{24}\sin(4\pi ft) - \frac{1}{16}\sin(6\pi ft) + \frac{1}{23}\sin(8\pi ft) \qquad \text{(Appendix2-1A)}$$

It was expected that the window equation, that simulates the shape of the pulses as a deterministic sequence, will simulate at a PRBS sequence as well. Figure Appendix2-A4 shows the measured spectrum of a PRBS PCM data stream and Appendix2-A5 shows the Matlab result.



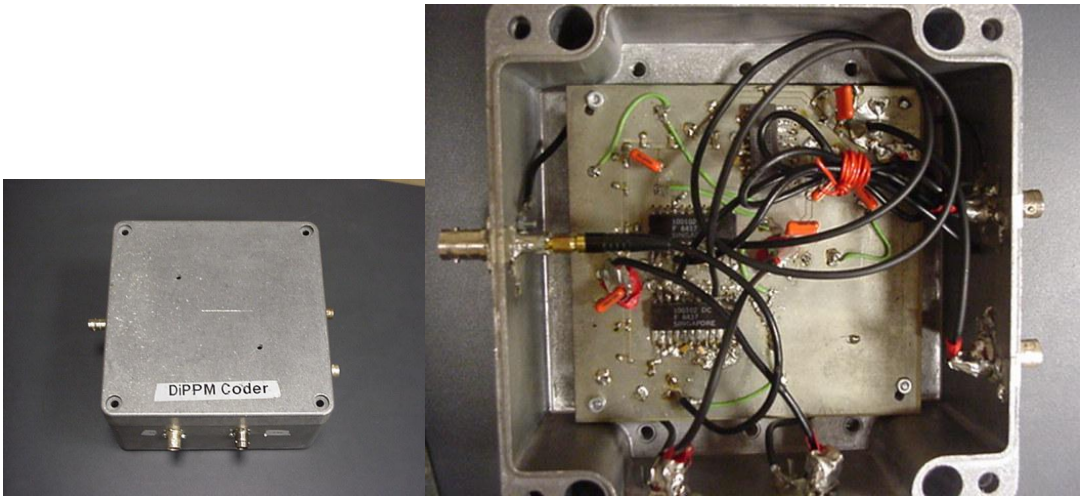
Appendix2-A4: PRBS*PCM PSD from pulse generator.*



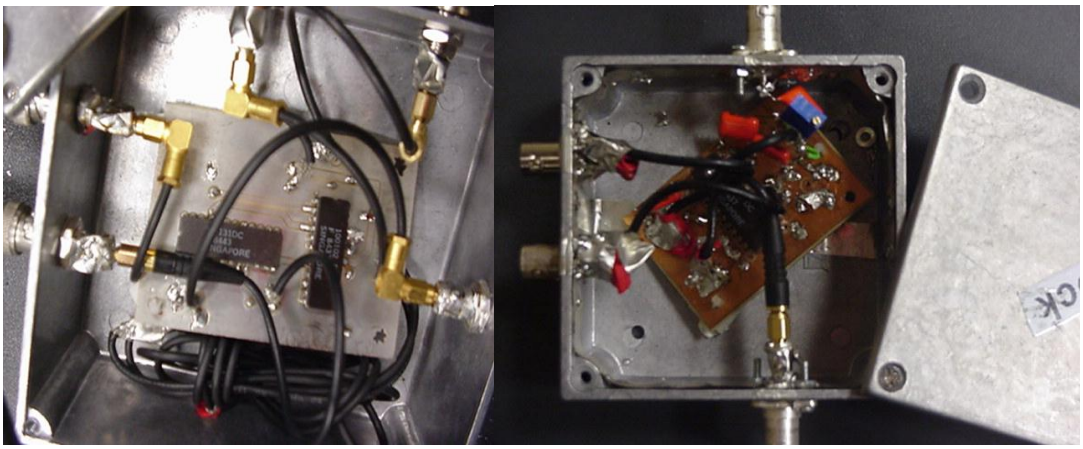
Appendix2-A5: *PRBSPCM PSD from simulation.*

For the DiPPM sequence, the SET and RESET had different pulse widths and so the window equation of AppendixA2-1A can not be used. Thus, a different window, found using the same process, must be used for DiPPM.

Appendix3

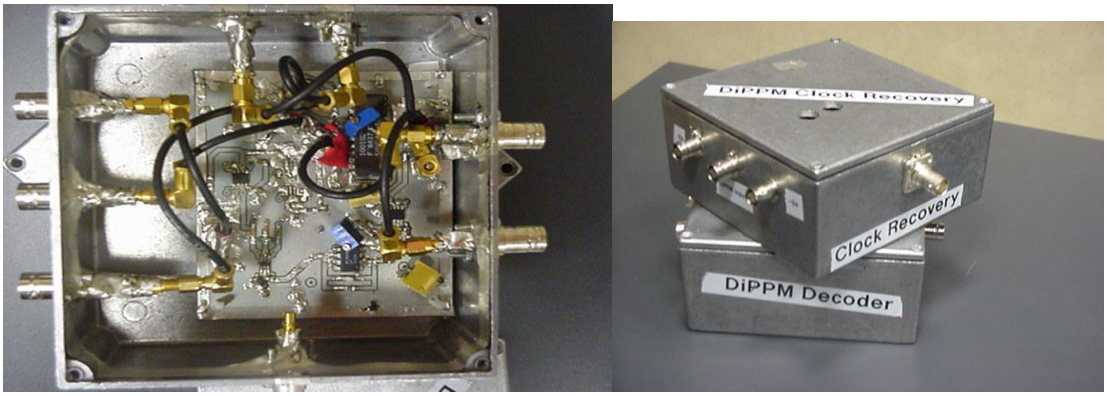


Photos A: *DiPPM Coder*



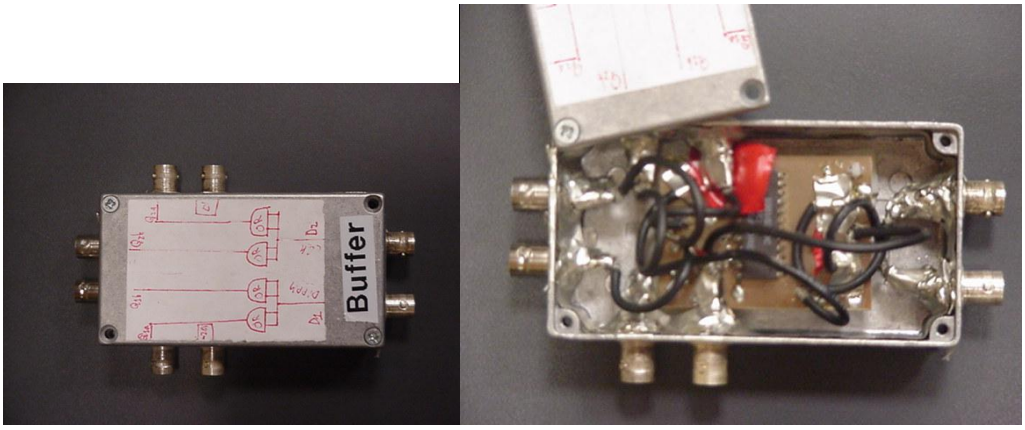
Photos B: *DiPPM Decoder*

Photos C: *Clock Buffer*

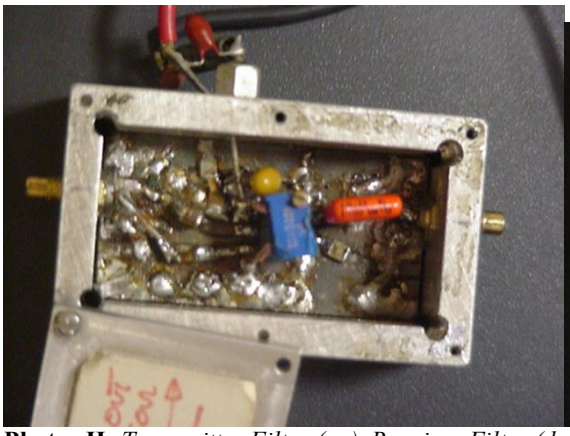


Photos D: *DiPPM Timing Extraction*

Photos E: *DiPPM Timing Extraction and Decoder*

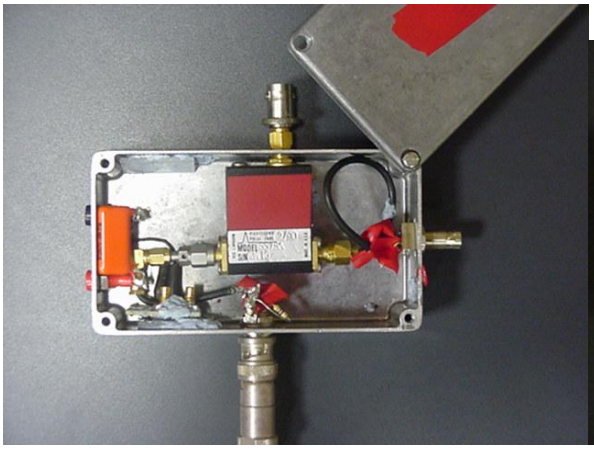


Photos F: *Buffer*

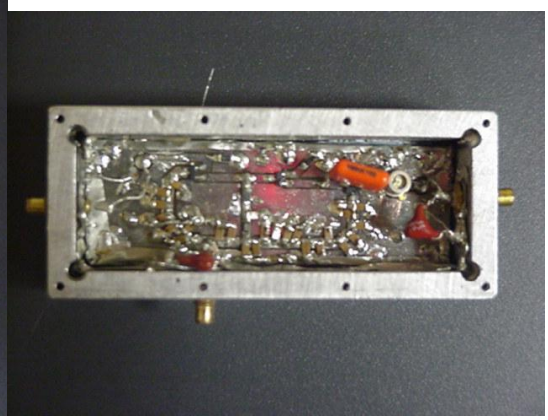


Photos H: *Transmitter Filter (up), Receiver Filter (down)*

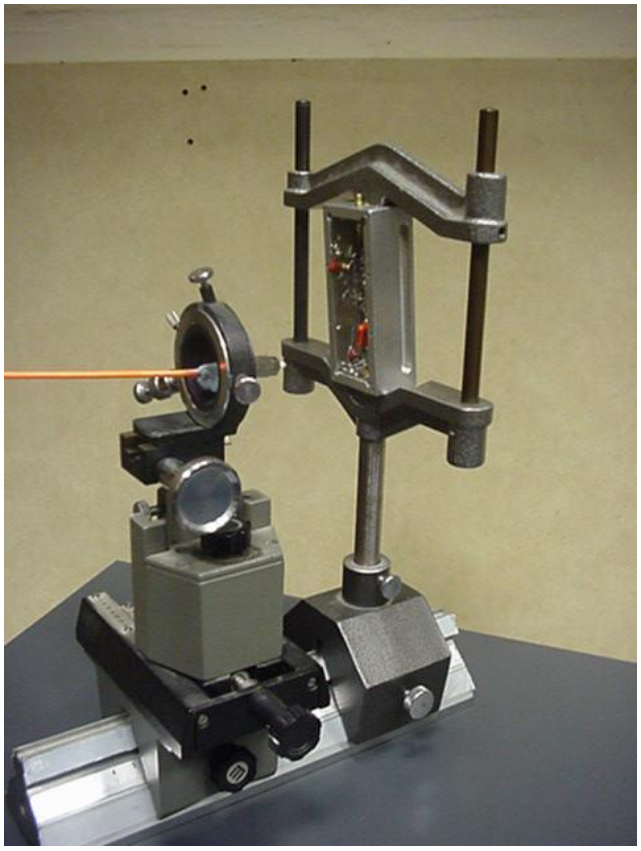
Photos G: *Clock Buffer*



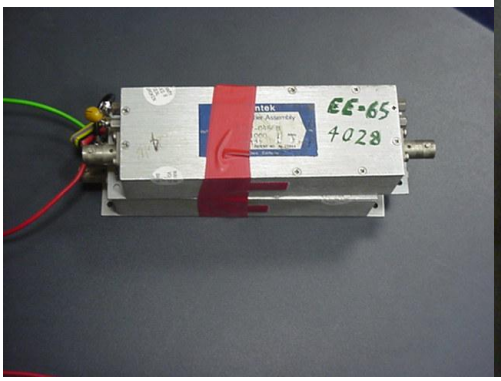
Photos L: *Si PIN pre-amplifier optical receiver*



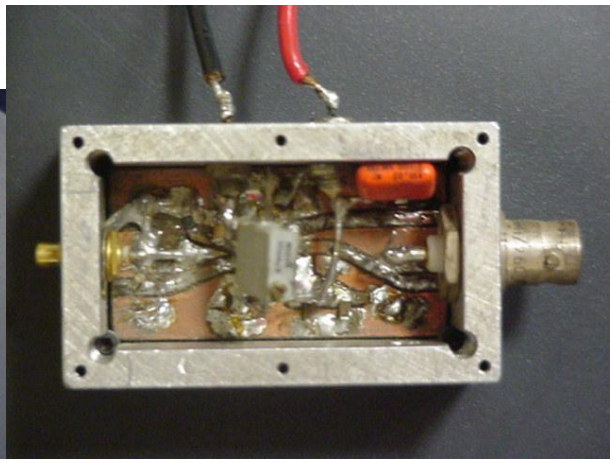
Photos J: *Emitter optic Transmitter*



Photos K: *Position of Optical Transmitter and Receiver*



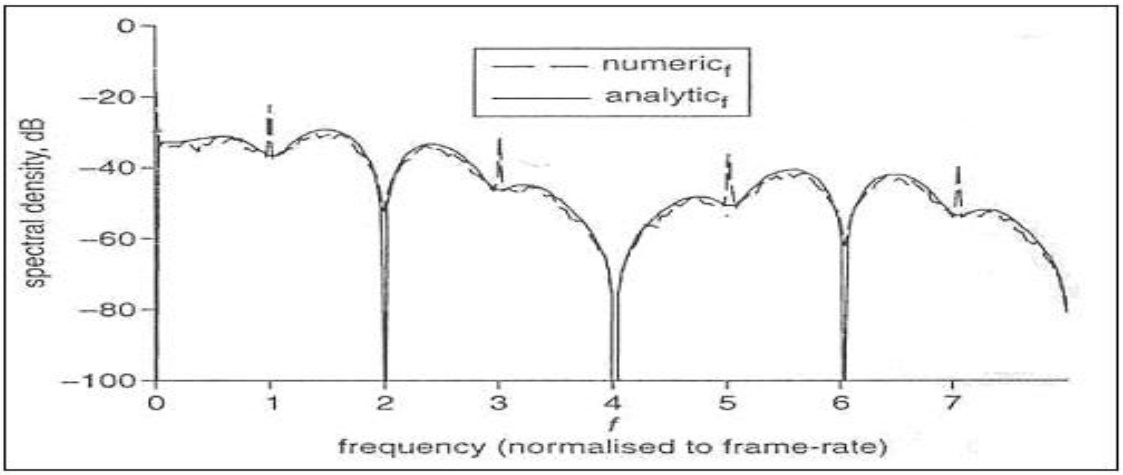
Photos N: *Invertid Amplifier*



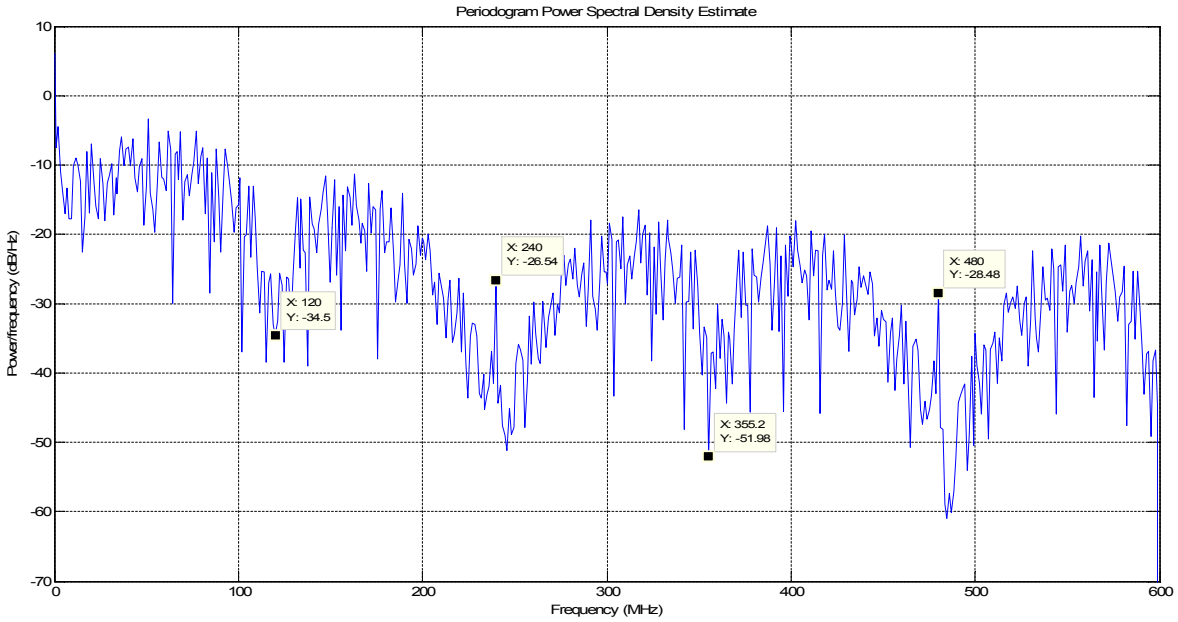
Photos M: *Threshold Detector*

Appendix4

Figure Appendix4-A1 shows the random DiPPM PSD of publication [108], while figure Appendix4-A2 shows the random DiPPM PSD that has been presented in this thesis.



Appendix4-A1: *Random DiPPM PSD from [108]*



Appendix4-A2: *Random DiPPM PSD that has been presented in this thesis.*

For the comparison of those random DiPPM PSDs that have been presented, the process and analysis of the commands that have been used to plot Appendix4-A2 have to be presented.

PSD of figure Appendix4-A2 uses the window equation and the sample frequency. The sample frequency normalizes the graph to the frame-rate. The command that plots the PSD is:

```
periodogram(dippm_signal_time*7500,w,'onesided',1010,Sampling_frequency);
```
(Appendix4-1)

The second element is the W (the window) and the last (5^{th}) the sampling frequency (the rest are not needed to be explained now, for more information see chapter 5).

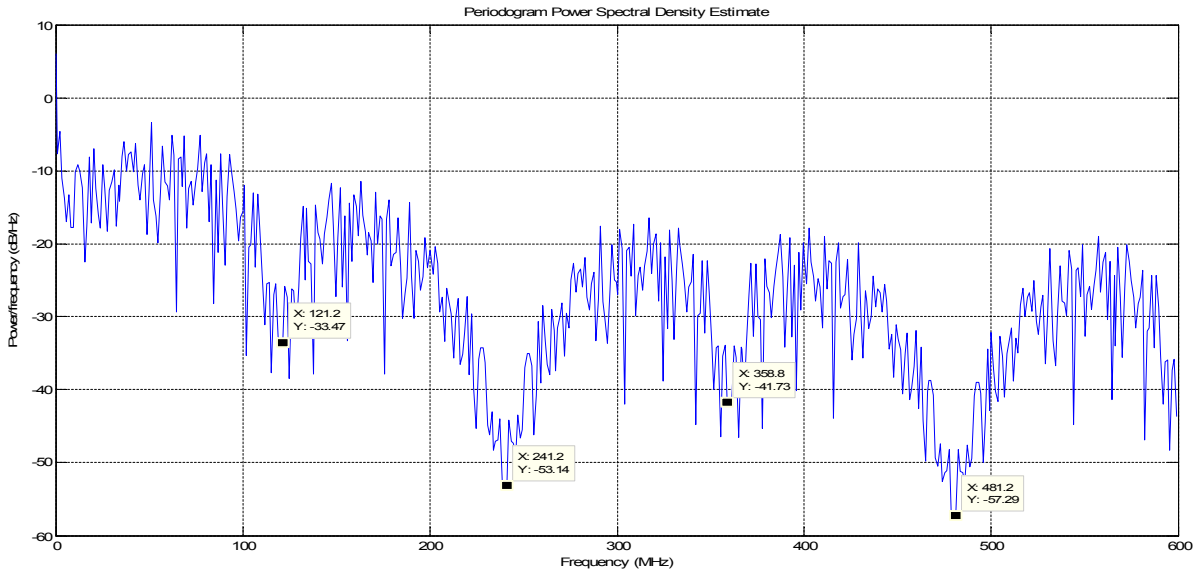
The window (W) is the equation that gives the width of the pulse. With W the window is used and with [] the window is not used, hence the pulse is rectangular.

The Sampling frequency, when it is used, gives frequencies along the X axis of the graph. If it is not used (its place is empty) then the X axis is normalised to frame-rate and X axis is 1,2,3…etc.

When the command (Appendix-4-1) does not use the window:

```
periodogram(dippm_signal_time*7500,[],'onesided',1010,Sampling_frequency);
```
(Appendix4-2)

then the plot of random DiPPM PSD is:



Appendix4-A3: *Random DiPPM PSD plot without the use of window (w).*

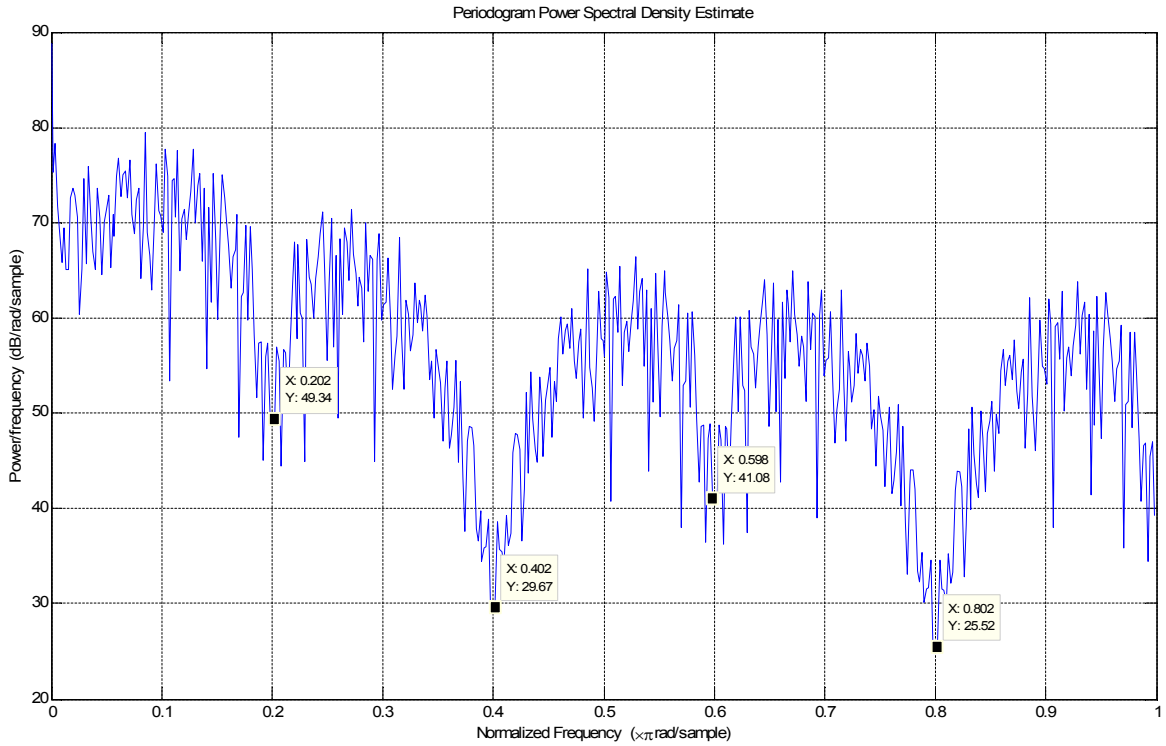
Hence, from figure Appendix4-A3 it can be seen that the powers (120,240,360….) reduced, as it was expected, because now the DiPPM pulses are rectangular giving a null at 1/pulse time. The PSD of Appendix4-A3 has similarities with the PSD of [108] but is not exactly the same.

If the sampling frequency from the command Appendix4-2 has been removed, the PSD graph will be normalised to frame-rate (1,2,3…) hence it will look same as that of [108].

The PSD command now is as:

```
periodogram(dippm_signal_time*7500,[],'onesided',1010);          (Appendix4-3)
```

Plotting Appendix4-3, which is window and sampling frequency not included, the PSD is:

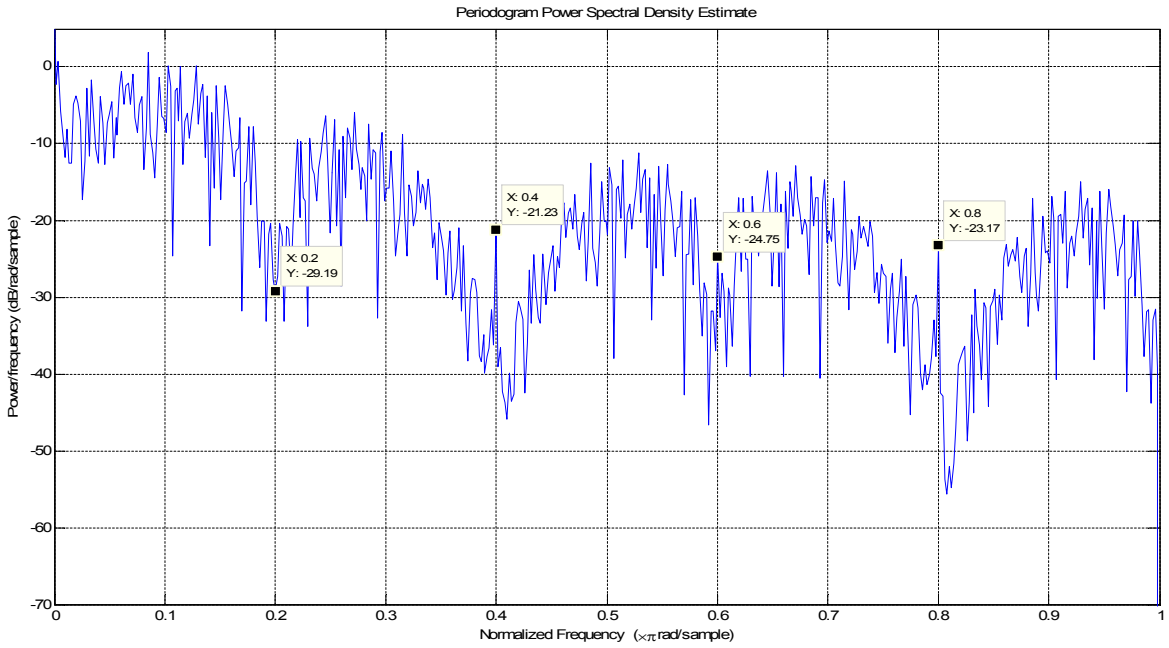


Appendix4-A4: *Random DiPPM PSD plot without the use of window (w) norand sampling frequency.*

DiPPM random PSD is shown normalised to frame-rate in figure Appendix4-A4. It is similar to that of [108]. The datatips, now, are not showing the frequency but the length of X axes. From comparison of A4 and A3, the powers (harmonics-120, 240, 360…) are at the place of 2,4,6,8….etc in figure Appendix4-A4. Hence the places 1,3,5…etc give the sub harmonics (where random gives and PRBS does not).

It should be mentioned that the PSD [108] produces the right PSD with random rectangular DiPPM but the powers have not presented. PSD of [108] shows just the sub-

harmonics (1,2,3,…etc). Finally, random DiPPM PSD with window but normalised to frame-rate, is:



Appendix4-A5: *Random DiPPM PSD plot without the use of window (w) normalised to frame-rate.*

Figure Appendix4-A5 shows the random DiPPM PSD with the experimental width and normalised to frame-rate.

REFERENCES

1. Sibley, M.J.N.: *"Dicode pulse-position modulation: a novel coding scheme for optical-fibre communications,"* IEE Proc., Optoelectron., 2003,150, (2), pp. 125-131.
2. Sibley, M.J.N.: *"Suboptimal filtering in zero-guard, Dicode PPM system operating over dispersive optical channels,"* IEE Proc., Optoelectron., 2004,151, (4), pp. 237-243.
3. Sibley, M.J.N.: *"Analysis of dicode pulse position modulation using a PINFET receiver and a slighty / highly dispersive optical channel,"* IEE Proc., Optoelectron., 2003,150, (3), pp. 205-209.
4. R.A. Cryan and Sibley, M.J.N.: *"Minimising intersymbol interference in optical-fibre Dicode PPM systems,"* IEE Proc., Optoelectron., 2006,153, (3), pp. 93-99.
5. Shannon, C.E.:*'A mathematical theory of communication',* Bell System Tech. Jour., vol.27, pp.379-423 and pp. 623-656, 1948.
6. Shannon, C.E., Weaver, W.: *''The mathematical theory of communication'*, Univ. Illinois Press, 1949.
7. Golay, M.J.E.: '*Note on the theoretical efficiency of information reception with PPM'*, PIRE, vol. 9, pp.1031, 1949.
8. Jacobs, J.R.A.: '*Theoretical and practical limitations of weak signal processing techniques',* Space Research 11, page 413, 1961.
9. Viterbi, A.: '*Classification and evaluation of coherent synchronous sampled late telemetry systems',* IRE Trans. On SET, No.1, page 13, 1962.
10. Gruenberg, E.L.: '*Handbook of telemetry and remote control',* McGraw Hill Book Company, pp.9-30, 1967.
11. Chang, J., Yingcai, C.: '*A review of thirty years developments and researchers on uncoded QPPM systems',* International Telemetry Conference: Int. Found. Telemetering, pp. 153-164, 1982.

12. Hubbard, W.M.: '*Utilisation of Optical-frequency Carrirers for Low and Moderate-Bandwidth Channels',* Bell Syst. Tech. J., vol 52, pp.731-765, 1973.

13. Holden, W.S.: *A Pulse-Position Modulation Experiment for Optical Fibre Transmission',*Bell Syst. Tech. j., vol 54, pp. 258-296, 1975.

14. Muoi, T.V., Hullett, J.L.: '*Receiver bandwidth for optical PPM systems',* IEEE Trans.Commun., vol. COM-26, pp. 295-300, 1978.

15. Luciano, B., Pirani, G.: '*A new signaling format for optical communications',* Electronics Letters, vol.14 No.3, pp.71-73, 1878.

16. S. Karp and R.M. Gagliardi, *'The design of a Pulse-position modilated Optical communication System',* IEEE, trans. Commun. Technol., vol. COM-17, pp. 670-676, 1969.

17. Borisov E.V.: '*Noise Immunity of Reception of Coded Messages in Optical Communication Lines',* Radio eng. Electron. Phys, vol.24, pp. 121-124, 1978.

18. Trusov A.G.: *'Estimation of the Optical Signal Arrival Time Under Conditions of Photon Counting in Free Space',* Telecomm. Radio Eng., pt. II, vol. 33, pp. 137-139, 1978.

19. Lee G.M. and Schroeder G.W.:*'Optical Pulse Position Modulation with Multiple Positions per Pulsewidth',* IEEE Trans. Commun. Vol. COM-25, pp. 360-365, 1977.

20. Gol'dsteyn Yu.A. and Frezinskiy B. Ya.:*An investigation of the transmission of a Multi-position PPM optical Signal Through a Communications Line Containing Repeaters',* Radio Eng. Electron. Phys., vol. 24, pp. 65-71, 1978.

21. Dolinar, S.: *'A near optimum receiver structure fot the detection of M-ary optical PPM signals',* Int. Found. Telemetering Conf, San Diego, page 145, 1982.

22. Pires J.J.O. and Prof. da Rocha J.R.F.:' *Digital Pulse Position Modulation over Optical Fibres with Avalanche Photo-diode Receivers',* IEE Proceedings, Vol 133, Pt. J, No 5, October 1986.

23. Mecherle, G.S.: *Detection alternatives for pulse position modulation (PPM) optical communication',* Optical Technologies for Communication Satellites, SPIE, vol. 616, pp. 105-116, 1986.

24. Mecherle, G.S.: '*Impact of laser diode performance on data rate capability of PPM optical communication',* IEEE Military Communication Conference: MILCOM 85, vol.1, pp. 115-121, 1985.

25. Garrett I.:' *Digital Pulse-position modulation for Transmission over Optical Fiber Channels with Direct and Heterodyne Detection',* IEEE Trans. Comm. Vol COM-31, pp.518-527, 1983.

26. Garrett I.: '*Digital Pulse-position Modulation over Dispersive Optical Fiber Channels',* International Workshop on Digital Communications, Tirrenia, Italy, Aug. 15-19, 1983.

27. N. Calvert: '*Digital Pulse Position Modulation',* PH.D. Thesis, C.N.A.A., 1988

28. M.N. Calvert, M.J.N. Sibley, R.T. Unwin: '*Experimental optical fibre digital pulse-position modulation system',* Electonics Letters, no. 24, pp. 129-131, 1988.

29. M.N. Calvert, M.J.N. Sibley, R.T. Unwin, I Garrett, R.A. Cryan: '*Optimal filtering of digital PPM transmitted over optical fibre channels employing PIN-BJT teceivers',* IEE Saraga Colloquim on Electronic Filters, Digest No. 1989/97, pp. 5/1-5, London, 1989.

30. I. Garrett, N.M. Calvert, M.J.N. Sibley, R.T. Unwin, R.A. Cryan: *Optical fibre digital pulse position modulation',* Br Telecom Technol. J, 7, no. 3, July 1989.

31. R.A. Cryan, R.T. Unwin, A.J. Massarella, M'J.N. Sibley: '*Performance analysis of a homodyne digital PPM system',* IEE 2nd Bangor Symposium on communications, pp.126-129, Bangor, England, 23-24 May 1990.

32. R.A. cryan, R.T. Unwin, A.J. Massarella, M.J.N. Sibley, I Garret: *'Coherent Detection: n-ary PPM vs PCM',* SPIE Proc. On Coherent Lightwave Communications. San Jose, California, 16-21 September, 1990.

33. A.J. Massarella, M.J.N. Sibley: '*Experimental results on suboptimal filtering for optical digital pulse-position modulation',* Electronics Letters, vol. 27, no. 21, pp. 1953-1955, 1991.

34. A.J. Massarella, M.J.N. Sibley: '*Optical digital pilse position modulation: Experemantal results for heterodyne detection using suboptimal',* Electronics Letters, vol. 28, no. 6, pp. 574-575, 1992.

35. Bennett W.R.: '*Statistic of regenerative digital transmission',* The Bell system technical J., pp. 1501-1542, Nov. 1958.

36. Takasaki Y.: '*Timing extraction in baseband pulse transmission',* IEEE Trans. On Comm., vol. COM-20, No 5, pp.877-883, Oct. 1972.

37. Franks L.E.: *'Carrier and bit synchronization in data communication – A tutorial review',* IEEE Trans. On Comm., vol. COM-28, No 8, pp. 1107-1121, Aug. 1980.

38. Gitlin R.D. Salz J.: *'Timing recovery in PAM systems',* The Bell system technical Jrl., pp. 1645-1649, May-June 1971.

39. Datta D. and Gengopadhyay R.: '*Simulation studies on nonlinear bit synchronizers in APD-based optical receivers',* IEEE Trans. On Comm. Vol. COM-35, No 9, pp. 909-917, Sept. 1987.

40. Rosenberg R.L., Chamzas C. and Fishman D.A.:' *Timing recovery with SAW transversal filters in the regenerators of undersea long-haul fibre transmission systems',* IEEE J. on selected areas in Comm., vol. SAC-2, No 6, pp. 957-65.

41. Gardner F.M.: '*Phaselock Techniques',* 2nd ed., New York: Wiley, 1979.

42. Gagliardi R.M.: '*Synchronisation using pulse edge tracking in optical pulse-position modulated communication systems',* IEEE Trans. On Comm., pp. 1693-1702, Oct. 1974.

43. Gol'dsteyn A. Yu. and Frezinskiy B. Ya.: '*Statistical estimation of the time of arrival of a sequence of optical PPM pulses',* Telecomm. And Radio Eng, Part 2, vol. 35, No. 10, pp. 120-121, 1980.

44. Gagliardi R.M..: '*Time synchronisation in optical PPM Sequences',* IEEE Globcom 83, vol. 2. pp. 779-783, 1983.

45. Ling R. and Gagliardi R.M., '*Slot synchronisation in optical PPM communications',* IEEE Trans. On Comm., vol. COM-34, No. 12, pp 1202-1208, Dec.1986.

46. Gagliardi, R.M. and Ger Ling: *Slot synchronization in Optical PPM Communications',* IEEE Trans. On comm.., vol. COM-34, No. 12, Dec., 1986.

47. Chen C.C., Win M.Z., Marshall W.K. and Lesh J.R.: '*Low data rate coherent optical link demonstration using frequency stabilized solid state laser',* Free-Space Laser Comm. Techn. III, SPIE, vol. 1417, pp. 170-181, 1991.

48. Chen C.C. and Gardner R.M.:' *Performance of PLL synchronised optical PPM communication systems',* IEEE Trans. On Comm., vol. COM-34, No 10, pp.988-994, Oct. 1986.

49. Davidson F.M. and Sun X.: '*Slot clock recovery in optical PPM communication systems with avalanche photodiode photodetectors',* IEEE Trans. Commun., vol. COM-37, pp.1164-1172, 1989.

50. Sibley, M.J.N.: '*The design and construction of a Digital pulse-position modulation coder and decoder',* Post-Doctoral Fellowship Report, 1987.

51. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *Spectral properties of optical fibre digital PPM',* IEE 4th Bangor Comm. Symp., Bangor, UK, May 1992.

52. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *'Optical fibre n-ary PPM: The question of slot synchronisation',* SPIE OE/FIBRES 92, Boston, USA, pp. 8-12, 1992.

53. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *'Frame synchronisation of optical fibre digital PPM with dispersed pulse shape',* Microwave opt. Techn. Lett., Vol. 5, No. 14, 1992.

54. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *'Spectral characterization and frame synchronisation of optical fibre digital PPM',* Electron. Letters, Vol. 28, No 16, 1992.

55. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *Frame Synchronisation for optical fibre digital PPM',* IEEE International Conf. on Communication Syst. ICCS, pp. 16-20, Singapore, 1992.

56. Sun, X., Davidson, F.M.: '*Word timing recovery in direct detection optical PPM communication systems with avalanche photodiodes using phase lock loop',* Proc. Of the SPIE, vol. 1417, pp. 75-88, 1991.

57. J.J. O'Reilly and W. Yichao,: *' Line code design for digital pulse-position modulation'* IEEE Proc., pt.F, vol. 132, No 6, pp. 441-446, Oct. 1985.

58. J.J. O'Reilly and W. Yichao,: *'Repeatered optical fibre communication based on line-coded PPM transmission',* IEEE Conf. 1985.

59. G.E. Atkin, D.A. Fares: '*Coded multipulse position modulation in a noise channel',* Microwave and Optical Techn. Lett., vol. 2, no. 9, pp.336-340, 1989.

60. M.A. Herro, L. Hu: '*Milti-pulse PPM and a new look at coding for direct detection optical channels using APD receivers',* proceedings of the 23rd conference on Communication, Control and Comp., IL, USA: Univ. Illinois, pp. 401-410, 1985.

61. m. Takahashi et al:'*Capacity and effects of reed-solomon codes on multi-pulse PPM in optical communication',* IEICE Trans., vol. E-72, pp. 1198-1203, 1989.

62. I. Garret; '*Towards the fundamental limits of optical-fibre communications',* Journal of Lightwave technology, Vol LT-1, no. 1, pp. 37-43, 1983.

63. H. Sugiyama and K. Nosu: '*MPPM: A method for improving the band utilization efficiency in optical PPM',* IEEE j. Lightw. Technology, 1989, 7, (3), pp. 465-472.

64. H. Park and J.R. Barry: '*Performance analysis and channel capacity for multiple-pulse position modulation on multipath channels'* Proc. IEEE Int. Symp. On Personal, Indoor and Mobile Radio Communications, 1996, pp. 247-251.

65. G.M. Lee, G.W. Scroeder: '*Optical Pulse Position Modulation with Multiple Positions per Pulsewidth',* IEEE Trans. Comm., vol. COM-25, pp. 360-365, 1977.

66. A. Yu. Gol'dsteyn, B. Ya. Frezinskiy: '*An Investigation of the Transmisión of a Multi-Position PPM Optical Signal Through a Communications Line Containing Repeaters',* Radio. Eng. Electron Phus., vol. 24, pp.65-71, 1978.

67. S.D. Marougi, K.H. Sayhood: '*Noise performance of pulse interval modulation systems',* International Journal of Electronics. Vol. 55, no. 4, pp 603-614, 1983.

68. V.I. Yemin, A. V. Petrich: '*Noise immunity of the reception of optical multiposition pulse-time modulated signals under intersymbol noise conditions',* Radiotekhnika, No. 7, pp.86-87, 1984.

69. I. Bar-David, G. Kaplan: '*Information rates of photon-limited overlapping pulse position modulation channels',* IEEE Trans. Inform. Theory., Vol. IT-30, Iss.3, pp. 455-463, 1984.

70. H. Sugiyama, K. Nosu: '*MPPM: a method for improving the band utilization efficiency in optical PPM',* Journal of Lightwave Technology, vol 7, Iss 3, pp. 465-472, 1989.

71. S.P. et al. Majumder: ' *Effect of imperfect slot synchronisation on the performance of coded optical MPPM systems',* Journal of Optical Communications, vol. 12, Iss 3, pp. 96-100, 1991.

72. M.J.N. Sibley: '*Analysis of multiple pulse position modulation when operating over graded-index plastic optical fibre',* Iee Proc.-Optoelectron., Vol. 151, No. 6, Dec. 2004

73. R.A.Cryan, M.J.N. Sibley: '*Multiple pulse position modulation employing raised cosine filtering',* IEE Proc-Optoelectron., Vol. 153, No. 4, August 2006.

74. K. Nikolaidis M.J.N. Sibley: '*Investigation of an optical multiple PPM link over a highly dispersive optical channel',* IET Optoelectron., 2007, 1, (3), pp. 113-119.

75. K. Nikolaidis M.J.N. Sibley: '*Theoretical investigation into effects of data mapping in optical multiple PPM link',* electronics Letters 13th September 2007, Vol. 43, No. 19.

76. H. Park and J.R. Barry: '*Partial-response precoding scheme for multiple pulse-position modulation'* IEE Proc-Optoelectron., Vol. 150, No. 2, April 2003.

77. G.A. Shirokov, A. Bukhinnik:' *Evaluation of the reliability of signal transmission along digital optical fibre channels with differential pulse position keying',* Telecomm. And Radio Engineering, Pt. 2, vol. 39, no. 7, pp.109-11, 1984.

78. D. Zwillinger:'*Differential PPM has a higher throughput than PPM for the bandlimited and average power limited channel',* IEEE Trans, Inform. Theory, vol. IT-34, Iss. 5, Pt.2, pp.1269-1273, 1988.

79. R.E. Peile: '*Error correction, interleaving and differential pulse position modulation',* International Journal of Satelite Communications, vol. 6, no. 2, pp. 173-187, 1988.

80. Da-shan Shiu, Joseph M. Kahn:' *Differential pulse-position modulation for power-efficient optical communication',* IEEE Trans. On Comm, Vol.47, no. 8. August 1999.

81. U. sethakaest and T.A. Gulliver: '*Performance of differential pulse-position modulation (DPPM) with concatenated coding over optical wireless communication', IET Commun, 2008, 2, (1), pp.45-52.*

82. Z. Ghassemlooy and A.R. Hayes: '*Digital pulse interval modulation for IR communication systems-review',* Int. J. Commun. Syst., 2000, 13, pp. 519-536.

83. E.D. Kaluarachchi: '*Digital pulse interval modulation for optical communication systems',* PhD Thesis, Sheffield Hallam University, UK, 1997.

84. A.R. Hayes, Z. Ghassemlooy and N.L. Seed: '*The performance of digital pulse interval modulation in the presence of multipath propagation'* Proc. 2nd Int. Symp. On Commun. Syst. Networks and DSP, Bournemouth, UK, July 2000, pp. 141-146.

85. ,Z. Ghassemlooy, A.R. Hayes and B. Wilson: '*Reducing effects of intersympol interference in diffuse DPIM optical wireless communication',* IEE Proc-Optoelectron, Vol. 150, No. 5, October 2003.

86. N.M. Aldibbiat, Z. Ghassemlooy and R. Mclaughlin: '*Dual header pulse interval modulation for dispersive indoor optical wireless communication systems',* IEE Proc. Circuits Devices Syst., 2002, 148, (3), pp.187-192.

87. J.M. Kahn, J.R. Barry, M.D. Audeh, J.B. Carruthers, W.J. Krause and G.W. Marsh: *'Non-directed infrared links for high-capacity wireless LAN's',* IEEE Pers. Commun. 1994, 1, pp.12-25.

88. N.M. Aldibbiat, Z. Ghassemlooy and R. Mclaughlin: *'Performance of dual header pulse-interval modulation (DH-PIM) for optical wireless communication systems',* SPIE Proceedings Optical wireless communications III, 2001, 4214, USA.

89. N.M. Aldibbiat, Z. Ghassemlooy: '*Dual header-pulse interval modulation (DH-PIM) for optical communication systems',* Proceedings of CSNDSP-2000, 2000, Bournmouth, UK, pp. 147-152.

90. N.M. Aldibbiat, Z. Ghassemlooy and R. Mclaughlin: *'Error performance of dual header pulse interval modulation (DH-PIM) in optical wireless comminications'*, IEE Pro-optoelectron., vol. 1`48, No. 2, April 2001.

91. J.M.H. Elmirghani, R.A. Cryan, F.M. Clayton: *'Spectral analysis of timing jitter effects on the cyclostationary PPM format'',* Signal Processing, 43, (1995), pp. 269-277.

92. J.M.H. Elmirghani: *'Performance and optimal presynchronization filtering for direct-detection optical fibre PPM',* IEE. Proc-optoelectron., vol 142, No. 6, December 1995.

93. Moe Z. Win: '*On the Power Spectral Density of Digital Pulse Streams Generated by M-ary Cyclostationary Sequences in the Presence of Stationary Timing Jitter',* IEEE Trans. On Comm., Vol. 46, No. 9, Sept. 1998.

94. Hisashi Kobayashi: '*A Survey of Coding Schemes for Transmission or Recording of Digital Data',* IEEE Trans. On Comm. Tech., vol. com-19, no. 6, pp.1087-1100, Dec. 1971.

95. Peter Kabala ns Subbarayan Pasupathy: '*Partial-Response Signaling',* IEEE Trans. On Comm., vol. com-23, No. 9, pp. 921-934, Sept. 1975.

96. Mats Oberg and Paul H. Siegel: '*Performance Analysis of Turbo-Equalized Partial Response Channels',* IEEE Trans. On Comm., Vol.49, No. 3, pp. 436-444, March 2001

97. G. Katsaros, P.M. Lane and M. Murphy: '*A comparison of the impact of FWM on* binary*, duobinary and dicode modulation in DWDM systems',* Presented at LEOS 2000, Puerto Rico, 2000.

98. G. Katsaros and P.M. Lane: '*A comparison between duobinary, dicode and partial response class 4 modulation schemes for optical systems',* Presented at the London Communication Symposium, London, 1999.

99. G. Katsaros, P.M. Lane, J.J O'Reilly and M. Murphy:' *Comparison of the robustness of duobinary, dicode and partial response class 4 modulation schemes to binary transmission in a two channel WDM system',* Presented at the London Communication Symposium, London, 1999.

100. A. Costa, A. Alves and J.J. O'Reilly:'*Investigation of crossphase modulation in WDM systems with AM-PSK modulation formats',* Presented at the London Communication Symposium, London, 2001.

101. Abel Costa, A. Pimenta Alves and J.O'Reilly:*'Evaluation of Optical Fibre Transmisión Systems Base on Optical AM-PSK Signalling'.*

102. R.A. Cryan, R.T. Unwin, I. Garrett, M.J.N. Sibley and N.M. Calvert: '*Optical fibre digital pulse-position modulation assuming a Gaussian received pulse shape',* IEEE Proc. J. Optoelectron., 1990, 137, (4), pp. 89-96.

103. R.A. Cryan: '*Spectral characterisation of dicode PPPM Format'',* electronics letters, 3rd February 2005, Vol. 41, No. 3.

104. Romanos Charitopoulos, M.J.N. Sibley: *'Power Spectral Density of Dicode Pulse Position Modulation',* School of computing and Engineering Researchers' Conference, University of Huddersfield, Dec 2006.

105. R.A. Charitopoulos, M.J.N. Sibley: '*Dicode Pulse Position Modulation Coder Simulation',* School of computing and Engineering Researchers' Conference, University of Huddersfield, Dec 2007.

106. R.A. Charitopoulos, M.J.N. Sibley: '*Experimental Verification of the Power Spectral Density of Dicode PPM'*, submitted for publication at *IEE* Proc.-Optoelectron.

107. R.A. Charitopoulos, M.J.N. Sibley, '*Slot and Frame Synchronisation in Dicode Pulse-Position Modulation*', submitted for publication at *IEE* Proc.-Optoelectron.

108. M.J.N. Sibley: '*Performance analysis of a dicode PPM system, operating over plastic optical fibre, using maximum likelihood sequence detection',* IEE Proc.-Optoelectron., Vol. 152, No. 6, December 2005.

109. I.H. Al-Suleimani, A.J. Phillips and M.S. Woolfson, '*Performance evaluation of optically preamplified dicode pulse position modulation receivers'* Eur. Trans. Telecomms. 2008; 19: 47-52

110. M. Menon and R.A. Cryan, '*Optical wireless communications utilizing a Dicode PPM PIN-BJT Receiver',* Microw. And Opt. Techn. Letters / Vol.45, No. 4, May 20 2005.

Bibliography

- Kinematic and dynamic simulation of multibody systems : the real-time challenge / by García de Jalón, Javier., Bayo, Eduardo
- The scientist and engineer's guide to digital signal processing / by Smith, Steven W.
- First principles of discrete systems and digital signal processing / by Strum, R. D., Kirk, Donald E., 1937
- Digital signal processing : a practical guide for engineers and scientists / by Smith, Steven W.
- RF and microwave transistor oscillator design /Grebennikov, Andrei, 1956
- Electric machines : analysis and design applying Matlab / by Cathey, Jimmie J.
- Simulations of machines using Matlab and Simulink / by Gardner, John F.
- Electromechanical systems, electric machines, and applied mechatronics / by Lyshevski, Sergey Edward
- The VHDL reference : a practical guide to computer-aided integrated circuit design (including VHDL-AMS) / by Heinkel, Ulrich
- Design of phase-locked loop circuits, with experiments. by Berlin, Howard M.
- Phase-locked loops : design, simulation, and applications / by Best, Roland E.
- VHDL : a logic synthesis approach / by Naylor, David., Jones, Simon
- VHDL for logic synthesis / by Rushton, Andrew.
- Introduction to VHDL / by Hunter, R. D. M., Johnson, T. T.
- Digital design and modeling with VHDL and synthesis / by Chang, K. C 1957
- VHDL for designers / by Sjoholm, Stefan., Lindh, Lennart

- Phaselock techniques / by Gardner, Floyd Martin.
- Phase-locked loops : principles and practice. by Brennan, Paul V.
- Handbook of real-time fast Fourier transforms : algorithms to product testing / by Smith, Winthrop W, 1944-, Smith, Joanne M, 1947
- Signals, systems, and transforms / by Jackson, Leland B.
- Modulation, noise, and spectral analysis applied to information transmission / by Panter, Philip Faivel.
- Foundations of digital signal processing : theory, algorithms and hardware design / by Gaydecki, Patrick., Institution of Electrical Engineers.
- Probability, random variables and stochastic processes / by Papoulis, Athanasios, 1921
- Phase-locked loops : application to coherent receiver design / by Blanchard, Alain.
- Digital communications / by Proakis, John G.
- Random signals for engineers using MATLAB and Mathcad / by Jaffe, Richard C.
- Digital signal processing with examples in MATLAB / by Stearns, Samuel D.
- Continuous signals and systems with MATLAB / by ElAli, Taan S., Karim, Mohammad A.
- Phase-locked loops : application to coherent receiver design / by Blanchard, Alain.
- Advanced signal processing and digital noise reduction by Vaseghi, Saeed V.
- Telecommunications engineering / by Dunlop, J., Smith, D. G.
- CDMA : principles of spread spectrum communication / by Viterbi, Andrew J.
- Microwave and RF wireless systems / by Pozar, David M.
- Advanced signal proocessing and digital noise reduction. by Vaseghi, Saeed V.

- Modern communications and spread spectrum / by Cooper, George R. 1921-, McGillem, Clare D.
- Fields and waves in communication electronics / by Ramo, Simon, Whinnery, John R., Van Duzer, T.

Author's Publications

Dicode Pulse Position Modulation Coder Simulation

R.A. Charitopoulos, M.J.N. Sibley

Abstract: Dicode PPM has been proposed as an alternative coding scheme that appears to be more efficient than previous coding techniques. For the first time, Dicode PPM coder software simulation has been constructed and original results have been presented. Also, the window function which is the part of the routine which is used to predict the Power Spectrum Density, have been redefined. Agreement was obtained between experimental and theoretical results.

---

1. Introduction

Digital Pulse Position Modulation (PPM) has been considered in the past as the optimum transmission format for optical communication using both free-space and optical fibre [1-10]. In Digital PPM, the position of a pulse in a frame is controlled by the original PCM word. There are two main variables in Digital PMM signaling, the coding level (the number of PCM bits coded) and the modulation index (a measure of how much of the frame is used for data pulses). The large final line rate of Digital PPM, compared with that of PCM rate, places great demands on system design and this limits its usefulness. To maintain frame synchronisation, a scheme has been developed to generate phase bearing events. It relies on the presence of a pulse in the last slot of one frame followed by a pulse in the first slot of the next frame. Figure 1 shows the conversion of PCM to digital PPM

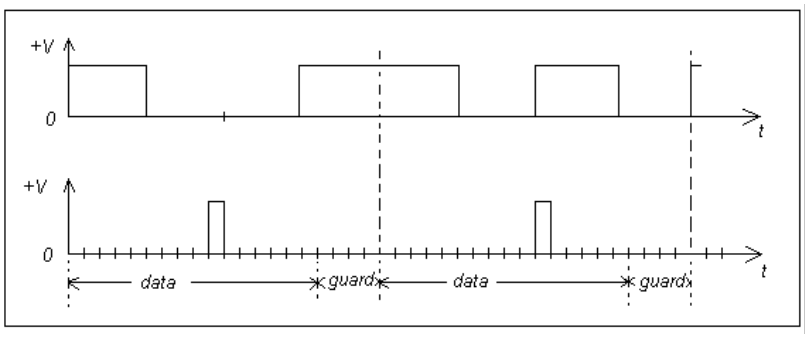


Fig 1: *Conversion of 4 bits of PCM (top trace) to digital PPM (bottom trace).*

Dicode pulse-position modulation (DiPPM) is an alternative coding scheme that is very simple to implement, as it uses only two data slots to code the data transitions in PCM signal [11-14]. This scheme forms the basis of this investigation. It could be said that DiPPM, is the combination of two other formats, Digital PPM and Dicode technique. The Dicode Technique is usually used in magnetic recording channels where bandwidth is limited, although there is some interest in using it in optical fibre links. In this signaling format, only data transitions are sent and no signal is transmitted when the data is constant. In DiPPM format, data transitions from logic zero to logic one are coded as +V and transitions from logic zero are coded as –V.

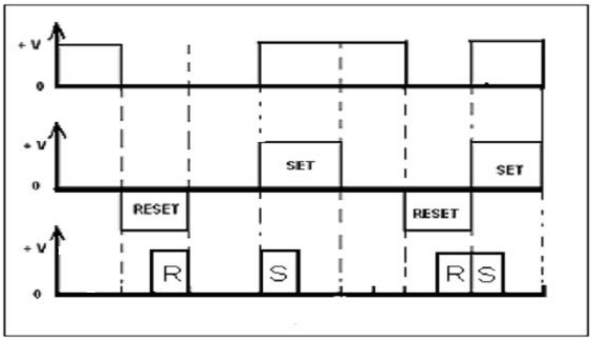


Fig 2: *Conversion of PCM data (top trace) into Dicode (middle trace) and Dicode PPM*

A zero signal is transmitted if there is no change in the PCM signal. The positive pulse can be regarded as setting the data to logic one (pulse SET), whereas the negative pulse resets the data to logic zero (pulse RESET) (figure 2). These Set and RESET signals are converted into two pulse positions in a data frame. Thus a PCM transition from zero to one produces a pulse in slot S and a one to zero transition generates a pulse in slot R. If the PCM data is constant, no signal is transmitted (Table 1). Two slots are used to transmit one bit of PCM, and so the line rate is two times that of the original PCM: considerable reduction in speed compared to digital PPM. Theoretical results show that a simple, leading edge, threshold-detection, DiPPM system gives sensitivities slightly better than that of digital PPM at high fibre bandwidths, whereas for low fibre bandwidths, the sensitivity is significantly greater. Figure 3 shows the conversion of PCM to DiPPM.

Table 1: Dicode PPM symbol alphabet

PCM	*Probability*	*DiPPM*	*Symbol*
00	*1/4*	*no pulse*	*N*
01	*1/4*	*SET*	*S*
10	*1/4*	*RESET*	*R*
11	*1/4*	*no pulse*	*N*

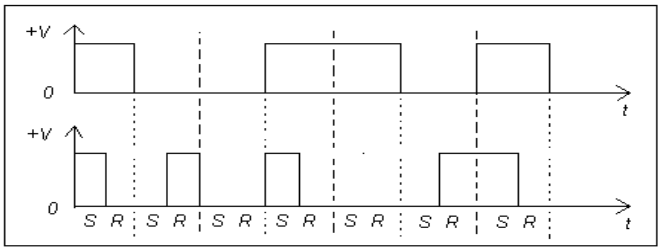


Fig 3: *Conversion of PCM data (top trace) to dicode PPM (bottom trace).*

This paper presents an original software simulation of a DiPPM coder and presents results that have been obtained from it. The use of a windowing equation is also presented.

2. DiPPM Simulation.

Four subroutines have been constructed for the software simulation of the DiPPM coder - the time and the binary (PCM) functions and two others derived from these. The binary function has been used to construct the third subroutine that gives the DiPPM and the fourth subroutine has been built to synchronize all of the other three subroutines (Appendix-A).

The main program has the ability to accept any binary data that the user cares to input. For a pseudo-random input sequence, a built-in Matlab routine has been used. The user can program a binary sequence by using the *''binary_generator= [];''* and the number of the bits of the binary sequence to *''number_of_bits=;''*. The number of bits of the binary sequence will be equal to *''number_of_bits=;''* times the *'' number_of_samples=;''*. The main program also controls the creation of time scale and synchronisation of the functions. The software simulation includes commands that plot the clock, the input binary generator and the DiPPM signal outputs so that the timing can be studied. Also, commands have been included in the main program that plot the Power Spectrum Density of the input binary generator and the DiPPM signal (Appendix-B).

3. Results & Discussion.

3.1 DiPPM Outputs.

Running the software with a frequency of 120 MHz and binary sequence of constant 1.0.1.0, the plot part of the main program produces the graphs of figure 4. Figure 4 shows the output of a DiPPM coder, its PCM binary input and the input clock signal.

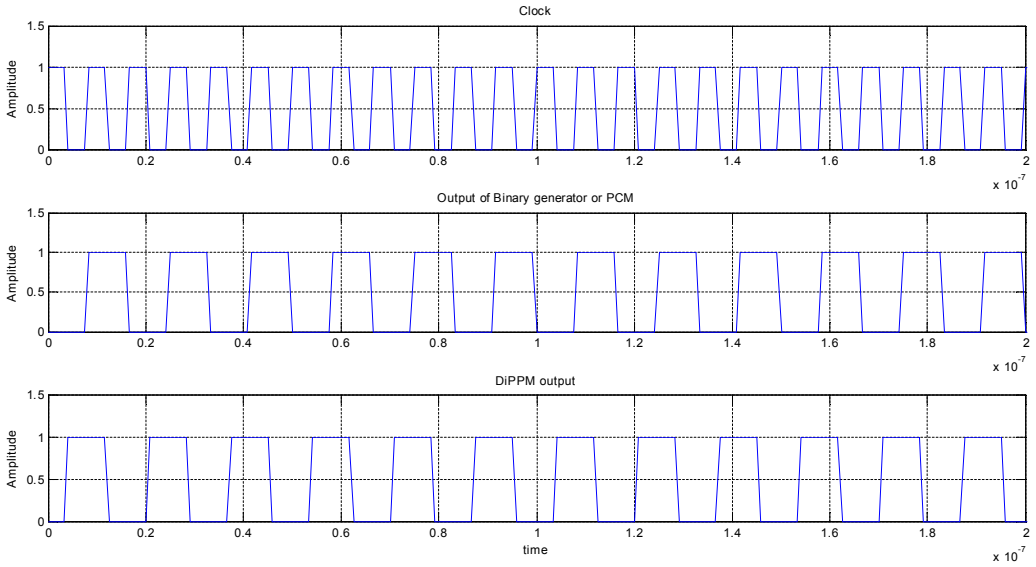


Fig.4: *Plot diagram of clock, PCM and DiPPM* Coder output [1,0,1…]

The output DiPPM pulses are as expected based on the theory of the DiPPM scheme. The signal is SET (10) when the binary data goes from 0 to 1 and RESET (01) when the binary data goes from 1 to 0. These signals are perfectly synchronized and both the binary and DiPPM signal are synchronised with the clock pulses.

Changing the input deterministic signal to PRBS PCM, the plot of figure 5 gives the DiPPM signal.

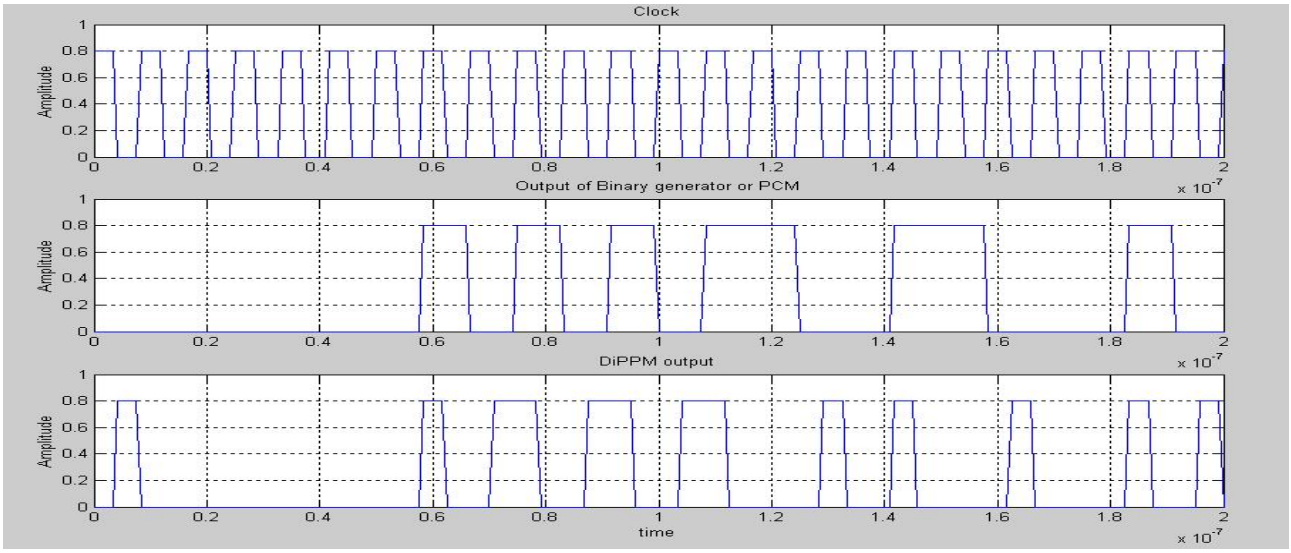


Fig.5 *Plot diagram of clock, Pseudo random generator and DiPPM coder output*

The output of pseudo-random pulses have been synchronised with the clock and the DiPPM coded pulse. The sets (10) and resets (01) of the DiPPM random sequence follow the theory. With a constant one (1) and zero (0) PCM signal the DiPPM signal remains zero (0) as expected.

3.2 Windowing Equation.

To ensure that the spectrum Matlab command simulates the spectrum analyzer equipment, spectral measurements and simulation were taken using a deterministic PCM signal.

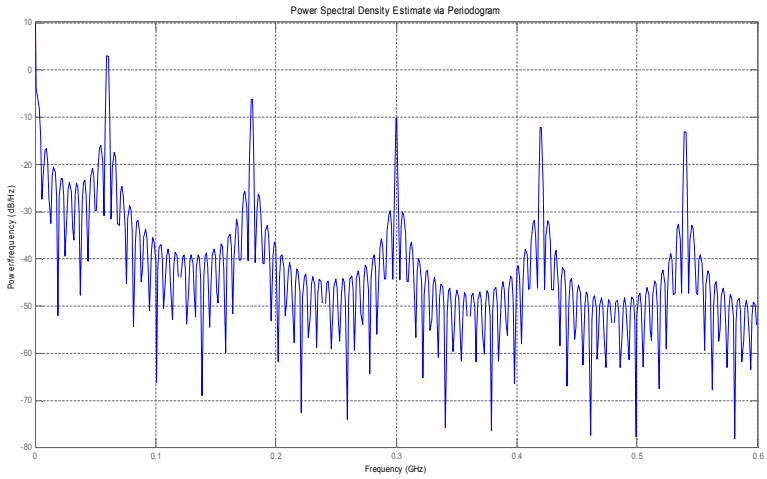


Fig.6: *PCM PSD Software.*

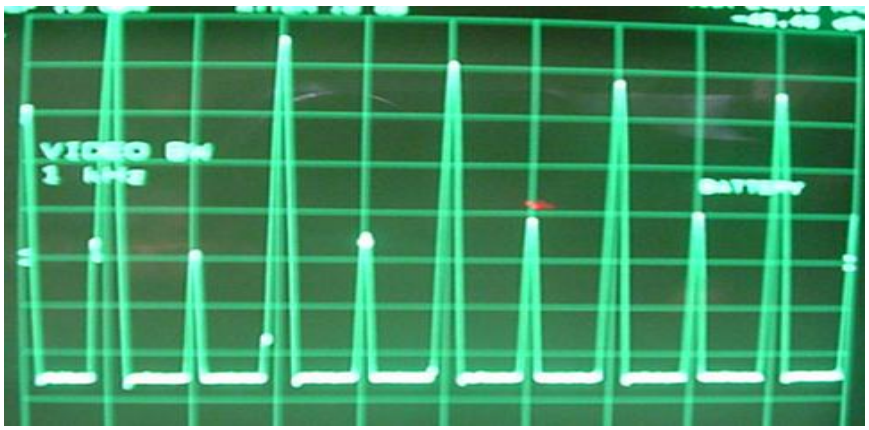


Fig.7: PCM PSD of pulse generator.

As can be seen from figures 6 and 7 the two power spectral densities do not agree. This is due to two effects: the mark:space ratio (m:s) of the input signal may not be 50:50 and the spectrum analyser has a filter at the input which was not modeled by the software.

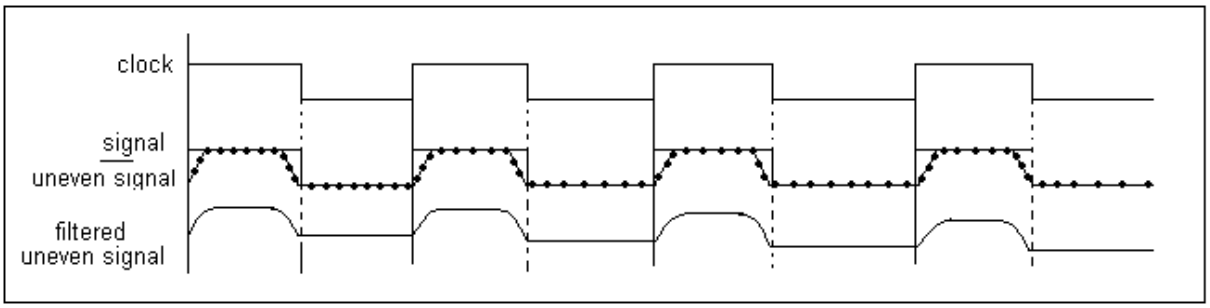


Fig.8: (Up) clock, (middle) signal and uneven signal, (down) filtered uneven signal.

In order to achieve agreement, a window (*filter*) had to be constructed in the computer software to accurately simulate the effects of the m:s ratio and the filter in the spectrum analyzer. This window was found as follows.

$$S(e^{j\omega}) = \frac{1}{n}\left|\sum_{\mu=1}^{n} \mathrm{X}\mu * e^{-j\omega\mu}\right|^2 \quad (1)$$

where [X1……..Xμ] is the signal sequence. In order to match the harmonics of the PSD to those measured by the spectrum analyzer, a window [w1,…., wμ] is introduced that weights the signal sequence. Thus equation 1 becomes

$$S(e^{j\omega}) = \frac{\frac{1}{n}\left|\sum_{\mu=1}^{n} W\mu * X\mu * e^{-j\omega\mu}\right|^2}{\frac{1}{n}\left|\sum_{\mu=1}^{n} W\mu\right|^2} \quad (2)$$

Equation 3 gives the window equation so that the results agree

$$W = \frac{15}{1} * \frac{1}{38}\sin(2\pi ft) + \frac{1}{24}\sin(4\pi ft) - \frac{1}{16}\sin(6\pi ft) + \frac{1}{23}\sin(8\pi ft) \quad (3)$$

where ω is equal to $2\pi f$. The following command, which is used to plot the spectrum, chooses a random window ([ ]):

```
periodogram(binary_generator_time,[],'onesided',1010);
```

The same software command, using the derived window given by 3, is written as:

```
periodogram(binary_generator_time,w,'onesided',1010);
```

Figure 9 shows the spectrum obtained from equation 3 and the mathematically predicted spectrum of the deterministic DiPPM signal. Thus the software now accurately models the hardware and so results obtained from the software program can be considered valid.

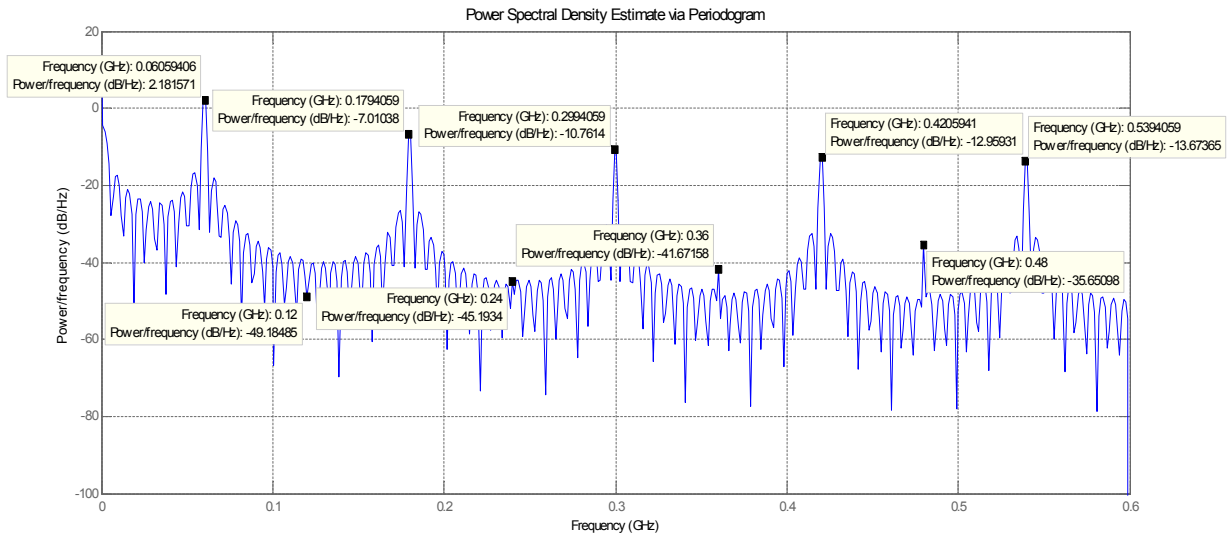


Fig.9: *PSD of PCM signal weighed by window.*

4. Conclusion.

For the first time, software simulation of the Dicode PPM has been given. Outcomes of PCM deterministic and pseudo-random binary sequences inputs have been coded successfully to DiPPM scheme. Also, windowing equation has been given so that agreement was obtained between experimental, theoretical predicted results.

5. References.

1. I. Garrett: *'Digital pulse-position modulation over dispersive optical fibre channels'*. Presented at Int. Workshop on Digital Communications, Tirrenia, Italy, 15-19 August 1983.

2. N.M. Calvert, M.J.N. Sibley and R.T. Unwin: '*Experimental optical fibre digital pulse-position modulation system'* Electroni. Lett. 1988, 24, pp.129-131.

3. R.A. Cryan, R.T. Unwin, I. Garrett, M.J.N. Sibley and N.M. Calvert: '*Optical fibre digital pulse-position modulation assuming a Gaussian received pulse shape'* IEE Proc. J. Optoelectron., 1990, 137, (4), pp 89-96.

4. R.A. Cryan and R.T. Unwin: '*Optimal and suboptimal detection of optical fibre digital PPM'*. IEE Proc. Optoelectronic, 1993, 140, (6), pp.367-375.

5. I. Garrett: *'Digital pulse-position over slightly dispersive optical fibre channels',* Proceedings of Int. Symp. On Information theory, St. Jovite, 1983, pp 78-79.

6. N.M. Calvert, M.J.N. Sibley, and R.T. Unwin.: *'Experimental optical fibre digital pulse-position modulation system',* Electron. Lett., 1988, 24, pp.129-131.

7. I. Garrett, N.M. Calvert, M.J.N. Sibley, R.T. Unwin and R.A. Cryan.: *Optical fibre digital pulse position modulation',* Br. Telecom. Technol. J., 1989, 7, (3), pp. 1953-1954.

8. R.A. Cryan, R.T. Unwin, A.J. Massarella, M.J.N. Sibley, I. Garrett and N.M. Calvert.: *'Optical fibre digital PPM: theoretical and experimental results'*. Presented at UK-USSR Symp. On Communications and applications, 1993.

9. M.J.N. Sibley and A.J. Massarella.: *Detection of digital pilse position modulation over highly/slightly dispersive optical channels',* Presented at SPIE Conf. on video communications and fiber optics networks, Berlin, 1993.

10. Sibley, M.J.N. and R.A. Cryan: *''Minimising intersymbol interference in optical-fibre dicode PPM systems,''* IEE Proc., Optoelectron., 2006,153, (3), pp. 93-99.

11. Sibley, M.J.N.: '*Design implications of high-speed digital PPM'*. Presented at SPIE Conf. on Gigabit Networks, San Jose, CA, 1994.

12. Sibley, M.J.N.: *''Dicode pulse-position modulation: a novel coding scheme for optical-fibre communications,''* IEE Proc., Optoelectron., 2003,150, (2), pp. 125-131.

13. Sibley, M.J.N.: *''Analysis of dicode pulse position modulation using a PINFET receiver and a slighty / highly dispersive optical channel,''* IEE Proc., Optoelectron., 2003,150, (3), pp. 205-209.

14. Sibley, M.J.N.: *''Suboptimal filtering in zero-guard, dicode PPM system operating over dispersive optical channels,''* IEE Proc., Optoelectron., 2004,151, (4), pp. 237-243.

Experimental Verification of the Power Spectral Density of Dicode PPM

R.A. Charitopoulos, M.J.N. Sibley
Department of Engineering and Technology
School of Computing and Engineering
University of Huddersfield
Queensgate
Huddersfield
West Yorkshire
HD1 3DH
ENGLAND

Abstract: This paper describes the construction of a Dicode PPM system and presents, for the first time, the spectrum of Dicode PPM obtained from a practical coder. Original theoretical and computer predicted results agree with experimental observations and this shows the accuracy of the method.

Key terms: Dicode Pulse Position Modulation, Power Spectral Density, Pseudo-Random Binary Sequence, Digital Pulse-Position Modulation, Pulse Coded Modulation, Dicode technique, Pulse Position Modulation.

1. Introduction

Digital Pulse Position Modulation (PPM) has been considered in the past as the optimum transmission format for optical communication using both free-space and optical fibre [1-9]. Unfortunately the format of digital PPM means that although the scheme offers a sensitivity advantage over standard PCM, it does so at the expense of a large bandwidth expansion factor resulting in a final data rate that can be prohibitively high [10].

Dicode pulse-position modulation (DiPPM) [11-13] has been proposed as an alternative coding scheme. It is very simple to implement, as it is only using two data slots to code the PCM signal [14] and can give a sensitivity comparable to digital PPM. DiPPM is the combination of Digital PPM and Dicode technique formats. The Dicode technique is usually used in magnetic recording channels where bandwidth is limited, although there is some interest in using it in optical fibre links. In this signaling format, only data transitions are sent and no signal is transmitted when the data is constant. In the Dicode technique, data transitions from logic zero to logic one are coded as +V and transitions from logic zero are coded as –V. A zero signal is transmitted if there is no change in the PCM signal.

The positive pulse can be regarded as setting the data to logic one (pulse SET), whereas the negative pulse resets the data to logic zero (pulse RESET). These SET and RESET signals are converted into two pulse positions in a data frame. Thus a PCM transition from zero to one produces a pulse in slot S and from one to zero transition generates a pulse in slot R. If the PCM data is constant, no signal is transmitted (Table 1). Two slots are used to transmit one bit of PCM, and so the line rate is two times that of the original PCM: considerable reduction in speed compared to digital PPM. Figure 1 shows the conversion of PCM to DiPPM.

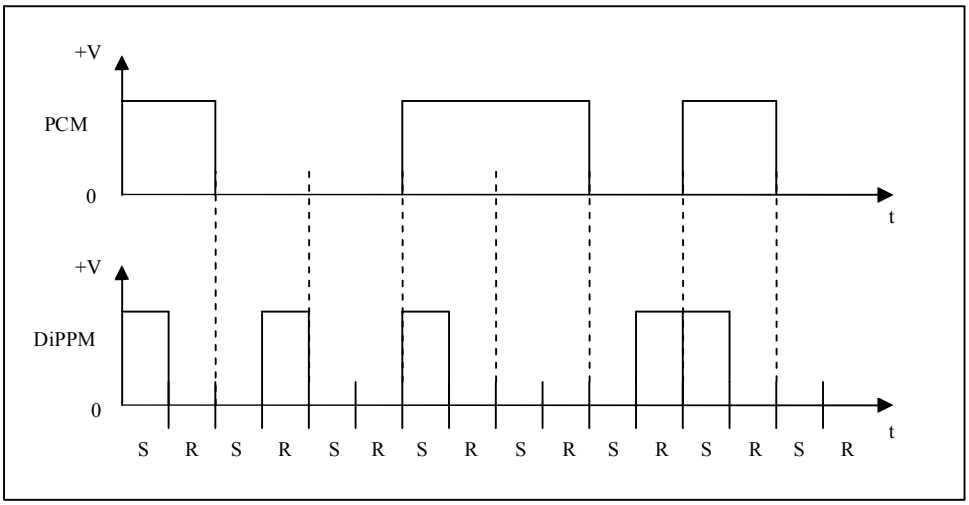


Fig 1: *Conversion of PCM data (top trace) to dicode PPM (bottom trace).*

A recent publication [15] has theoretically shown that power in the DiPPM spectrum is not concentrated near to DC and that it is possible to extract the DiPPM frame-rate component directly from the pulse stream. In order to validate these theoretical results, it is necessary to build a DiPPM system.

This paper presents results obtained from an experimental DiPPM coder system. Original results for the spectrum of a DiPPM signal are presented and show that initial indications were that the measured spectrum does not agree with the previous theoretical work. Also, an original derivation of the DiPPM spectrum has been included to this report and show how modifications can be made so that experimental and theoretical observations can agree.

2. Dicode PPM Spectrum

The spectrum of the DiPPM deterministic sequence, coding PCM at a speed of 120 MBit/s, is shown in figure 2.

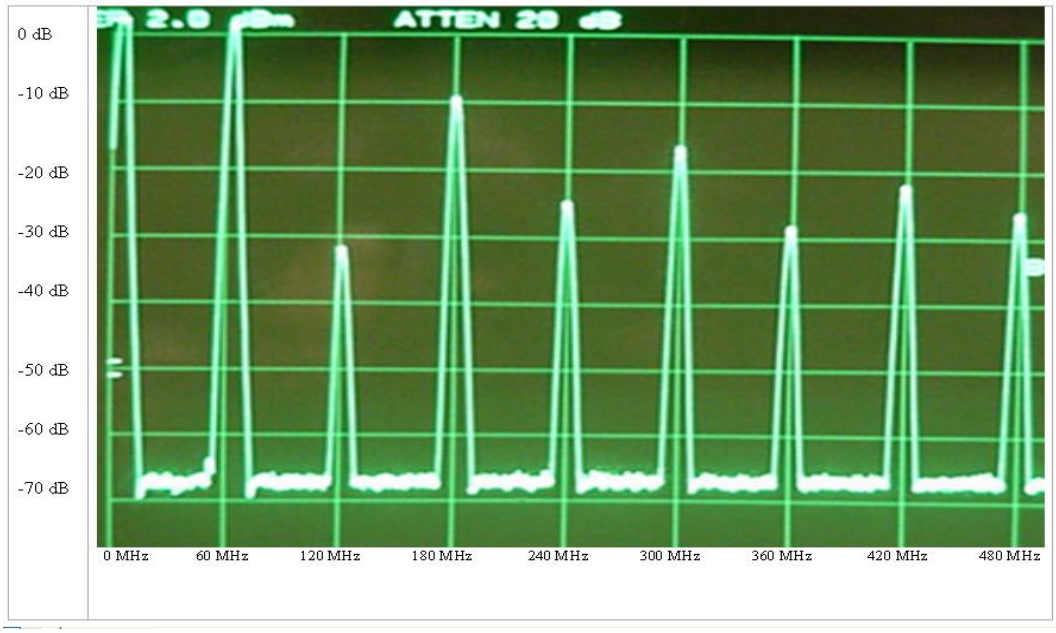


Fig 2: *DiPPM PSD of deterministic sequence (hardware).*

The spectrum of this deterministic DiPPM sequence is similar to the PCM deterministic sequence [11-14] as expected. The spectrum of the DiPPM signal with a Pseudo-Random Binary Sequence (PRBS) PCM signal is shown in figure 3(a) and 3(b).

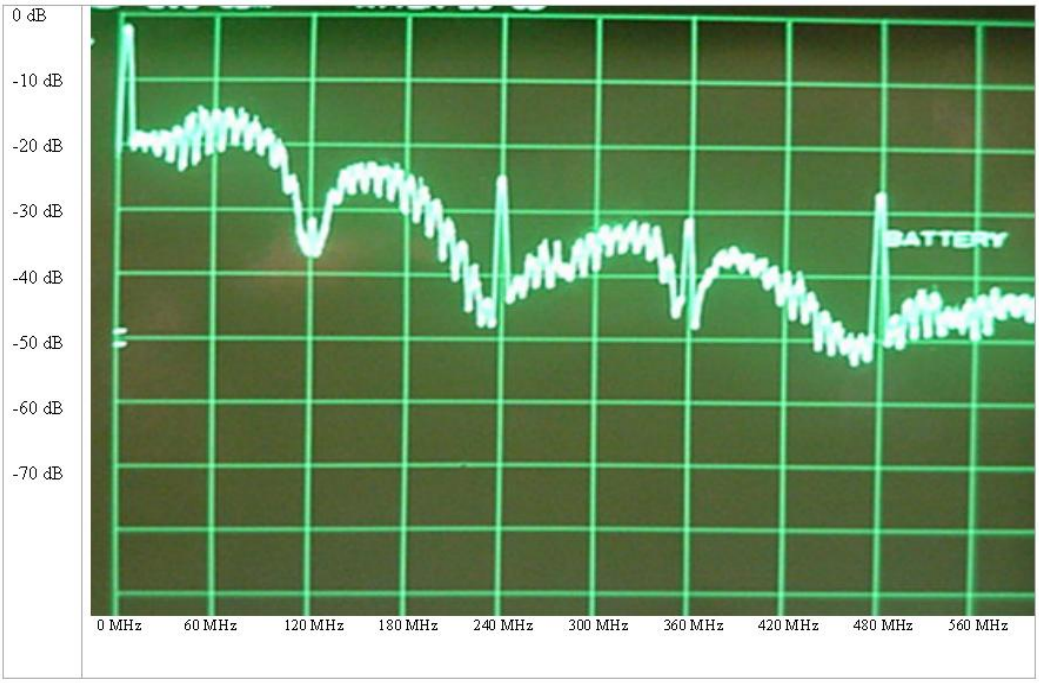


Fig. 3a: *PSD-PRBS DiPPM from coder*

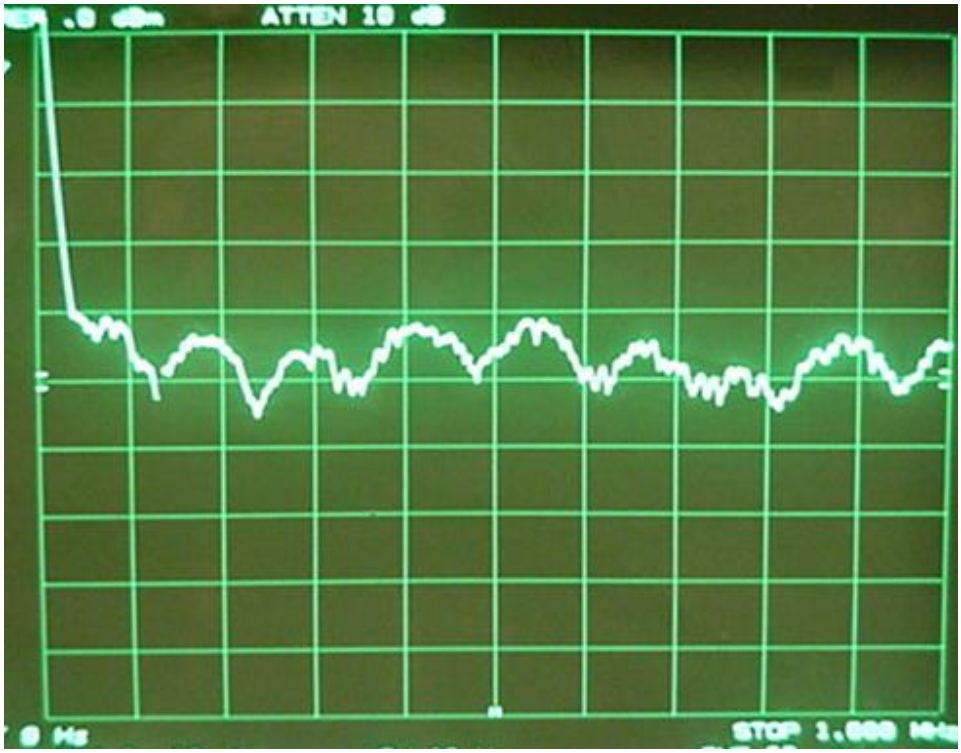


Fig. 3b: *PSD-PRBS DiPPM from coder (0 to 1MHz)*

As can be seen, there is a strong spectral component at 240 MHz and the spectrum is not concentrated near D.C.. Comparison with the results of a previous publication [15] reveals a disagreement, hence further investigation was necessary.

3. DiPPM Spectrum Verification.

In order to validate the theory, and check for correct operation of the coder, initial results were obtained with a deterministic PCM sequence – essentially an alternating sequence of 1s and 0s. Correct operation could then be verified with this simple code before operating with a PRBS. Two approaches were taken: a mathematical derivation and a computer simulation using MATLAB.

3.1 DiPPM Spectrum - Deterministic Sequence.

Mathematical representation

Figure 4 shows the relationship between the original PCM signal and the DiPPM signal. As can be seen, the DiPPM sequence can be regarded as the original PCM signal delayed by 1.5 cycles of the PCM clock.

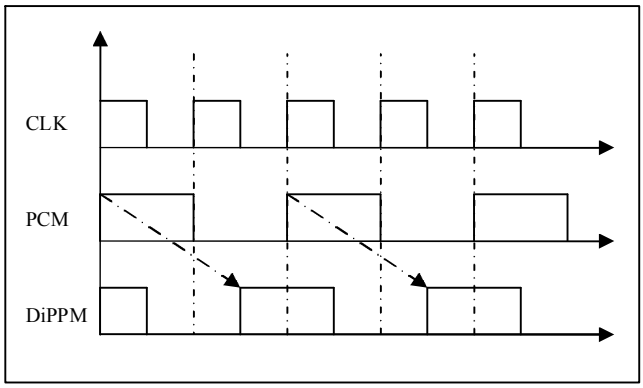


Fig. 4: *Relationship of deterministic PCM & DiPPM signals*

$$h(t) = 1/T \sum_{n=-\infty}^{\infty} Cn \exp(j2\pi ft) \quad PCM\ sequence \tag{1}$$

$$h(t_{DiPPM}) = 1/T \sum_{n=-\infty}^{\infty} Cn \exp(j2\pi ft) \exp(j2\pi f 3t/2) \quad DiPPM\ sequence \tag{2}$$

Equation 1 represents the Power Spectral Density (PSD) of a PCM sequence. Shifting the time axis does not affect the amplitude spectrum (Cn) but it does affect the phase spectrum by adding a phase shift of $\phi s = \pi mft$ to each component in the series. This relationship can be described mathematically with equation 2, where *Cn* is the amplitude spectrum and *f* is the frequency. The equations that produce the deterministic DiPPM signals are X(t) when PCM signal starts with a positive pulse (logic 1) and X(-t) when PCM input signal starts with no pulse (logic 0):

$$X(t) = \frac{A}{2} + \frac{2A}{\pi} * \sin\frac{\omega td}{2}\{\cos(\omega t) - \frac{1}{3}\cos 3\omega t + \frac{1}{5}\cos(5\omega t)....\} \tag{3a}$$

and

$$X(-t) = \frac{A}{2} + \frac{2A}{\pi} * \sin\frac{\omega td}{2}\{-\cos(\omega t) + \frac{1}{3}\cos 3\omega t - \frac{1}{5}\cos(5\omega t).....\} \tag{3b}$$

where *A* is the amplitude, ω is equal to $2\pi f$ and *td* is the width of the pulse. Figure 5 shows the results of these equations.

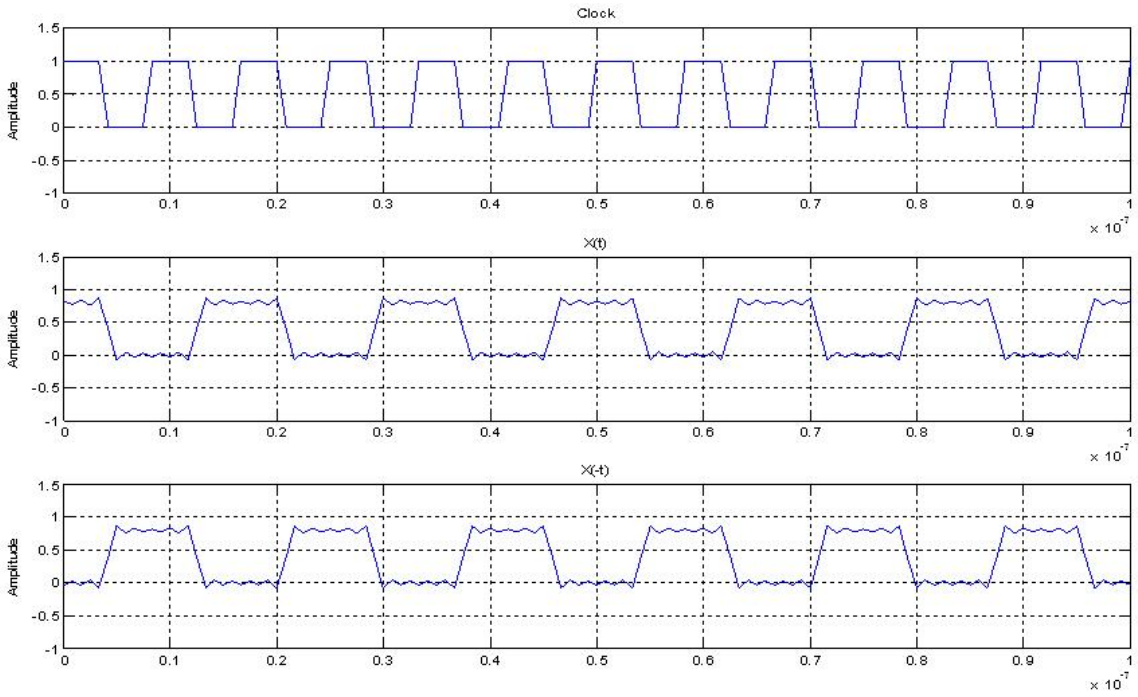


Fig. 5: *DiPPM deterministic sequence*

Software

A DiPPM coder simulation was constructed using Matlab Mathworks. This software simulation includes clock, binary generator and time translation functions which are used to produce a DiPPM signal from an input PCM signal. The simulation includes commands that plot the clock, the input binary generator signal and the DiPPM signal output waveforms. Additional commands were included in the main program to plot

the PSD of the input binary generator and the DiPPM signal. The simulation used exactly the same conditions (frequency, input signal, period, etc) as that of the experimental DiPPM coder.

Plots

The results of the software simulation are shown in figure 6, and figure 7 shows the results of the mathematical analysis, equation 3. Both analyses used exactly the same parameters as the practical test rig.

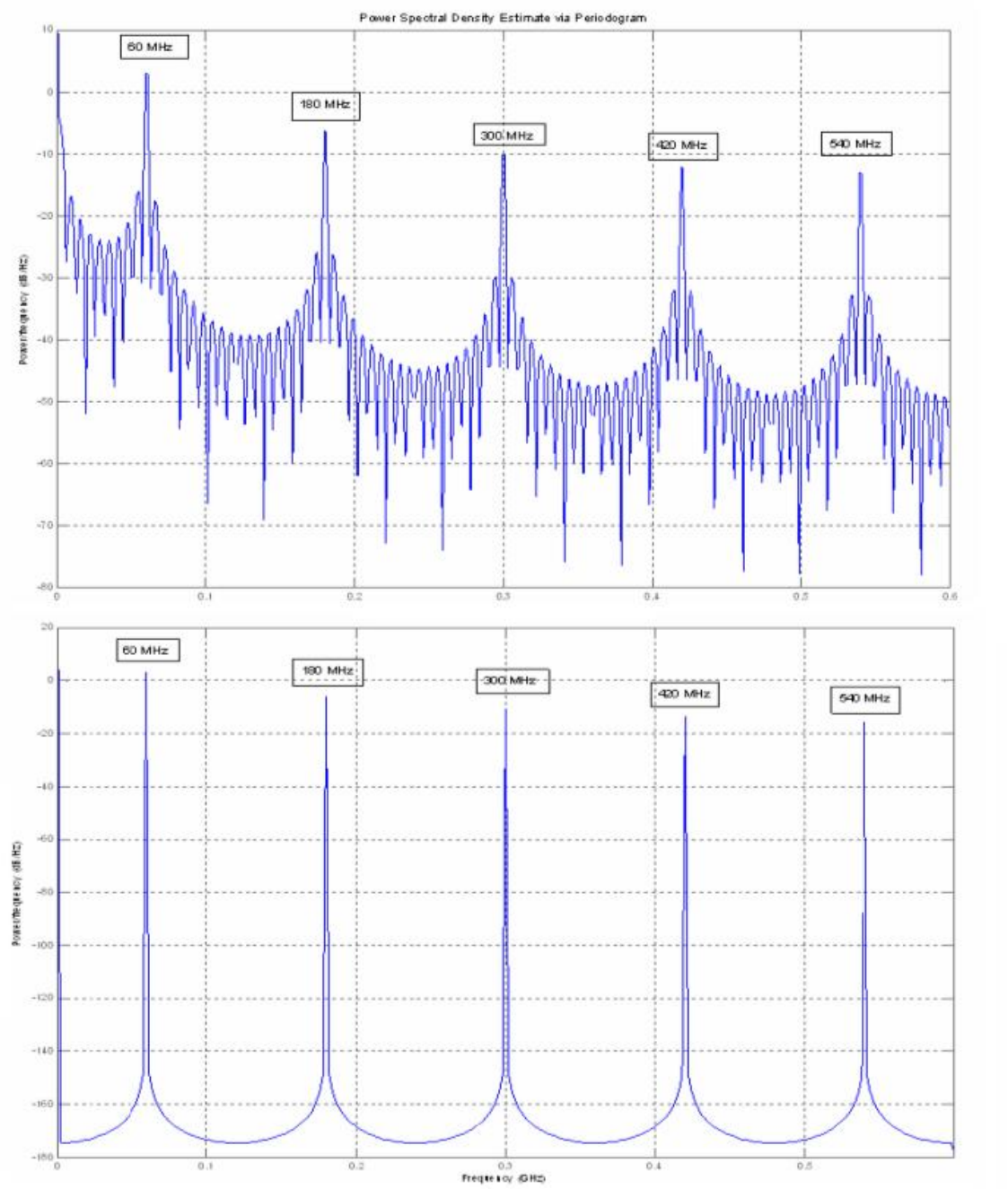


Fig. 6: *PSD of DiPPM software (top trace) coder Spectrum of equation 3 (bottom trace)*

As can be seen from these figures, the results obtained from the simulation and mathematical analyses do not agree with those obtained from the practical system (figure 2). The reason for this is because the software model used a window that did not correctly simulate the shape of the practical data pulses [16]. The practical DiPPM coder was constructed using ECL circuitry operating with a PCM data rate of 120 MBit/s. Due to practical limitations, the mark to space ratio for the SET and RESET signals was uneven. Hence, the DiPPM SET pulse is not equal to half the slot period T. This has the effect of moving the nulls

in the spectrum to a slightly higher frequency and, as a result, this must be accounted for when comparing theoretical and experimental results. All of the computer simulations were modified to take account of this.

In order to achieve the uneven mark:space ratio signals generated in practice, the window had to be modified so that the computer predicted results contained even harmonics. This window was found as follows.

Windowing

Equation 4 gives the PSD of an arbitrary signal sequence [X1……..Xμ] corresponding to the DiPPM signal.

$$S(e^{j\omega}) = \frac{1}{n}\left|\sum_{\mu=1}^{n} X\mu * e^{-j\omega\mu}\right|^2 \tag{4}$$

In order to match the harmonics of the PSD to those of the spectral analyzer, a window [w1,…., wμ] is introduced that weighs the signal sequence. Thus equation 4 becomes

$$S(e^{j\omega}) = \frac{\frac{1}{n}\left|\sum_{\mu=1}^{n} W\mu * X\mu * e^{-j\omega\mu}\right|^2}{\frac{1}{n}\left|\sum_{\mu=1}^{n} W\mu\right|^2} \tag{5}$$

Equation 6 gives the window equation used to simulate the filter of the spectral analyzer so that the results agree

$$W = \frac{15}{1} * \frac{1}{2.4}\sin(2\pi ft) + \frac{1}{0.8}\sin(4\pi ft) - \frac{1}{1.5}\sin(6\pi ft) + \frac{1}{1.2}\sin(8\pi ft) \tag{6}$$

where ω is equal to $2\pi f$. Figure 7 shows the spectrum obtained from equation 6 and the mathematically predicted spectrum of the deterministic DiPPM signal. As can be seen, the results agree with each other.

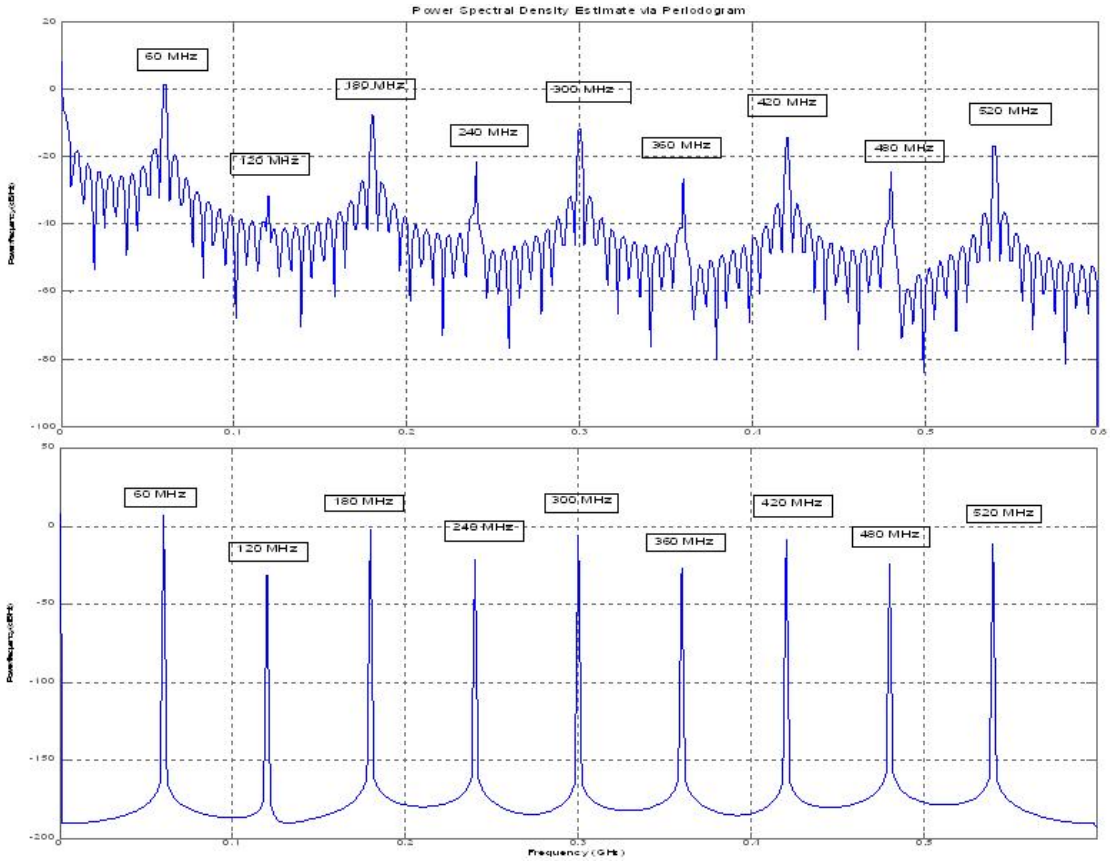


Fig. 7: Deterministic DiPPM spectrums from simulation (upper) and equation 3 (lower) with the use of window.

3.2. DiPPM Spectrum - PRBS Sequence

Mathematical representation

A DiPPM signal, with a PRBS input, can be represented by a pulse stream that includes SET (10) and RESET (01) pulses in random sequence with the proviso that a SET always follows a RESET and vice versa. This can be achieved by adding the equations for four independent pulse trains (Figure 8).

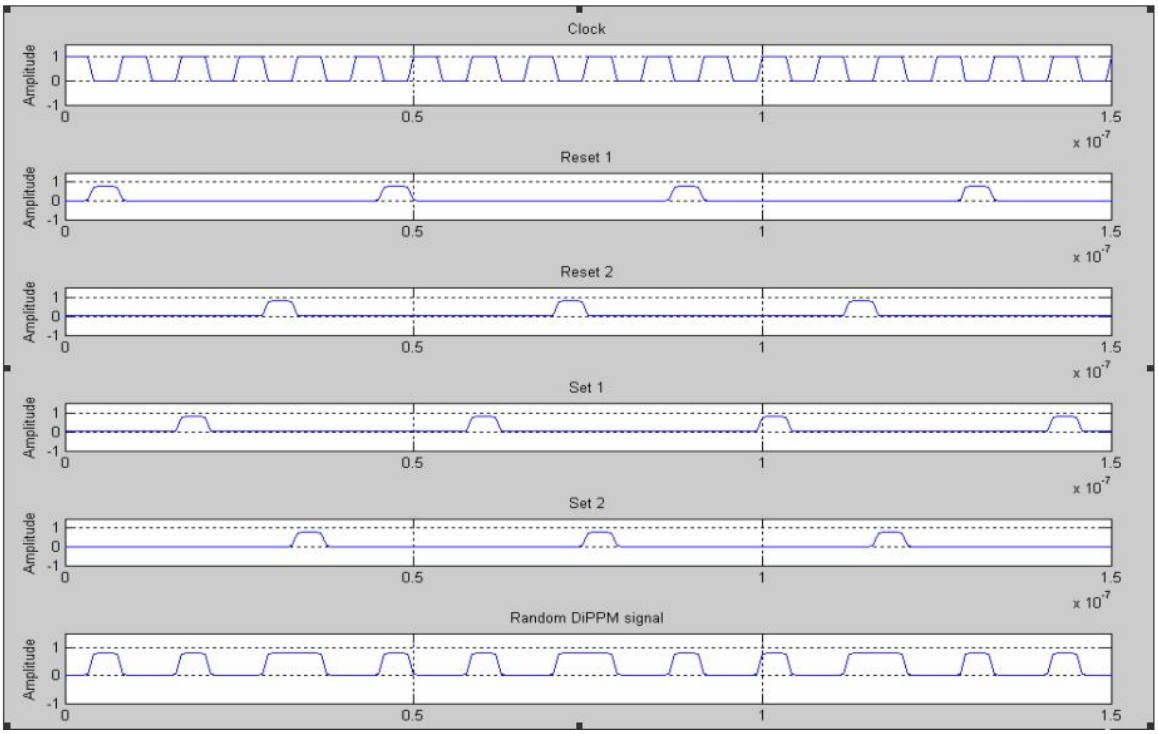


Fig. 8: *Pulse train – DiPPM PRBS*

The equations that produce the pulse trains are:

$$\text{Re } set1 = \frac{Atd}{T} + \frac{2A}{n\pi}\sin(\frac{n\omega td}{2})\cos(n\omega(t-1.4td))dt \tag{7a}$$

$$\text{Re } set2 = \frac{Atd}{T} + \frac{2A}{n\pi}\sin(\frac{n\omega td}{2})\cos(n\omega(t-7.4td))dt \tag{7b}$$

$$Set1 = \frac{Atd}{T} + \frac{2A}{n\pi}\sin(\frac{n\omega td}{2})\cos(n\omega(t-4.4td))dt \tag{8a}$$

$$Set2 = \frac{Atd}{T} + \frac{2A}{n\pi}\sin(\frac{n\omega td}{2})\cos(n\omega(t-8.4td))dt \tag{8b}$$

where A is the amplitude of the pulse, td is the width of the pulse, T is the clock period, and ω is equal to 2πf where f is the clock frequency (1/T). A DiPPM PRBS is produced when equations 7a, 7b, 8a and 8b are added. The phase shifts of the equations have been calculated so that no SET will be followed by another SET and no RESET will be followed by a RESET. Adding these equations gives the PRBS DiPPM signal as:

$$DiPPMprbs = \frac{Atd}{T} + \frac{2A}{n\pi}\sin(n\omega td/2) + (\cos(n\omega(t-1.4td)) + \cos(n\omega(t-7.4td)) + \cos(n\omega(t-4.4td)) + \cos(n\omega(t-8.4)))dt \quad (9)$$

Equation 9 produces a 10 bit PRBS DiPPM waveform. In order to make a longer PRBS sequence, a 12 bit time window is considered for analysis. In this way the sequence appears to repeat every 60 bits (figure 9).

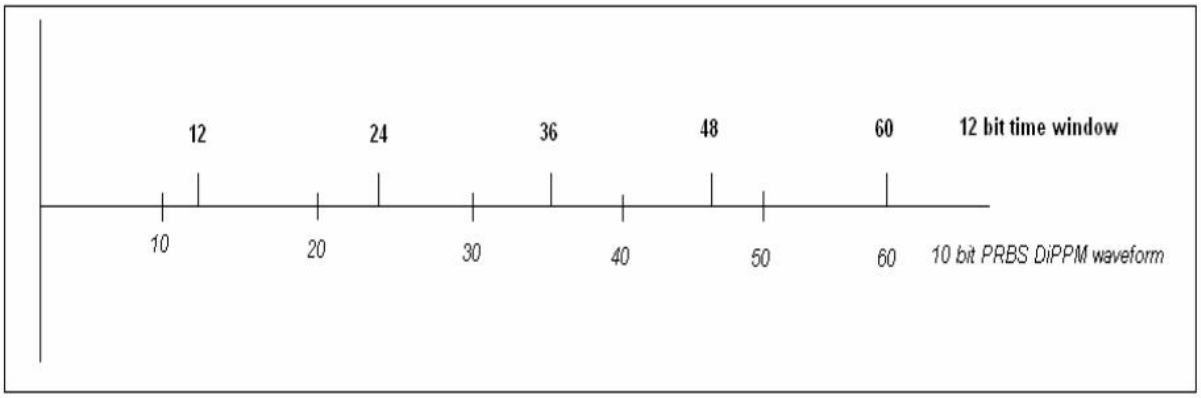


Fig 9: *PRBS DiPPM of 60 bits.*

Software

The software simulation of the DiPPM coder was easily modified to produce a DiPPM PRBS sequence when the input, PCM signal, is a PRBS.

Plot

Figure 10 compares the computer predicted and theoretical PSDs of the DiPPM when the window given by equation 6 is used.

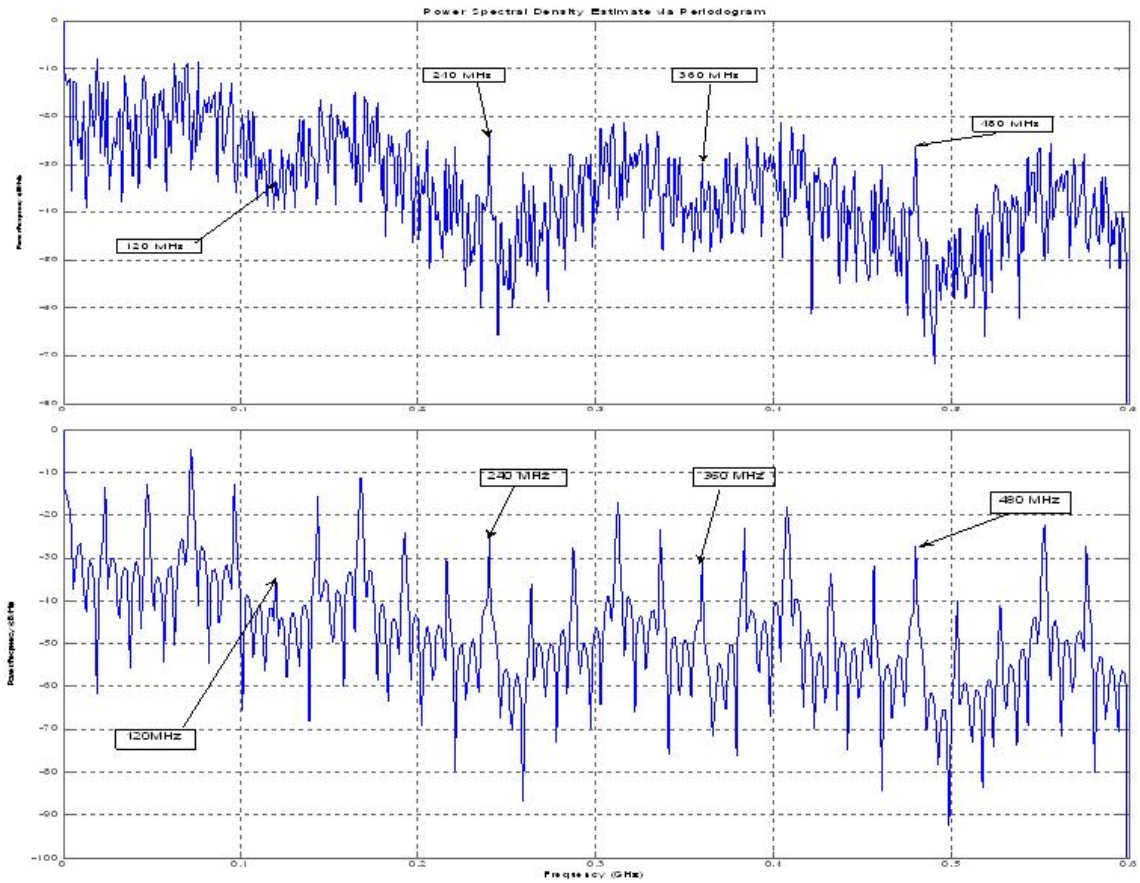


Fig. 10: PRBS DiPPM spectrums from simulation (upper) and equation 6 (lower) with the use of window.

For comparison purposes, the DiPPM simulation software was run with random PCM input sequence and figure 11 shows the result.

Fig. 11: Random DiPPM spectrums from simulation with the use of window.

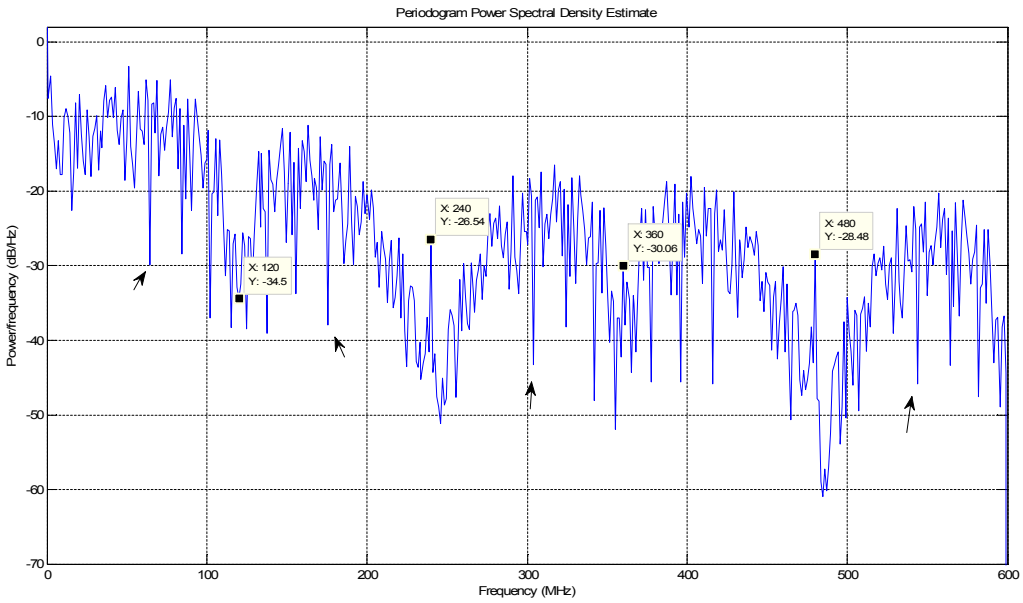


Fig. 11: Random DiPPM spectrums from simulation with the use of window.

4. Discussion.

A Dicode PPM coder has been constructed and the spectrum measured for the first time. In order to check coder operation, a deterministic PCM sequence was coded. A theoretical model was generated as well as one using computer software (MATLAB). It was found that the measured spectrum did not agree with the original predictions (figure 6).

Further investigation showed that the window function used in the software as part of the routine that predicts the PSD of a signal, did not accurately reflect the properties of the practical pulse. Consequently the window was redefined (equation 10) so that agreement was obtained between experimental, theoretical and computer predicted results. Results obtained when a PRBS signal is coded show good agreement only when the window is used. Table 1 compares the powers of several spectral components for the mathematical, MATLAB simulation and measured spectra. As can be seen, there is excellent agreement between all four spectra indicating the validity of the methods.

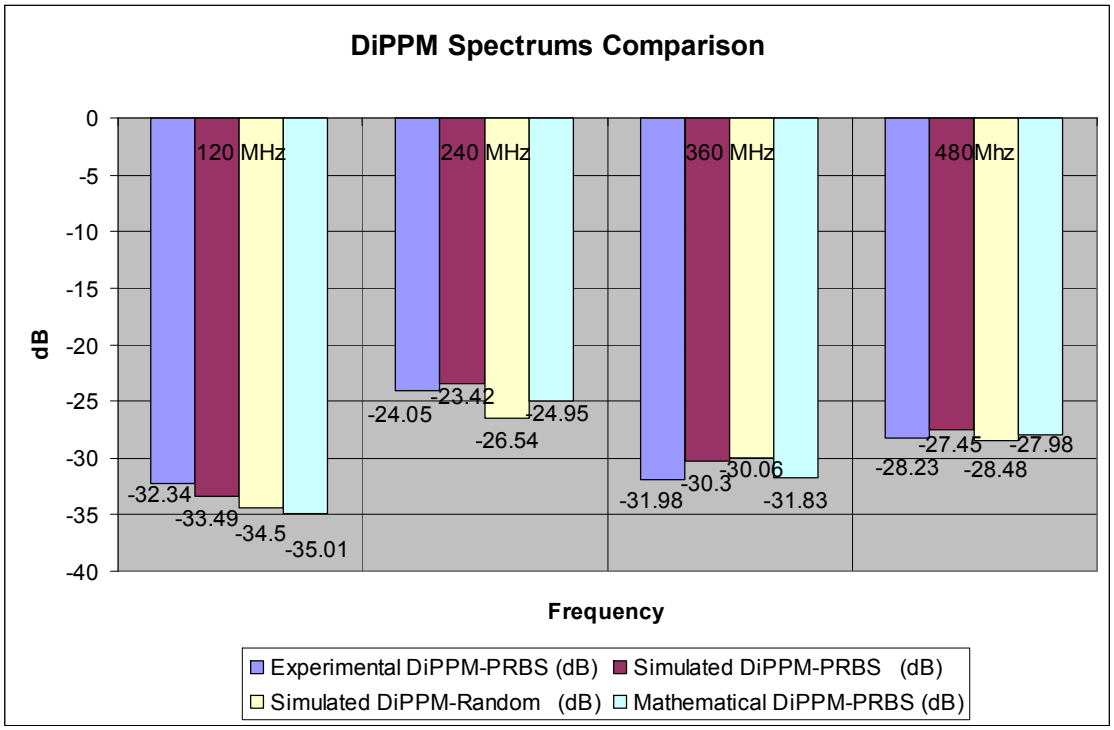


Table 1: *Comparison of DiPPM Spectrums.*

In a previous publication [15], a derivation has been presented for the PSD of a PRBS DiPPM signal. The results of this work agree with the work presented here only if the pulses are ideal (i.e. no window is used).

5. Conclusion.

This paper has presented original results for the spectrum of a DiPPM system when operating with a deterministic and a PRBS PCM input. The data was encoded at a speed of 120 MBit/s. The original work presented here has shown agreement between experimental, theoretical and software simulation results. It has been shown that a window function is required to accurately predict the experimental results.

6. References.

15. I. Garrett: *'Digital pulse-position modulation over dispersive optical fibre channels'*. Presented at Int. Workshop on Digital Communications, Tirrenia, Italy, 15-19 August 1983.

16. N.M. Calvert, M.J.N. Sibley and R.T. Unwin: '*Experimental optical fibre digital pulse-position modulation system'* Electron. Lett. 1988, 24, pp.129-131.

17. R.A. Cryan, R.T. Unwin, I. Garrett, M.J.N. Sibley and N.M. Calvert: '*Optical fibre digital pulse-position modulation assuming a Gaussian received pulse shape'* IEE Proc. J. Optoelectron., 1990, 137, (4), pp 89-96.

18. R.A. Cryan and R.T. Unwin: '*Optimal and suboptimal detection of optical fibre digital PPM'*. IEE Proc. Optoelectronic, 1993, 140, (6), pp.367-375.

19. I. Garrett: *'Digital pulse-position over slightly dispersive optical fibre channels',* Proceedings of Int. Symp. On Information theory, St. Jovite, 1983, pp 78-79.

20. N.M. Calvert, M.J.N. Sibley, and R.T. Unwin.: *'Experimental optical fibre digital pulse-position modulation system',* Electron. Lett., 1988, 24, pp.129-131.

21. I. Garrett, N.M. Calvert, M.J.N. Sibley, R.T. Unwin and R.A. Cryan.: *Optical fibre digital pulse position modulation',* Br. Telecom. Technol. J., 1989, 7, (3), pp. 1953-1954.

22. R.A. Cryan, R.T. Unwin, A.J. Massarella, M.J.N. Sibley, I. Garrett and N.M. Calvert.: *'Optical fibre digital PPM: theoretical and experimental results'.* Presented at UK-USSR Symp. On Communications and applications, 1993.

23. M.J.N. Sibley and A.J. Massarella.: *Detection of digital pilse position modulation over highly/slightly dispersive optical channels',* Presented at SPIE Conf. on video communications and fiber optics networks, Berlin, 1993.

24. Sibley, M.J.N.: '*Design implications of high-speed digital PPM'.* Presented at SPIE Conf. on Gigabit Networks, San Jose, CA, 1994.

25. Sibley, M.J.N.: *''Dicode pulse-position modulation: a novel coding scheme for optical-fibre communications,''* IEE Proc., Optoelectron., 2003,150, (2), pp. 125-131.

26. Sibley, M.J.N.: *''Analysis of dicode pulse position modulation using a PINFET receiver and a slightly / highly dispersive optical channel,''* IEE Proc., Optoelectron., 2003,150, (3), pp. 205-209.

27. Sibley, M.J.N.: *''Suboptimal filtering in zero-guard, dicode PPM system operating over dispersive optical channels,''* IEE Proc., Optoelectron., 2004,151, (4), pp. 237-243.

28. Sibley, M.J.N. and R.A. Cryan: *''Minimising intersymbol interference in optical-fibre dicode PPM systems,''* IEE Proc., Optoelectron., 2006,153, (3), pp. 93-99.

29. R.A Cryan:'*Spectral characterisation of dicode PPM format'.* Electronic Letters, 2005, vol. 41, No. 3, pp. 149-151.

30. R.A. Charitopoulos, M.J.N. Sibley, **'***Dicode Pulse Position Modulation Coder Simulation',* School of Computing and Engineering Researchers' Conference, University of Huddersfield, Dec 2007

''Slot and Frame Synchronisation in Dicode Pulse-Position Modulation ''

R.A. Charitopoulos, M.J.N. Sibley
Department of Engineering and Technology
School of Computing and Engineering
University of Huddersfield
Queensgate
Huddersfield
West Yorkshire

HD1 3DH
ENGLAND
romanos_22@hotmail.com

Abstract

Dicode PPM has been postulated as a coding scheme that appears to be more efficient than some of previous coding techniques. Previous publications reported theoretical simulations of Dicode PPM coder and spectrums of both practical and theoretical Dicode PPM coders have been presented. In this paper, for the first time, the extraction of a clock from a 240 Mbit/s Dicode PPM signal is demonstrated. Original experimental, theoretical and computer predicted results for the spectrum of a Dicode PPM signal are presented. The results obtained agree with each other. It is shown that DiPPM is a very attractive PPM scheme for optical communication

---

Index terms

Dicode Pulse Position Modulation, Power Spectral Density, Pseudo-Random Binary Sequence, Digital Pulse-Position Modulation, Pulse Coded Modulation, Dicode technique, Pulse Position Modulation.

Introduction

The advantages of many pulse position modulation (PPM) schemes in direct detection pulsed laser communications have been well documented in the past [1-5]. PPM schemes are mostly used over optical communications, using both free-space and optical fibre. As only the data is transferred over the optical system, no clock signal is transmitted to the PPM decoder. But symbol synchronisation is required in every digital communication system which transmits information synchronously. Thus, a synchronised clock signal is necessary for the decoder to decide which of the M slots occurring during a frame time contains the optical pulse. Hence, timing extraction must be implemented for slot synchronisation and clock recovery.

Bennett [6] was one of the first authors who addressed statistical and timing description of the timing extraction synchronizations. Many other publications have been mentioned on timing extraction in digital transmission systems [7-10] and on the structure of the

synchronizer [11-12]. Gagliardi [13], another one of the first authors on synchronisation of PPM systems, presented a method that pulse edge tracking had been used to feed the error signal to an oscillator running at the slot frequency. Six years after the publication of Gagliardi in 1980, Gol'dsteyn and Frezinskiy [14] presented an algorithm that defines the position of digital PPM pulse under non-ideal clock synchronisation. Gagliardi again, presented the time synchronisation problem that exists in PPM systems [15] and he considered a method using a correlator to establish the error signal that corrects the oscillator [16-17]. The use of full slot width pulses cancels the spectral component and consists of squaring the photo detector current before feeding the signal to a phase lock loop (PLL) [18-19].

A successful timing recovery has been reported by Davidson [20]. In 1986, Sibley [21] considered a practical measurement of the timing component within digital time frame. Spectral properties of PRBS digital PPM data have been publicised [22-25] in 1992. An analytical solution has been presented for a given coding level and modulation index and requirements for slot and frame synchronisation have been considered. Elmirghani also developed an original technique for frame synchronisation that does not need the use of low modulation indexes in order to increase the timing component for high coding level systems [26].

1.1 Timing Extraction Methods

A popular method to extract the clock from a PPM format, especially with satellite optical space PPM, is that with maximum likelihood detector. The synchronisation sequences usually are inserted periodically into the data stream, where at the receiver, the frame phase is estimated. Hence they are synchronised with the generated frame clock. Because of these inserted sequences do not contribute towards the data but they represent a redundant power, hence, the sensitivity of the system has been affected and that is a disadvantage of this method because the sensitivity is the main attraction of PPM. The system operating bit rate will be reduced as sequences have to be added at the transmitter and to be deleted from the receiver later.

Tracking pairs of back to back pulses is another timing extraction method, in which sequences do not suffer from any limitations, is one that has been used in free-space PPM. But as the optical fibre PPM is different of that of satellite PPM structure, higher coding levels has to be used so the optimum sensitivity could be achieved. The method that does not affect the sensitivity of the PPM systems is the one that uses phase lock loop (PLL) in a timing extraction and synchronisation systems. It could be used in most PPM formats. The clock which is synchronised with the PPM signal, generated by the PLL, hence the PPM format does not lose its sensitivity.

A last method uses redundancy introduced by a line code which limits the maximum pulse deviation from the centre of the frame and estimates the frame phase. However, this method adds complexity at both transmitter and receiver as it requires coder and decoders, which makes its implementation difficult.

1.2 Alternative PPM Schemes

In the alternative PPM schemes, such as Multiple PPM (MPPM) and Differential PPM (DPPM), the process of clock recovery and synchronisation is more difficult. With multiple PPM a PCM data word is formatted into a data frame of several pulses. Three PCM bits can be coded into a two pulse, five slot frame. The location of the pulses in the frame determines the PCM data word (e.g.. PCM=>Multiple PPM, '000'=>'00011', '011'=>'10001', '110'=> '10010'). Although the performance of multiple PPM is superior to both digital and Dicode PPM (DiPPM), the constant change of the pulse position makes it not attractive to the most methods of frame phase synchronisation. On the other hand, Differential PPM is a simpler scheme to multiple PPM. Every frame of Differential PPM starts with a SET ('1') bit and guard is following ('00'). Thus in the first stage of binary scale, the PCM word '000' is represented as '100' in differential PPM format. As we count up in the PCM binary scale, more '0' bits are added to the differential PPM frame (e.g.. PCM=> DPPM, '001'=> '1000', '011'=> '100000'). As Differential PPM format contains one SET ('1') bit at each frame and it is always at the beginning of the frame, a PLL could be used for the timing extraction. But its format does

not make it suitable for optical communication because its sequence increases the period time of the frame depended on the input PCM word and that makes the throughput lower.

2.3 Digital PPM and Dicode PPM Review

Digital pulse position modulation is the scheme, that has been considered in the past as the optimum transmission format for optical communication using both free-space and optical fibre [27-32]. Although the digital PPM scheme offers a sensitivity advantage over standard pulse coded modulation (PCM), it does so at the expense of a large bandwidth expansion factor resulting in a final data rate that can be prohibitively high [33]. Lately an alternative PPM format, dicode pulse position modulation (DiPPM), has been theoretically presented [34-36]. Unlike digital PPM, the DiPPM system achieves a better sensitivity than PCM at a slot rate of twice the original PCM data rate. Theoretical results showed that a simple, leading edge, threshold-detection DiPPM system gives sensitivities slightly better than that of digital PPM at high fibre bandwidths, whereas for low fibre bandwidths, the sensitivity is significantly greater [37].

As few theoretical analyses have been presented about DiPPM format, less experimental results have been published. A complete coding simulation of DiPPM has been shown, while theoretical and practical power spectrum density (PSD) of deterministic DiPPM have been presented [38]. From the comparison of these two DiPPM PSDs with those of deterministic PCM spectrums, a windowing equation has been given to obtain the agreement between experimental and theoretical results. The use of the windowing method, which produces a pulse sequence with an uneven mark-space ratio, has been analysed and original results of deterministic and pseudo-random binary sequences (PRBS) DiPPM have been presented. Also mathematical equations of DiPPM PSDs have been given and their reliability has been proved through the comparison of original experimental, theoretical and computer predicted results for the spectrum of a Dicode PPM signal [39]. Analysis of the DiPPM PSD showed that DiPPM format is highly suited for deployment over optical channels, as its spectrum is not concentrated at near DC frequencies. Although it has been mentioned that a DiPPM coder/decoder has been constructed [38-39], no reference of any optical measurement have been reported.

A timing extraction system is an integral part of a complete optical DiPPM hardware system that maintains the necessary decoder timing to accomplish the DiPPM data detection. Such system, after the photodetector [36], produces a slot clock that is in phase with the received slots that the receiver decoder requires.

In this paper, for the first time, timing extraction slot and frame synchronisation for DiPPM scheme and optical receiving system have been presented. Measurements of its Power Spectrum Density (PSD) through a simple optical link have been given. Comparison and theoretical analysis with that of previous DiPPM coder PSD have been also included. Before examining the performance of the optical DiPPM timing extraction system, it will be useful to present briefly the DiPPM scheme.

2. Dicode PPM

Dicode PPM is a coding scheme that is very simple to implement. Two slots are used to transmit one bit of PCM, and so the line rate is two times that of the original PCM: considerable reduction in speed compared to digital PPM. For example, in Fig. 1, it can been seen that a PCM transition from zero to one gives a pulse in the first time slot, slot S for set (symbol S), while a transition from one to zero at PCM data gives a pulse in the second time-slot, slot R for reset (symbol R). Constant PCM data as such 00 or 11, results in no pulses, symbol N (no pulse).At the decoder a decoding flip-flop is either set or reset according the position of the received pulse in a frame.

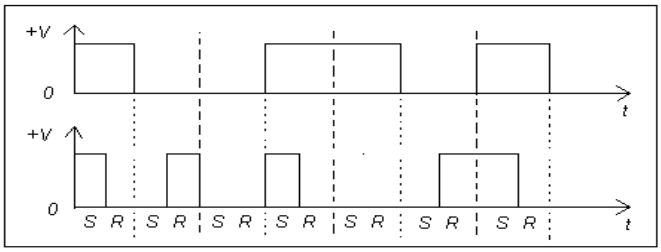


Fig.1: *Conversion of PCM data (top trace) to dicode PPM (bottom trace).*

Table 1 shows the dicode PPM symbol alphabet. As the symbols are four, each symbol has a probability of ¼. However, the probability of a no pulse sequence (N) is ½ as it occurs at both 00 and 11 PCM sequence. The S symbol has the same probability as the no

pulse (N), because there are only two possible PCM sequences (00 or 01) after an R pulse has been transmitted. A typical dicode sequence would be S, xN, R with probability of ½, $(1/2)^x$ and ¼ respectively.

Table 1: Dicode PPM symbol alphabet

PCM	Probability	DiPPM	Symbol
00	1/4	no pulse	N
01	1/4	SET	S
10	1/4	RESET	R
11	1/4	no pulse	N

Table 1: *Dicode PPM alphabet.*

3. DiPPM Optimum Receiver

The error sources and probabilities of DiPPM have been presented and analysed in [36] where the DiPPM format included ISI guards and later on at [37] with minimised intersymbol interference as it has been presented in section 2. The symbol error probability of DiPPM, with the hypothesis that the receiver output noise voltage is a Gaussian random variable, is

$$P_{es} = P_r + P_s + P_f \tag{1}$$

where P_r is the erasure error probability

$$P_r = 0.5 erfc\left(\frac{Q_r}{\sqrt{2}}\right) \tag{2}$$

where

$$Q_r^2 = \frac{\left(v_p - v_d\right)^2}{\left\langle n_o(t)^2 \right\rangle} \tag{3}$$

where $\left\langle n_0(t)^2 \right\rangle$ is the mean square receiver output noise, v_d is the receiver output at the threshold crossing time t_d and v_p is the peak receiver output which occurs at time t_p. P_s is the wrong slot error probability

$$P_s = erfc\left(\frac{Q_s}{\sqrt{2}}\right) \tag{4}$$

where

$$Q_s^2 = \left(\frac{t_s}{2}\right)^2 \frac{1}{\left\langle n_o(t)^2 \right\rangle} \left(\left. \frac{dv_0(t)}{dt} \right|_{td} \right)^2 \tag{5}$$

while t_s is the DiPPM slot time. At last P_f is the probability per time slot of a false alarm error

$$P_f = \frac{t_s}{T_R} 0.5 erfc\left(\frac{Q_f}{\sqrt{2}}\right) \tag{6}$$

where

$$Q_f^2 = \frac{(v_d - v_{0ISI})^2}{\left\langle n_0(t)^2 \right\rangle} \tag{7}$$

where t_s / T_R is the number of uncorrelated samples per time slot. T_R is the time at which the autocorrelation function of the receiver filter has become small.

The DiPPM optimum receiver constituted by four parts: a pre-amplifier, an amplifier, a simple low-pass filter and threshold-crossing detection (figure 2).

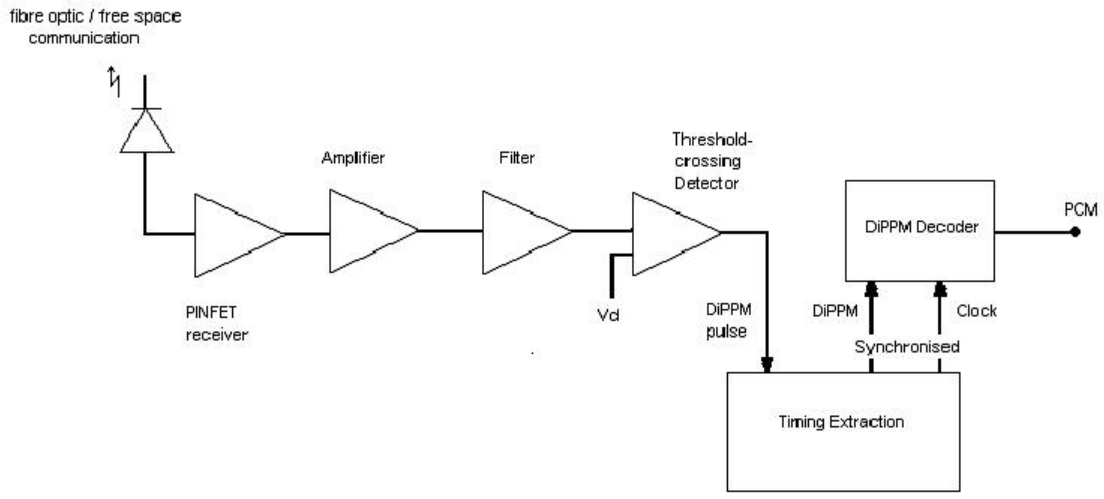


Fig.2: *DiPPM Optical receiver connected to DiPPM Timing Extraction.*

The DiPPM coded signal was used to modulate an 850nm VCSEL. As the VCSEL was packaged, a fibre pig-tail was used to couple into a silicon PIN photodetector. The pre-

amplifier was based on a common-collector front-end [36] having a -3dB bandwidth of 240 MHz. A post amplifier fed a low-pass, 240 MHz filter and a threshold crossing detector sliced the pulse stream.

Si PIN Pre-amplifier Receiver

The Si PIN pre-amplifier receiver (figure 3) is composed by two buffers (Q1 and Q3) and an amplifying circuit (Q2).

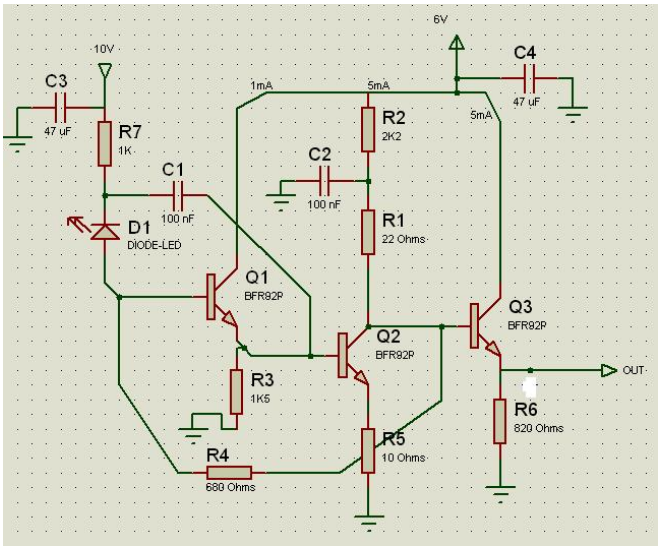


Fig. 3: *Si PIN pre-amplifier receiver.*

The bandwidth of the pre-amplifier is equal to

$$BdW = \frac{1}{2\pi \frac{R_4}{(1+A_v)}\left(C_d(1-A_1)+C_c+(1+A_v)C_f\right)} \tag{8}$$

where

$$A_v = -g_{m_2} R_1 \tag{9}$$

$$g_m = I_E / 25mv \tag{10}$$

while A_1 is the input voltage at Q_1, C_d is the emitter-base capacitance, C_c is the collector emitter capacitance and the C_f is the capacitance at the feedback resistor R_4.

4. DiPPM Timing Extraction

It has been shown [38-39] that DiPPM is an attractive scheme for optical communication and it can be implemented very easily. In addition, timing extraction, slot and frame synchronisation is simple because of the position of the pulses in the frames.

To achieve the clock recovery from a random DiPPM sequence and synchronisation of the extracted clock with the DiPPM signal, a *second-order* phase-locked loop (PLL) system has been used in the DiPPM Timing Extraction circuit (Fig.4).

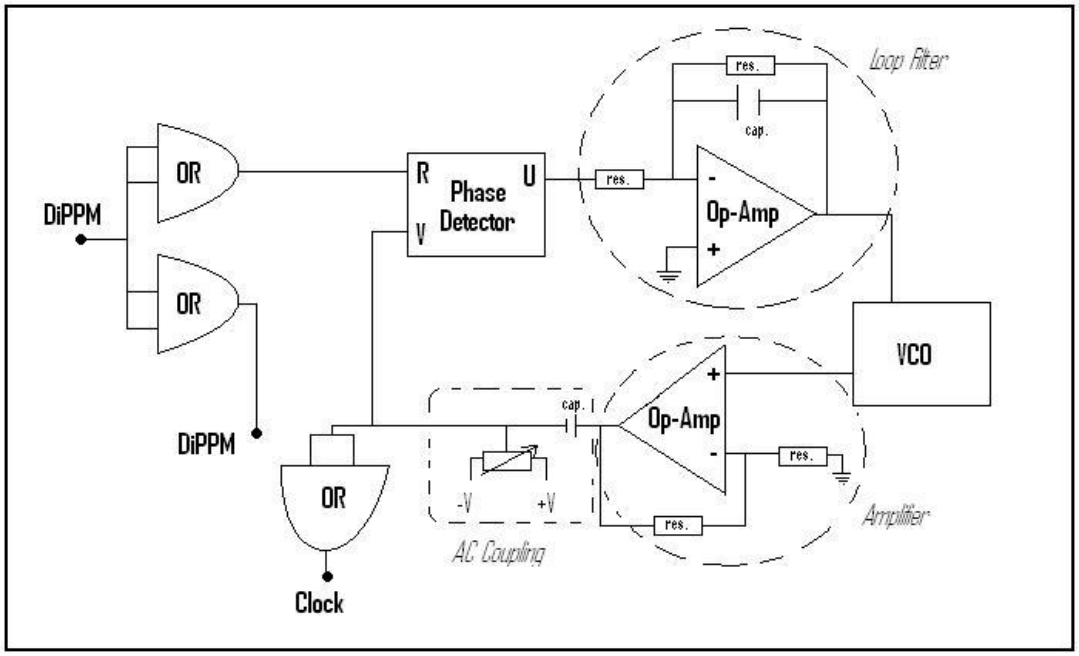


Fig 4: *DiPPM timing extraction diagram.*

Two OR gates have been used to buffer the input DiPPM signal (Fig 4). The output of one of the OR gates output feeds the DiPPM decoder (see Fig. 2). The output of the other OR gate goes to the PLL system, to produce the synchronized clock signal. The PLL circuit that has been constructed consists of a phase detector, an active low-pass filter, a

Voltage-Controlled Oscillator (VCO) and an amplifier for any necessary amplification of the VCO output signal (fig. 4). As the practical DiPPM coder [38] has been constructed using ECL circuitry, and the phase detector is an ECL component, an AC coupling circuit has been added to shift the VCO output signal to ECL standards.

The *Loop Filter* (Fig.4) which determines the dynamic performance of the loop, including the capture and lock ranges, the bandwidth and the transient response, has a cutoff frequency (ω_{LPF}) equal to 100 kHz.

$$\omega_{LPF} = \frac{1}{R_2 C} \tag{12}$$

where R_2 is the resistor in parallel with the capacitor C. Both resistors that have been used at the filter are equal.

As the practical DiPPM coder [18] was operating with a PCM data rate of 120 MBit/s, the output data rate of the DiPPM will be expected to be 240MBit/s.Thus, the VCO has to operate at 240 MHz (Fig.5 – VCO Clock).

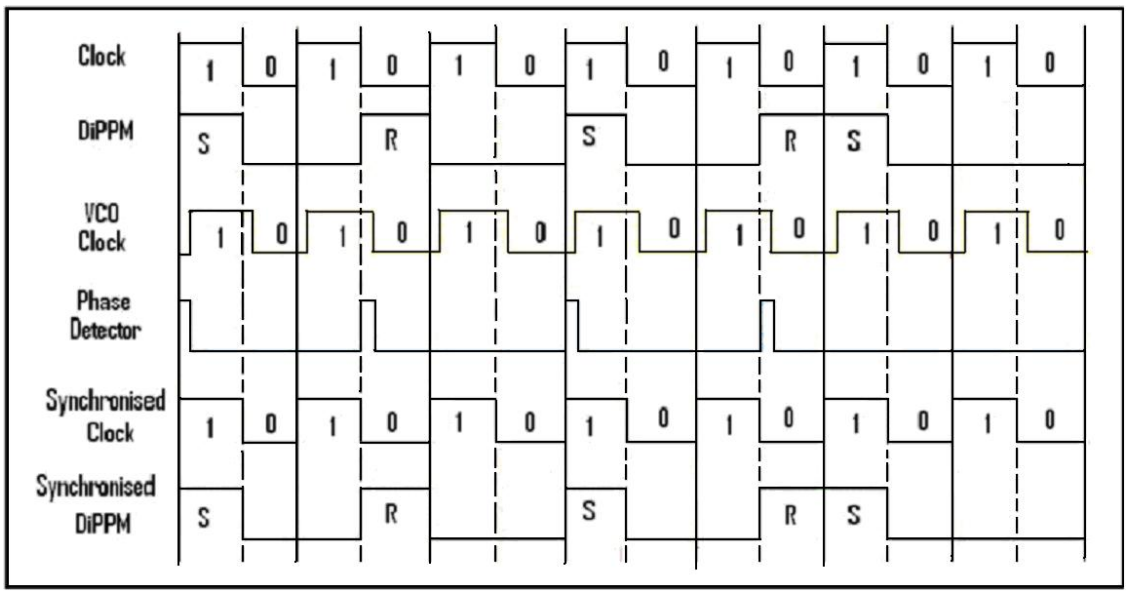


Fig 5: *DiPPM timing extraction waveforms.*

5. Results and Discussions

Measurements of the outcomes of the DiPPM Timing Extraction hardware prove the truth of the theory that already has been described in this paper. Figure 6 shows the synchronised PRBS DiPPM and clock outputs of the hardware.

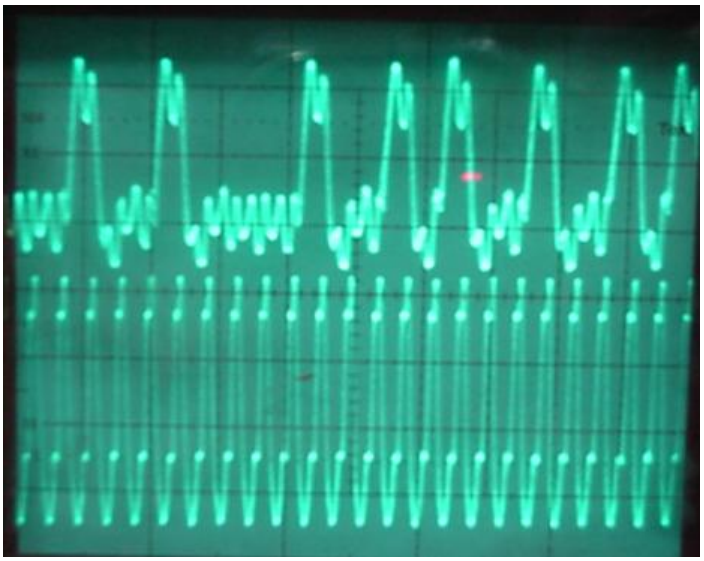


Fig. 6: *PRBS DiPPM (top trace) Recovered Clock (bottom trace).*

As the synchronised outputs of the Timing Extractor have been passed to the DiPPM decoder, the DiPPM signal decoded to its original PCM format (figure 7).

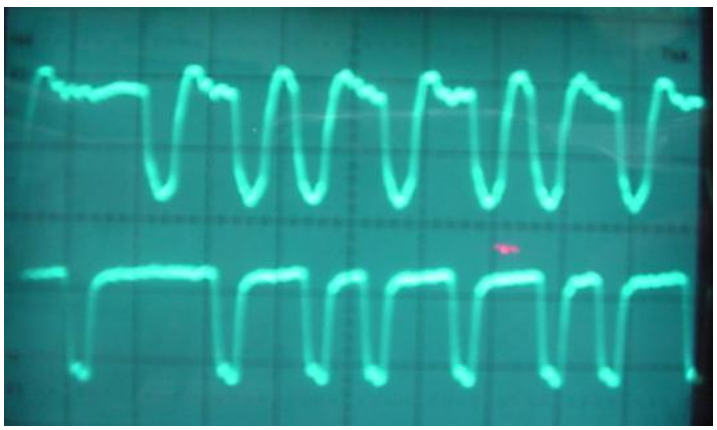


Fig. 7: *PCM Input of Complete Optical DiPPM system (top trace) PCM Output of Complete Optical DiPPM system (bottom trace).*

Any PCM signal of the DiPPM decoder that comes out can be produced only by synchronised input data. Figure 8 shows the PSD of the DiPPM optical signal.

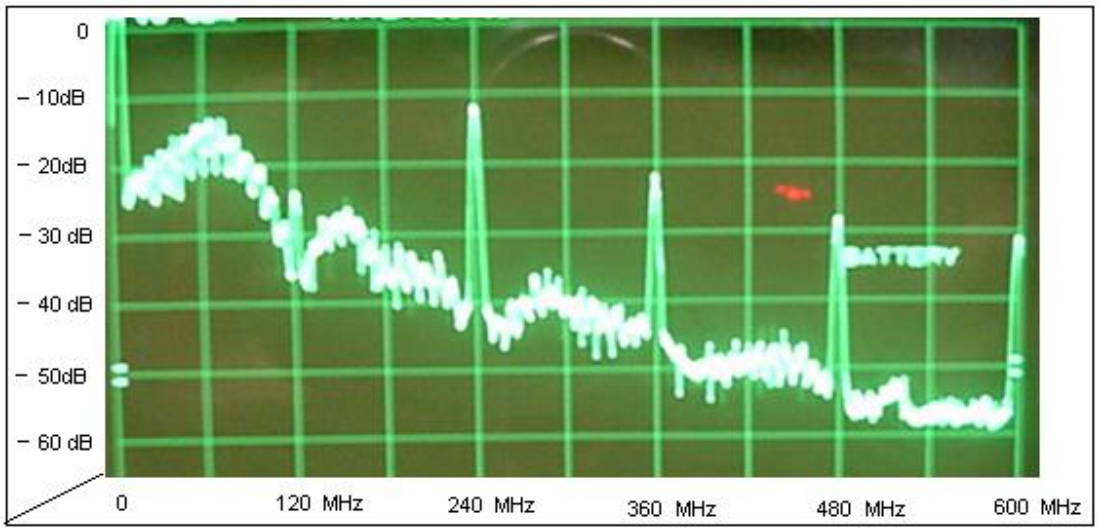


Fig. 8: *Optical DiPPM spectrum (synchronised DiPPM-Recovered clock).*

The DiPPM PSD of figure 8 disagrees with the DiPPM PSD result of previous publication [39] with the power of the fourth harmonic. This is because the width and the shape of the DiPPM pulses are different to those used in the previous publication [39]. Following the steps that have been described at [39], a windowing equation (14) has been constructed that simulates the shape and the width of the experimental pulses, both PRBS PCM - DiPPM (see [39])

$$W_{opt.} = 15 - \frac{1}{0.9}\cos(2\pi ft) + \frac{1}{0.2}\cos(2\pi ft) - \frac{1}{0.7}\cos(6\pi ft) + \frac{1}{1.42}\cos(8\pi ft)dt \qquad (14)$$

Running the simulation code of [38] and the equation 6 of publication [39] with windowing equation 14 of this report, the spectrum results are coming out are:

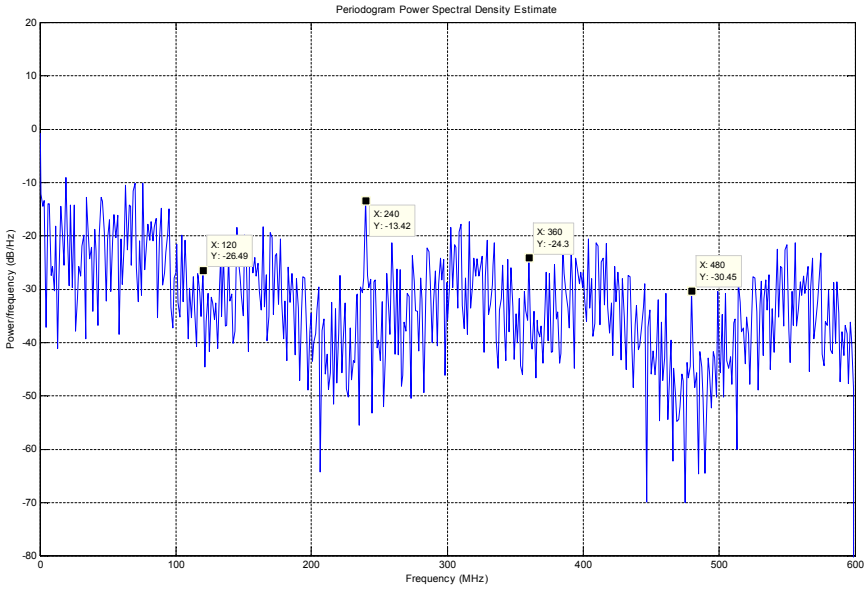


Fig.8: *Optical DiPPM spectrum (simulated).*

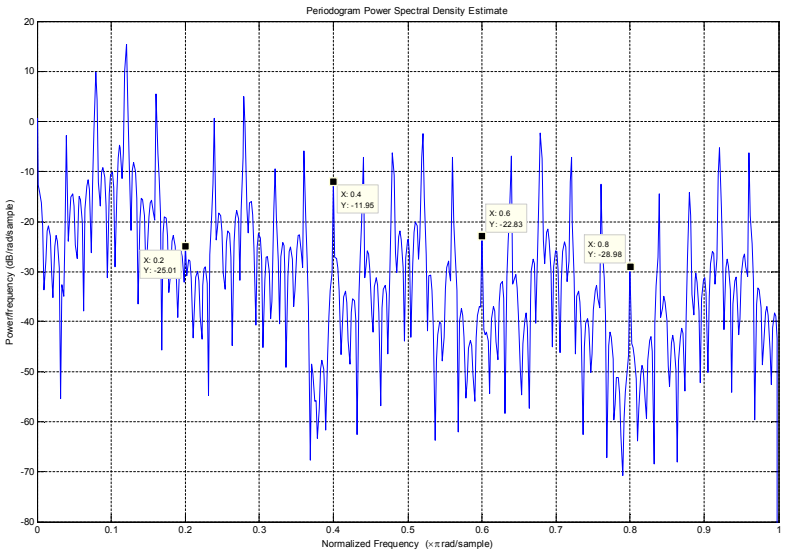


Fig.9: *Optical DiPPM spectrum (from equation 3 of [39]).*

From the comparison of figures 7, 8 and 9 it can easily be seen the resemblance of the spectrums. Their differences at the values of their corresponding powers are negligible that makes them more probable.

6. Conclusion

For the first time, a system for extracting a clock from a DiPPM system has been presented. Theoretical descriptions of the process of the timing extraction system have been given and measurements of practical signals have shown that slot and frame synchronisation has been achieved. This synchronisation has been achieved without the need for adding redundant sequences. For this reason and because DiPPM is very easy to implement (coder, timing extraction, decoder), DiPPM appears to be more efficient and than previous coding techniques.

The power spectral density of an optical DiPPM signal has been presented and has been compared to that of previous publication [39]. The reasons for any differences between the PSDs presented in this paper and [39] have been analysed and explained. A window equation has been given that simulates the experimental optical DiPPM PRBS sequence. Power spectral densities of DiPPM have been presented from mathematical and software simulation with the use of this window equation. Comparison of the spectra shows the accuracy of the mathematical and software predictions and the experimental results. An optical system has been described and the PIN bipolar optical receiver, used in the experiment, has been presented.

References

1. H.Sugiyama and K. Noso: 'MPPM: A method for improving the band-utilization efficiency in optical PPM', IEEE J. Light. Technolo, 1989, 7, (3), pp.465-472.

2. D. Zwillinger: 'Differential PPM has a higher throughput than PPM for the band-limited and power-limited optical channel'. IEEE Trans. Inform. Theory, 1988, 34, (5). Pp. 1269-1273.

3. H.M.H. Shalaby: 'A performance analysis of optical overlapping PPM-CDMA communication systems' IEEE J. Light. Tecnol, 1999, 17, (3), pp 426-434.

4. Da-shan Shiu, Joseph M. Kahn : 'Differential Pulse-Position Modulation for Power-Efficient Optical Communication', IEEE Trans. On comm., vol 47, No 8, August 1999.

5. Ger Ling, Robert M. Gagliardi: 'Slot Synchronization in Optical PPM Communication', IEEE Trans. On Comm, vol. com-34, No.12 December 1986.

6. Bennett W.R.: '*Statistic of regenerative digital transmission',* The Bell system technical J., pp. 1501-1542, Nov. 1958.

7. Takasaki Y.: '*Timing extraction in baseband pulse transmission',* IEEE Trans. On Comm., vol. COM-20, No 5, pp.877-883, Oct. 1972.

8. Franks L.E.: *'Carrier and bit synchronization in data communication – A tutorial review',* IEEE Trans. On Comm., vol. COM-28, No 8, pp. 1107-1121, Aug. 1980.

9. Gitlin R.D. Salz J.: *'Timing recovery in PAM systems',* The Bell system technical Jrl., pp. 1645-1649, May-June 1971.

10. Datta D. and Gengopadhyay R.: '*Simulation studies on nonlinear bit synchronizers in APD-based optical receivers',* IEEE Trans. On Comm. Vol. COM-35, No 9, pp. 909-917, Sept. 1987.

11. Rosenberg R.L., Chamzas C. and Fishman D.A.:*' Timing recovery with SAW transversal filters in the regenerators of undersea long-haul fibre transmission systems',* IEEE J. on selected areas in Comm., vol. SAC-2, No 6, pp. 957-65.

12. Gardner F.M.: '*Phaselock Techniques',* 2nd ed., New York: Wiley, 1979.

13. Gagliardi R.M.: '*Synchronisation using pulse edge tracking in optical pulse-position modulated communication systems',* IEEE Trans. On Comm., pp. 1693-1702, Oct. 1974.

14. Gol'dsteyn A. Yu. and Frezinskiy B. Ya.: '*Statistical estimation of the time of arrival of a sequence of optical PPM pulses',* Telecomm. And Radio Eng, Part 2, vol. 35, No. 10, pp. 120-121, 1980.

15. Gagliardi R.M..: '*Time synchronisation in optical PPM Sequences',* IEEE Globcom 83, vol. 2. pp. 779-783, 1983.

16. Ling R. and Gagliardi R.M., '*Slot synchronisation in optical PPM communications',* IEEE Trans. On Comm., vol. COM-34, No. 12, pp 1202-1208, Dec.1986.

17. Gagliardi, R.M. and Ger Ling: *Slot synchronization in Optical PPM Communications',* IEEE Trans. On comm.., vol. COM-34, No. 12, Dec., 1986.

18. Chen C.C., Win M.Z., Marshall W.K. and Lesh J.R.: '*Low data rate coherent optical link demonstration using frequency stabilized solid state laser',* Free-Space Laser Comm. Techn. III, SPIE, vol. 1417, pp. 170-181, 1991.

19. Chen C.C. and Gardner R.M.:*' Performance of PLL synchronised optical PPM communication systems',* IEEE Trans. On Comm., vol. COM-34, No 10, pp.988-994, Oct. 1986.

20. Davidson F.M. and Sun X.: '*Slot clock recovery in optical PPM communication systems with avalanche photodiode photodetectors',* IEEE Trans. Commun., vol. COM-37, pp.1164-1172, 1989.

21. Sibley, M.J.N.: '*The design and construction of a Digital pulse-position modulation coder and decoder',* Post-Doctoral Fellowship Report, 1987.

22. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *Spectral properties of optical fibre digital PPM',* IEE 4th Bangor Comm. Symp., Bangor, UK, May 1992.

23. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *'Optical fibre n-ary PPM: The question of slot synchronisation',* SPIE OE/FIBRES 92, Boston, USA, pp. 8-12, 1992.

24. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *'Frame synchronisation of optical fibre digital PPM with dispersed pulse shape',* Microwave opt. Techn. Lett., Vol. 5, No. 14, 1992.

25. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *'Spectral characterization and frame synchronisation of optical fibre digital PPM',* Electron. Letters, Vol. 28, No 16, 1992.

26. Elmirghani, J.M.H., Cryan, R.A., Clayton, F.M.: *Frame Synchronisation for optical fibre digital PPM',* IEEE International Conf. on Communication Syst. ICCS, pp. 16-20, Singapore, 1992.

27. N.M. Calvert, M.J.N. Sibley and R.T. Unwin: 'Experimental optical fibre digital pulse-position modulation system' Electron. Lett. 1988, 24, pp.129-131.

28. R.A. Cryan, R.T. Unwin, I. Garrett, M.J.N. Sibley and N.M. Calvert: 'Optical fibre digital pulse-position modulation assuming a Gaussian received pulse shape' IEE Proc. J. Optoelectron., 1990, 137, (4), pp 89-96.

29. N.M. Calvert, M.J.N. Sibley, and R.T. Unwin.: 'Experimental optical fibre digital pulse-position modulation system', Electron. Lett., 1988, 24, pp.129-131.

30. I. Garrett, N.M. Calvert, M.J.N. Sibley, R.T. Unwin and R.A. Cryan.: Optical fibre digital pulse position modulation', Br. Telecom. Technol. J., 1989, 7, (3), pp. 1953-1954.

31. R.A. Cryan, R.T. Unwin, A.J. Massarella, M.J.N. Sibley, I. Garrett and N.M. Calvert.: 'Optical fibre digital PPM: theoretical and experimental results'. Presented at UK-USSR Symp. On Communications and applications, 1993.

32. M.J.N. Sibley and A.J. Massarella.: Detection of digital pilse position modulation over highly/slightly dispersive optical channels', Presented at SPIE Conf. on video communications and fiber optics networks, Berlin, 1993.

33. Sibley, M.J.N.: 'Design implications of high-speed digital PPM'. Presented at SPIE Conf. on Gigabit Networks, San Jose, CA, 1994.

34. Sibley, M.J.N.: ''Dicode pulse-position modulation: a novel coding scheme for optical-fibre communications,'' IEE Proc., Optoelectron., 2003,150, (2), pp. 125-131.

35. Sibley, M.J.N.: ''Suboptimal filtering in zero-guard, dicode PPM system operating over dispersive optical channels,'' IEE Proc., Optoelectron., 2004,151, (4), pp. 237-243.

36. Sibley, M.J.N.: ''Analysis of dicode pulse position modulation using a PINFET receiver and a slighty / highly dispersive optical channel,'' IEE Proc., Optoelectron., 2003,150, (3), pp. 205-209.

37. R.A. Cryan and Sibley, M.J.N.: ''Minimising intersymbol interference in optical-fibre dicode PPM systems,'' IEE Proc., Optoelectron., 2006,153, (3), pp. 93-99.

38. R.A. Charitopoulos, M.J.N. Sibley, 'Dicode Pulse Position Modulation Coder Simulation', School of Computing and Engineering Researchers' Conference, University of Huddersfield, Dec 2007.

39. R.A. Charitopoulos, M.J.N. Sibley, 'Experimental Verification of the Power Spectral Density of Dicode PPM', submitted for publication at IEEE Trans. On Comm.

Datasheets

National Semiconductor

August 2000

LM741
Operational Amplifier

General Description

The LM741 series are general purpose operational amplifiers which feature improved performance over industry standards like the LM709. They are direct, plug-in replacements for the 709C, LM201, MC1439 and 748 in most applications.

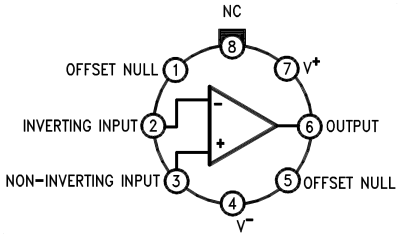
The amplifiers offer many features which make their application nearly foolproof: overload protection on the input and output, no latch-up when the common mode range is exceeded, as well as freedom from oscillations.

The LM741C is identical to the LM741/LM741A except that the LM741C has their performance guaranteed over a 0˚C to +70˚C temperature range, instead of −55˚C to +125˚C.

Features

Connection Diagrams

Metal Can Package

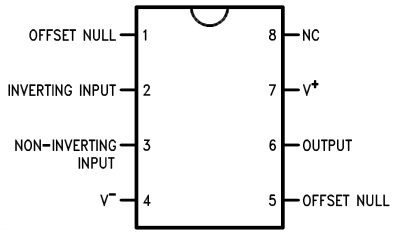


00934102

Note 1: LM741H is available per JM38510/10101

Order Number LM741H, LM741H/883 (Note 1)**, LM741AH/883 or LM741CH**
See NS Package Number H08C

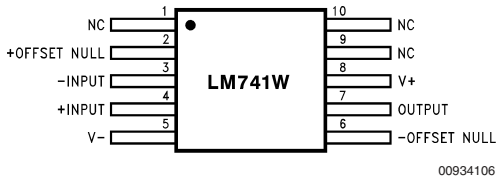
Dual-In-Line or S.O. Package



00934103

Order Number LM741J, LM741J/883, LM741CN
See NS Package Number J08A, M08A or N08E

Ceramic Flatpak

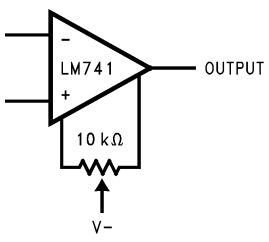


00934106

Order Number LM741W/883
See NS Package Number W10A

Typical Application

Offset Nulling Circuit



00934107

Absolute Maximum Ratings (Note 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

(Note 7)

	LM741A	LM741	LM741C
Supply Voltage	±22V	±22V	±18V
Power Dissipation (Note 3)	500 mW	500 mW	500 mW
Differential Input Voltage	±30V	±30V	±30V
Input Voltage (Note 4)	±15V	±15V	±15V
Output Short Circuit Duration	Continuous	Continuous	Continuous
Operating Temperature Range	−55˚C to +125˚C	−55˚C to +125˚C	0˚C to +70˚C
Storage Temperature Range	−65˚C to +150˚C	−65˚C to +150˚C	−65˚C to +150˚C
Junction Temperature	150˚C	150˚C	100˚C
Soldering Information			
N-Package (10 seconds)	260˚C	260˚C	260˚C
J- or H-Package (10 seconds)	300˚C	300˚C	300˚C
M-Package			
Vapor Phase (60 seconds)	215˚C	215˚C	215˚C
Infrared (15 seconds)	215˚C	215˚C	215˚C
See AN-450 “Surface Mounting Methods and Their Effect on Product Reliability” for other methods of soldering surface mount devices.			
ESD Tolerance (Note 8)	400V	400V	400V

Electrical Characteristics (Note 5)

Parameter	Conditions	LM741A			LM741			LM741C			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Input Offset Voltage	$T_A = 25˚C$										
	$R_S \le 10\ k\Omega$					1.0	5.0		2.0	6.0	mV
	$R_S \le 50\Omega$		0.8	3.0							mV
	$T_{AMIN} \le T_A \le T_{AMAX}$										
	$R_S \le 50\Omega$			4.0							mV
	$R_S \le 10\ k\Omega$						6.0			7.5	mV
Average Input Offset Voltage Drift				15							µV/˚C
Input Offset Voltage Adjustment Range	$T_A = 25˚C$, $V_S = \pm 20V$	±10				±15			±15		mV
Input Offset Current	$T_A = 25˚C$		3.0	30		20	200		20	200	nA
	$T_{AMIN} \le T_A \le T_{AMAX}$			70		85	500			300	nA
Average Input Offset Current Drift				0.5							nA/˚C
Input Bias Current	$T_A = 25˚C$		30	80		80	500		80	500	nA
	$T_{AMIN} \le T_A \le T_{AMAX}$			0.210			1.5			0.8	µA
Input Resistance	$T_A = 25˚C$, $V_S = \pm 20V$	1.0	6.0		0.3	2.0		0.3	2.0		MΩ
	$T_{AMIN} \le T_A \le T_{AMAX}$, $V_S = \pm 20V$	0.5									MΩ
Input Voltage Range	$T_A = 25˚C$							±12	±13		V
	$T_{AMIN} \le T_A \le T_{AMAX}$				±12	±13					V

Electrical Characteristics (Note 5) (Continued)

Parameter	Conditions	LM741A			LM741			LM741C			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Large Signal Voltage Gain	$T_A = 25^\circ C$, $R_L \geq 2\ k\Omega$										
	$V_S = \pm 20V$, $V_O = \pm 15V$	50									V/mV
	$V_S = \pm 15V$, $V_O = \pm 10V$				50	200		20	200		V/mV
	$T_{AMIN} \leq T_A \leq T_{AMAX}$,										
	$R_L \geq 2\ k\Omega$,										
	$V_S = \pm 20V$, $V_O = \pm 15V$	32									V/mV
	$V_S = \pm 15V$, $V_O = \pm 10V$				25			15			V/mV
	$V_S = \pm 5V$, $V_O = \pm 2V$	10									V/mV
Output Voltage Swing	$V_S = \pm 20V$										
	$R_L \geq 10\ k\Omega$	±16									V
	$R_L \geq 2\ k\Omega$	±15									V
	$V_S = \pm 15V$										
	$R_L \geq 10\ k\Omega$				±12	±14		±12	±14		V
	$R_L \geq 2\ k\Omega$				±10	±13		±10	±13		V
Output Short Circuit	$T_A = 25^\circ C$	10	25	35		25			25		mA
Current	$T_{AMIN} \leq T_A \leq T_{AMAX}$	10		40							mA
Common-Mode	$T_{AMIN} \leq T_A \leq T_{AMAX}$										
Rejection Ratio	$R_S \leq 10\ k\Omega$, $V_{CM} = \pm 12V$				70	90		70	90		dB
	$R_S \leq 50\Omega$, $V_{CM} = \pm 12V$	80	95								dB
Supply Voltage Rejection	$T_{AMIN} \leq T_A \leq T_{AMAX}$,										
Ratio	$V_S = \pm 20V$ to $V_S = \pm 5V$										
	$R_S \leq 50\Omega$	86	96								dB
	$R_S \leq 10\ k\Omega$				77	96		77	96		dB
Transient Response	$T_A = 25^\circ C$, Unity Gain										
Rise Time			0.25	0.8		0.3			0.3		µs
Overshoot			6.0	20		5			5		%
Bandwidth (Note 6)	$T_A = 25^\circ C$	0.437	1.5								MHz
Slew Rate	$T_A = 25^\circ C$, Unity Gain	0.3	0.7			0.5			0.5		V/µs
Supply Current	$T_A = 25^\circ C$					1.7	2.8		1.7	2.8	mA
Power Consumption	$T_A = 25^\circ C$										
	$V_S = \pm 20V$		80	150							mW
	$V_S = \pm 15V$					50	85		50	85	mW
LM741A	$V_S = \pm 20V$										
	$T_A = T_{AMIN}$			165							mW
	$T_A = T_{AMAX}$			135							mW
LM741	$V_S = \pm 15V$										
	$T_A = T_{AMIN}$					60	100				mW
	$T_A = T_{AMAX}$					45	75				mW

Note 2: "Absolute Maximum Ratings" indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not guarantee specific performance limits.

Electrical Characteristics (Note 5) (Continued)

Note 3: For operation at elevated temperatures, these devices must be derated based on thermal resistance, and T_j max. (listed under "Absolute Maximum Ratings"). $T_j = T_A + (\theta_{jA} P_D)$.

Thermal Resistance	Cerdip (J)	DIP (N)	HO8 (H)	SO-8 (M)
θ_{jA} (Junction to Ambient)	100˚C/W	100˚C/W	170˚C/W	195˚C/W
θ_{jC} (Junction to Case)	N/A	N/A	25˚C/W	N/A

Note 4: For supply voltages less than ±15V, the absolute maximum input voltage is equal to the supply voltage.

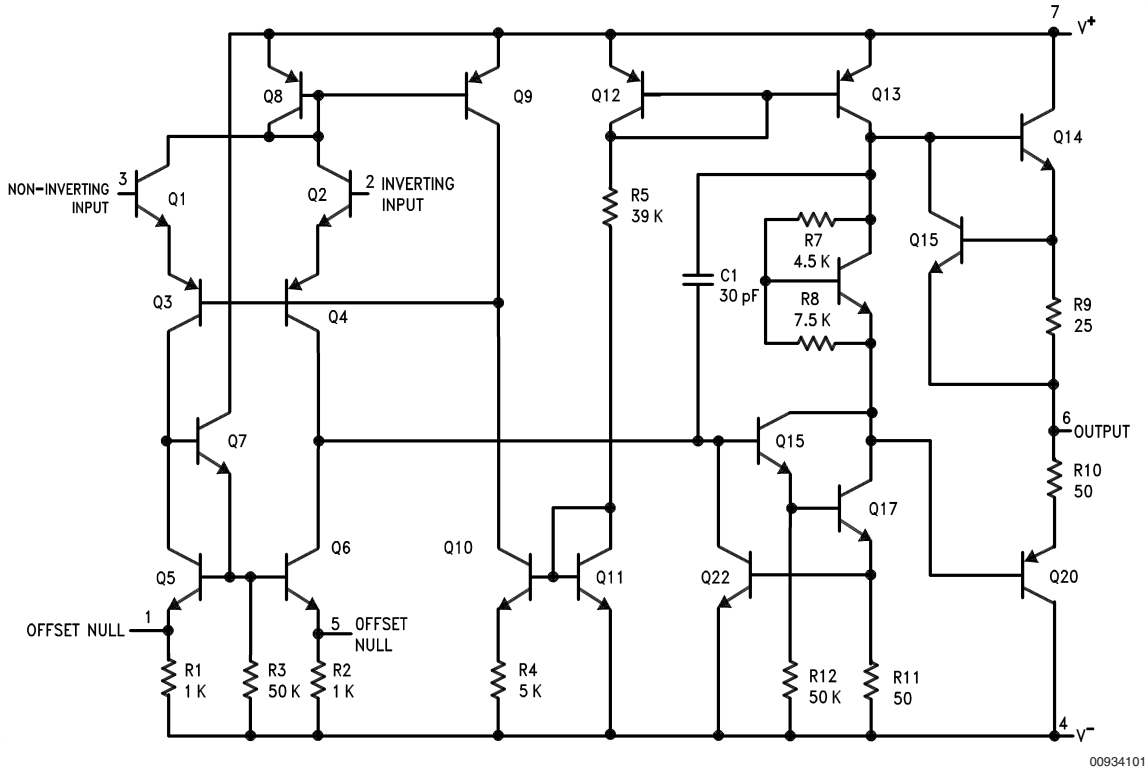
Note 5: Unless otherwise specified, these specifications apply for V_S = ±15V, −55˚C ≤ T_A ≤ +125˚C (LM741/LM741A). For the LM741C/LM741E, these specifications are limited to 0˚C ≤ T_A ≤ +70˚C.

Note 6: Calculated value from: BW (MHz) = 0.35/Rise Time(μs).

Note 7: For military specifications see RETS741X for LM741 and RETS741AX for LM741A.

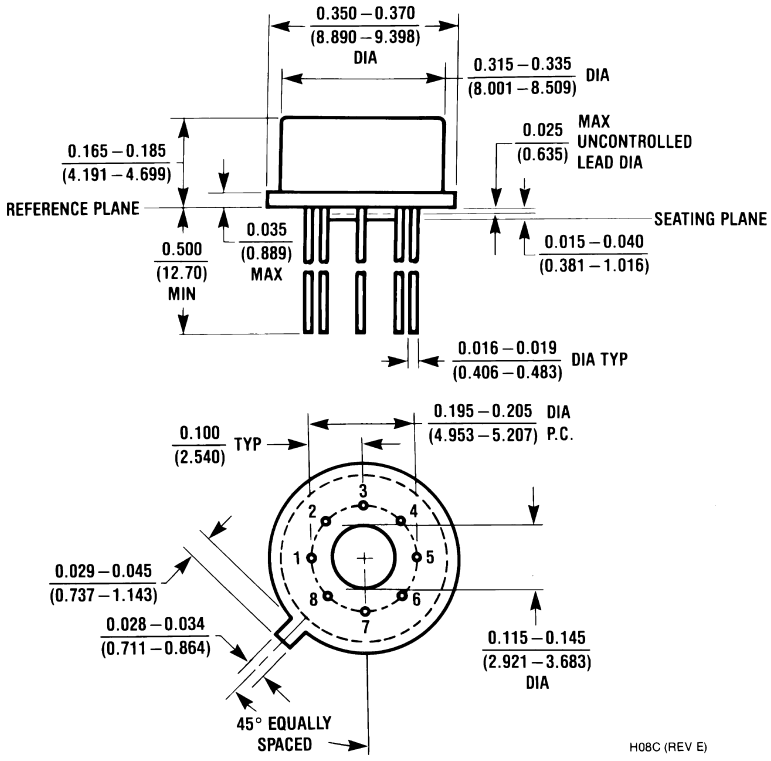
Note 8: Human body model, 1.5 kΩ in series with 100 pF.

Schematic Diagram



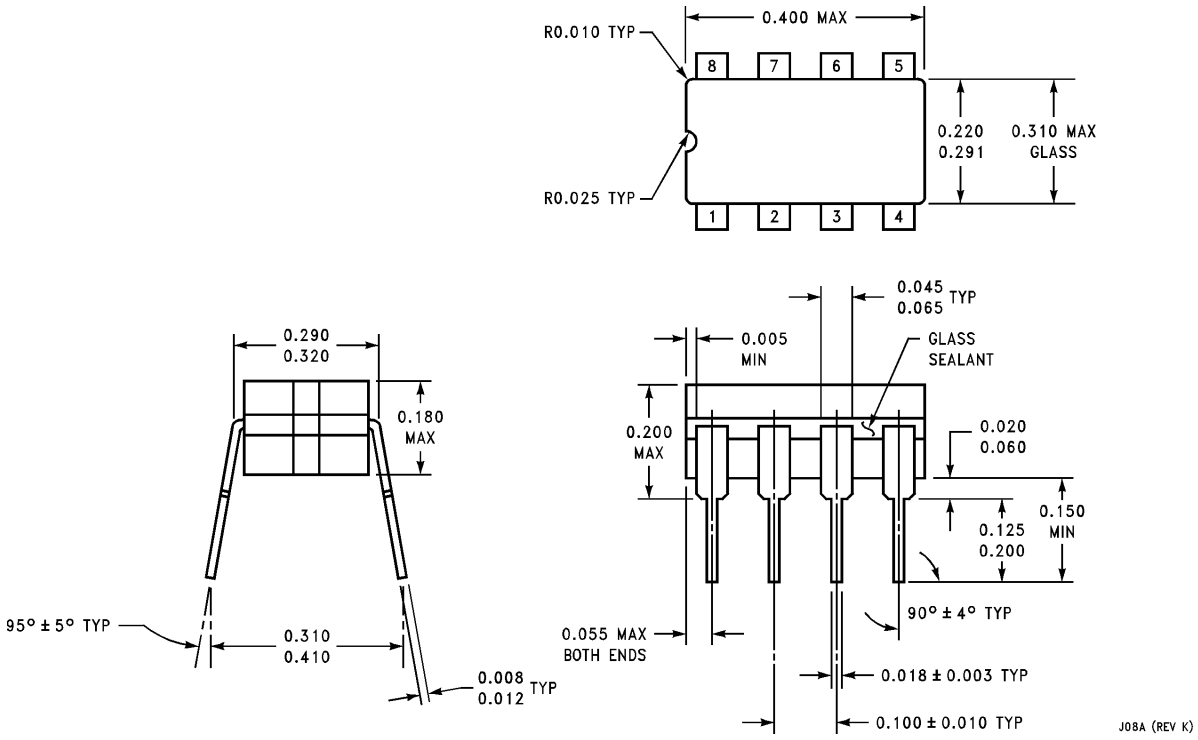
Physical Dimensions inches (millimeters)

unless otherwise noted

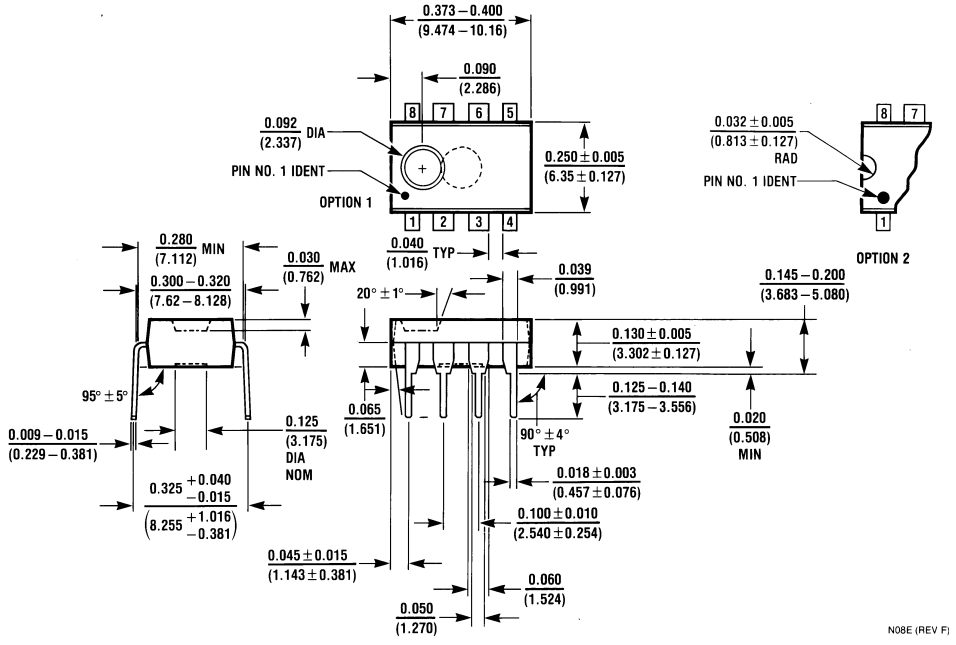


Metal Can Package (H)
Order Number LM741H, LM741H/883, LM741AH/883, LM741AH-MIL or LM741CH
NS Package Number H08C

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)

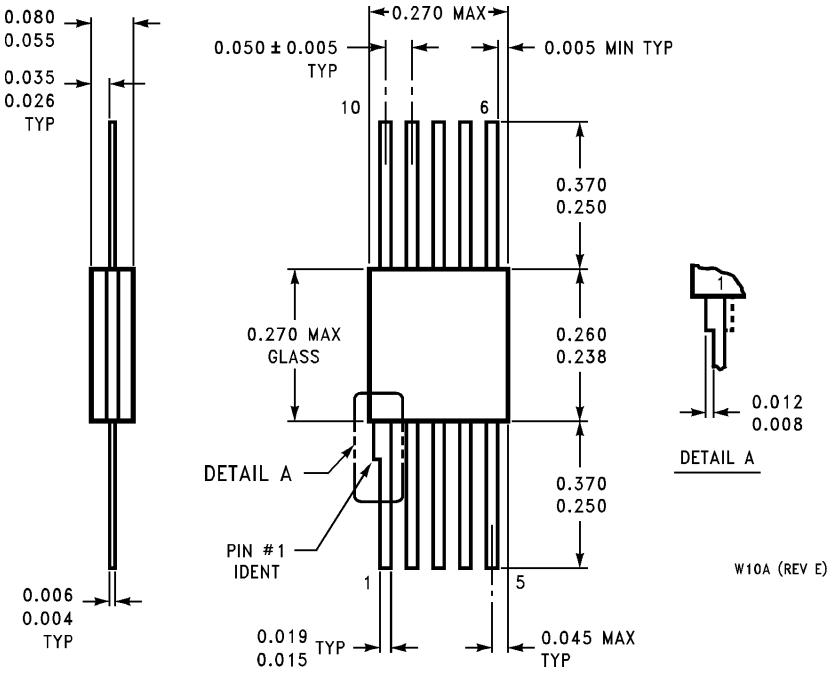


Ceramic Dual-In-Line Package (J)
Order Number LM741J/883
NS Package Number J08A



Dual-In-Line Package (N)
Order Number LM741CN
NS Package Number N08E

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



10-Lead Ceramic Flatpak (W)
Order Number LM741W/883, LM741WG-MPR or LM741WG/883
NS Package Number W10A

National Semiconductor
Americas Customer
Support Center
Email: new.feedback@nsc.com
Tel: 1-800-272-9959

www.national.com

National Semiconductor
Europe Customer Support Center
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

National Semiconductor
Asia Pacific Customer
Support Center
Email: ap.support@nsc.com

National Semiconductor
Japan Customer Support Center
Fax: 81-3-5639-7507
Email: jpn.feedback@nsc.com
Tel: 81-3-5639-7560

19-1673 Rev 0; 9/00

MAXIM

MAX2605–MAX2609 Evaluation Kits

General Description

The MAX2605–MAX2609 evaluation kits (EV kits) simplify evaluation of this family of voltage-controlled oscillators (VCOs). These kits enable testing of the devices' performance and require no additional support circuitry. Both signal outputs use SMA connectors to facilitate connection to RF test equipment.

These EV kits are fully assembled and tested. Their oscillation frequencies are set to approximately the midrange of the respective VCOs.

Component Suppliers

SUPPLIER	PHONE	FAX	WEBSITE
AVX	803-946-0690	803-626-3123	avx-corp.com
Coilcraft	847-639-6400	847-639-1469	coilcraft.com
EFJohnson	402-474-4800	402-474-4858	efjohnson.com
Murata	814-237-1431	814-238-0490	murata.com

MAX2605 Component List

DESIGNATION	QTY	DESCRIPTION
C1, C4	2	1000pF ±5% ceramic capacitors (0603) Murata GRM39COH102J025A or AVX 06035A102JAT2A
C2, C3	2	12pF ±5% ceramic capacitors (0603) Murata GRM39COG120J050A
C5	1	10µF ±10%, 16V tantalum capacitor AVX TAJC106K016
R1	1	270Ω ±5% resistor (0603)
R2, R3	2	Not installed
R4	1	10Ω ±5% resistor (0603)
L1, L2	2	Not installed
L4, L5	2	680nH inductors Coilcraft 1008CS-681XJBC
L3	1	1.2µH inductor Coilcraft 1206CS-122XJBC
U1	1	MAX2605EUT
J1, J2	2	SMA connectors (edge mount) EFJohnson 142-0701-801 or Digi-Key J502-ND
VCC,GND, TUNE	6	Test points Digi-Key 5000K-ND
None	1	MAX2605/6/7 EV kit circuit board
None	1	MAX2605–MAX2609 data sheet

Features

- **Easy Evaluation**
- **Complete, Tunable VCO Test Board with Tank Circuit**
- **Low Phase Noise**
- **Fully Assembled and Tested**

Ordering Information

PART	TEMP. RANGE	IC PACKAGE
MAX2605EVKIT	-40°C to +85°C	6 SOT23-6
MAX2606EVKIT	-40°C to +85°C	6 SOT23-6
MAX2607EVKIT	-40°C to +85°C	6 SOT23-6
MAX2608EVKIT	-40°C to +85°C	6 SOT23-6
MAX2609EVKIT	-40°C to +85°C	6 SOT23-6

MAX2606 Component List

DESIGNATION	QTY	DESCRIPTION
C1, C4	2	1000pF ±5% ceramic capacitors (0603) Murata GRM39COH102J025A or AVX 06035A102JAT2A
C2, C3	2	4.7pF ±0.25% ceramic capacitors (0603) Murata GRM39COG4R7C050A
C5	1	10µF ±10%, 16V tantalum capacitor AVX TAJC106K016
R1	1	270Ω ±5% resistor (0603)
R2, R3	2	Not installed
R4	1	10Ω ±5% resistor (0603)
L1, L2	2	Not installed
L3	1	270nH ±2% inductor Coilcraft 1008CS-271XJBC
L4, L5	2	330nH inductor Coilcraft 1008CS-331XJBC
U1	1	MAX2606EUT
J1, J2	2	SMA connectors (edge mount) EFJohnson 142-0701-801 or Digi-Key J502-ND
VCC,GND, TUNE	6	Test points
None	1	MAX2605/6/7 EV kit circuit board
None	1	MAX2605–MAX2609 data sheet

MAX2607 Component List

DESIGNATION	QTY	DESCRIPTION
C1, C4	2	1000pF ±5% ceramic capacitors (0603) Murata GRM39COH102J025A or AVX 06035A102JAT2A
C2, C3	2	3pF ±0.25% ceramic capacitors (0603) Murata GRM39COG030C050A
C5	1	10µF ±10%, 16V tantalum capacitor AVX TAJC106K016
R1	1	270Ω ±5% resistor (0603)
R2, R3	2	Not installed
R4	1	10Ω ±5% resistor (0603)
L1, L2	2	Not installed
L3	1	68nH ±2% inductor Coilcraft 1008CS-680XJBC
L4, L5	2	120nH inductors Coilcraft 1008CS-121XJBC
U1	1	MAX2607EUT
J1, J2	2	SMA connectors (edge mount) EFJohnson 142-0701-801 or Digi-Key J502-ND
VCC,GND, TUNE	6	Test points
None	1	MAX2605/6/7 EV kit circuit board
None	1	MAX2605–MAX2609 data sheet

MAX2608 Component List

DESIGNATION	QTY	DESCRIPTION
C1	1	1000pF ±5% ceramic capacitor (0603) Murata GRM39COH102J025A or AVX 06035A102JAT2A
C2, C3	2	1pF ±0.25% ceramic capacitors (0603) Murata GRM39COG010C050A
C4, C6	2	100pF ±5% ceramic capacitors (0603) Murata GRM39COG101J050A
C5	1	10µF ±10%, 16V tantalum capacitor AVX TAJC106K016
R1	1	270Ω ±5% resistor (0603)
R2, R3	2	Not installed
R4	1	10Ω ±5% resistor (0603)
L1	1	22nH ±2% inductor Coilcraft 0805CS-220XGBC
L4, L5	2	68nH ±2% inductors Coilcraft 0805CS-680XJBC
U1	1	MAX2608EUT
J1, J2	2	SMA connectors (edge mount) EFJohnson 142-0701-801 or Digi-Key J502-ND
VCC,GND, TUNE	6	Test points Digi-Key 5000K-ND
None	1	MAX2608/9 EV kit circuit board
None	1	MAX2605–MAX2609 data sheet

Quick Start

The MAX2605–MAX2609 evaluation kits are fully assembled and factory tested. Follow the instructions in the *Connections and Setup* section for proper device evaluation.

Test Equipment Required

- Low-noise power supplies (these are recommended for oscillator noise measurement). Noise or ripple will frequency-modulate the oscillator and cause spectral spreading. Batteries can be used in place of power supplies, if necessary.
  - Use a DC power supply capable of supplying +2.7V to +5.5V. Alternatively, use two or three 1.5V batteries.
  - Use a DC power supply capable of supplying +0.4V to +2.4V, continuously variable, for TUNE. Alternatively, use two 1.5V batteries with a resistive voltage divider or potentiometer.
- An RF spectrum analyzer that covers the operating frequency range of the MAX2605–MAX2609
- A 50Ω coaxial cable with SMA connectors
- An ammeter (optional)

Connections and Setup

1) Connect a DC supply (preset to +3V) to the V_{CC} and GND terminals (through an ammeter, if desired) on the EV kit.
2) Turn on the DC supply. If used, the ammeter reading

MAX2609 Component List

DESIGNATION	QTY	DESCRIPTION
C1	1	1000pF ±5% ceramic capacitors (0603) Murata GRM39COH102J025A or AVX 06035A102JAT2A
C2, C3	2	1pF ±0.25% ceramic capacitors (0603) Murata GRM39COG010C050A
C4, C6	2	100pF ±5% ceramic capacitors (0603) Murata GRM39COG101J050A
C5	1	10µF ±10%, 16V tantalum capacitor AVX TAJC106K016
R1	1	270Ω ±5% resistor (0603)
R2, R3	2	Not installed
R4	1	10Ω ±5% resistor (0603)
L1	1	8.2nH ±2% inductor Coilcraft 0805CS-080XGBC
L4, L5	2	27nH ±2% inductors Coilcraft 0805CS-270XJBC
U1	1	MAX2609EUT
J1, J2	2	SMA connectors (edge mount) EFJohnson 142-0701-801 or Digi-Key J502-ND
VCC,GND, TUNE	6	Test points Digi-Key 5000K-ND
None	1	MAX2608/9 EV kit circuit board
None	1	MAX2605–MAX2609 data sheet

approximates the typical operating current specified in the MAX2605–MAX2609 data sheet.

3) Connect the VCO output (OUT+ or OUT-) to a spectrum analyzer with a 50Ω coaxial cable.

4) Apply a positive variable DC voltage between 0.4V and 2.4V to TUNE.

5) Check the tuning bandwidth on the spectrum analyzer by varying the tuning voltage (+0.4V to +2.4V).

Layout Considerations

The EV kit PC board can serve as a guide for laying out a board using the MAX2605–MAX2609. Generally, the VCC pin on the PC board should have a decoupling capacitor placed close to the IC. This minimizes noise coupling from the supply. Also, place the VCO as far away as possible from the noisy section of a larger system, such as a switching regulator or digital circuits.

The VCO's performance is strongly dependent on the availability of the external tuning inductor. For best performance, use high-Q components and choose their values carefully. To minimize the effects of parasitic elements, which degrade circuit performance, place the tuning inductor and C_{BYP} close to the VCO. For higher-frequency versions, include the parasitic PC board inductance and capacitance when calculating the oscillation frequency. In addition, remove the ground plane around and under the tuning inductor to minimize the effect of parasitic capacitance.

Noise on TUNE translates into FM noise on the outputs; therefore, keep the trace between TUNE and the control circuitry as short as possible. If necessary, use an RC filter to further suppress noise, as done on the EV kits.

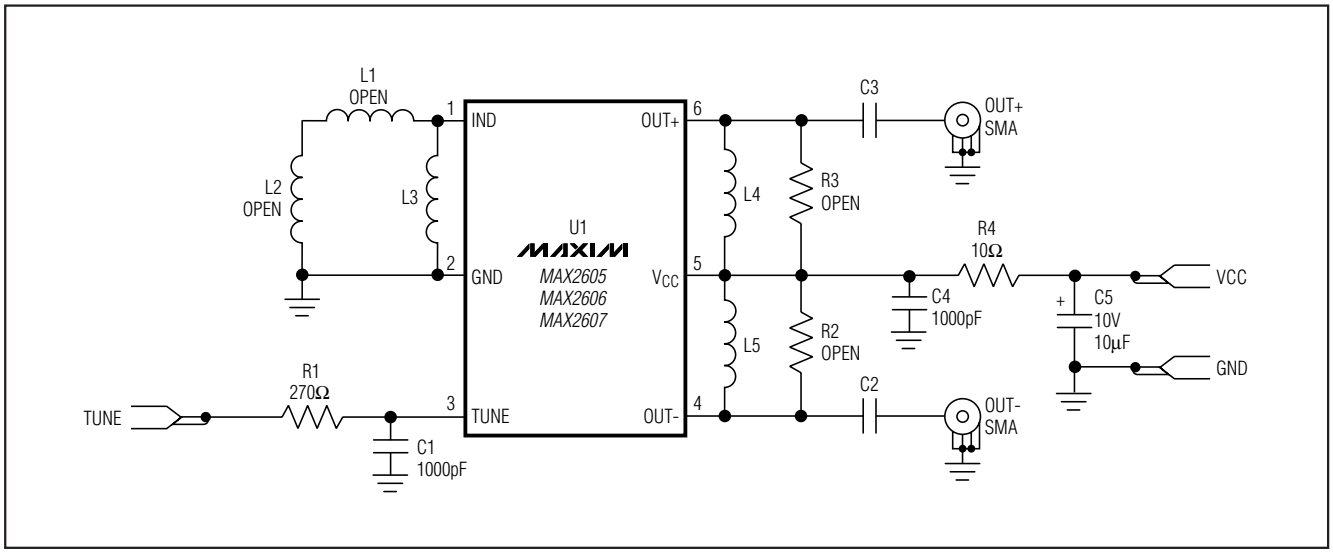


Figure 1. MAX2605/MAX2606/MAX2607 EV Kits Schematic

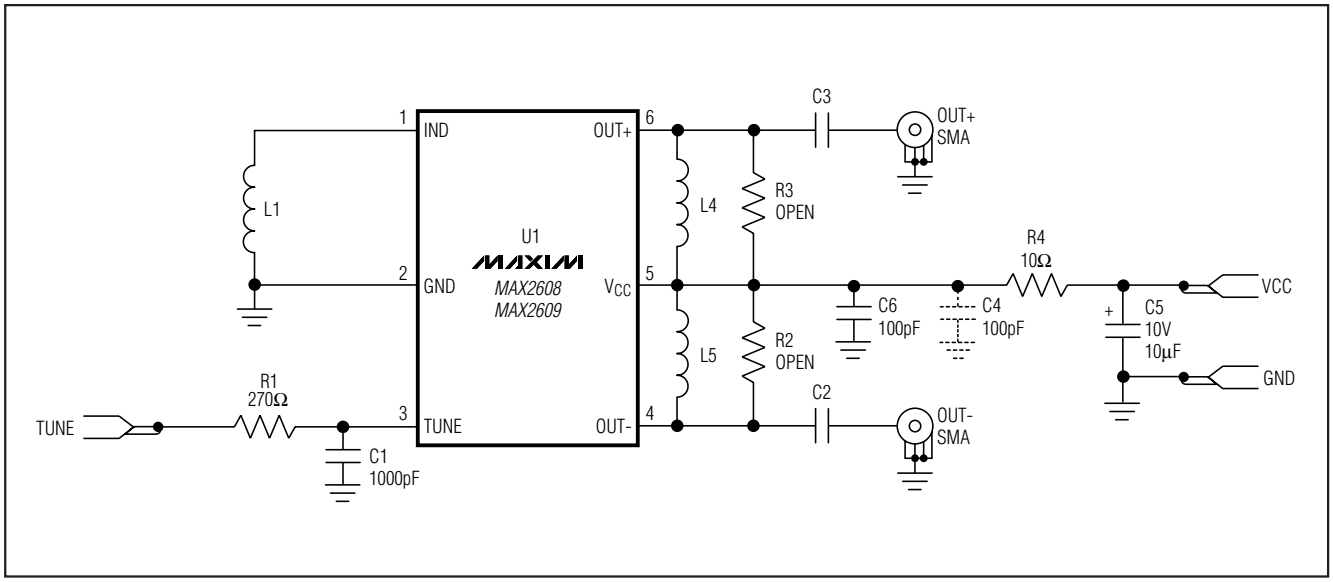


Figure 2. MAX2608/MAX2609 EV Kits Schematic

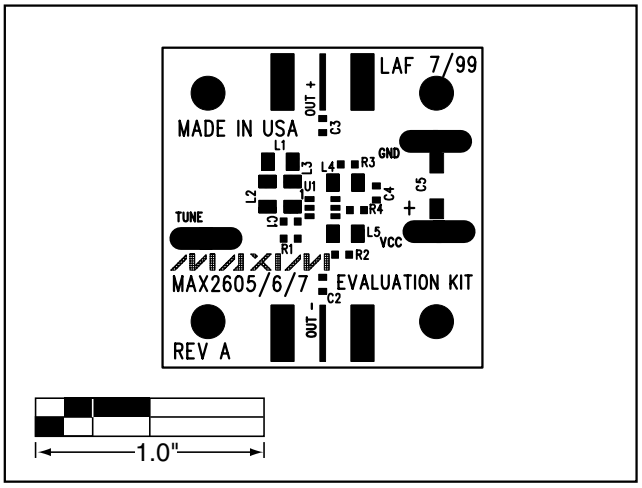


Figure 3. MAX2605/MAX2606/MAX2607 EV Kits Component Placement Guide—Top Silk Screen

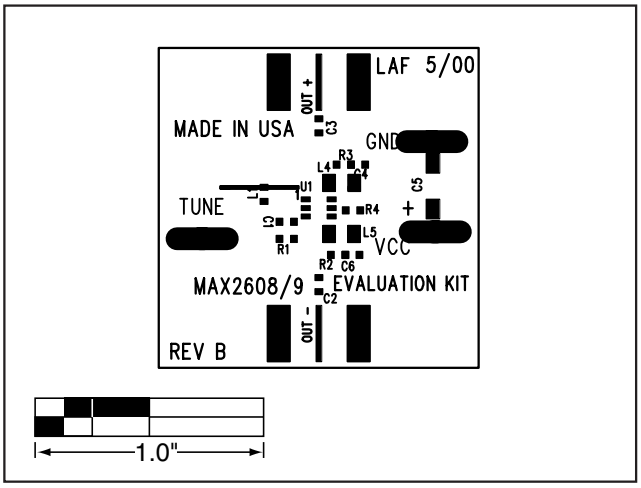


Figure 4. MAX2608/MAX2609 EV Kits Component Placement Guide—Top Silk Screen

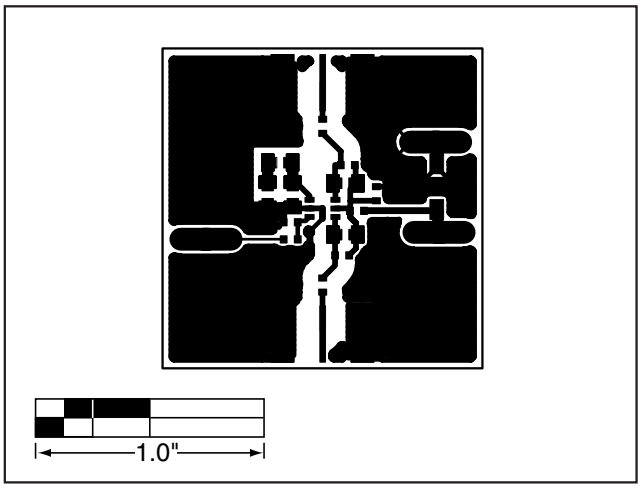


Figure 5. MAX2605/MAX2606/MAX2607 EV Kits PC Board Layout—Component Side

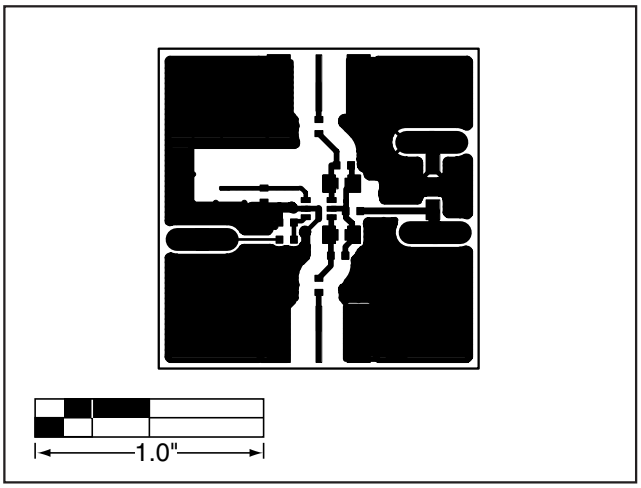


Figure 6. MAX2608/MAX2609 EV Kits PC Board Layout—Component Side

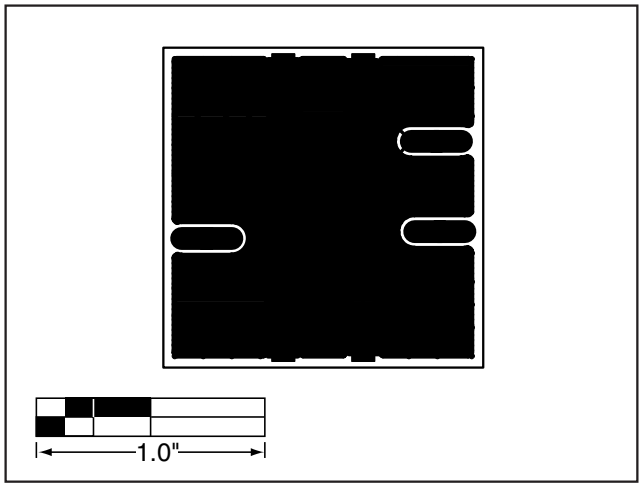


Figure 7. MAX2605/MAX2606/MAX2607/MAX2608/MAX2609 EV Kits PC Board Layout—Ground Plane

19-4757; Rev 3; 10/98

EVALUATION KIT MANUAL FOLLOWS DATA SHEET

740MHz, Low-Noise, Low-Distortion Op Amps in SOT23-5

General Description

The MAX4104/MAX4105/MAX4304/MAX4305 op amps feature ultra-high speed, low noise, and low distortion in a SOT23 package. The unity-gain-stable MAX4104 requires only 20mA of supply current while delivering 625MHz bandwidth and 400V/µs slew rate. The MAX4304, compensated for gains of +2V/V or greater, delivers a 730MHz bandwidth and a 1000V/µs slew rate. The MAX4105 is compensated for a minimum gain of +5V/V and delivers a 410MHz bandwidth and a 1400V/sec slew rate. The MAX4305 has +10V/V minimum gain compensation and delivers a 340MHz bandwidth and a 1400V/µs slew rate.

Low voltage noise density of 2.1nV/√Hz and -88dBc spurious-free dynamic range make these devices ideal for low-noise/low-distortion video and telecommunications applications. These op amps also feature a wide output voltage swing of ±3.7V and ±70mA output current-drive capability. For space-critical applications, they are available in a miniature 5-pin SOT23 package.

Applications

Video ADC Preamp

Pulse/RF Telecom Applications

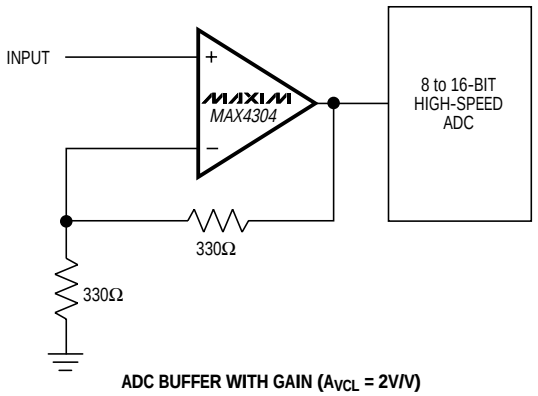
Video Buffers and Cable Drivers

Ultrasound

Active Filters

ADC Input Buffers

Typical Application Circuit



ADC BUFFER WITH GAIN (A_{VCL} = 2V/V)

Features

- **Low 2.1nV/√Hz Voltage Noise Density**
- **Ultra-High 740MHz -3dB Bandwidth (MAX4304, A_{VCL} = 2V/V)**
- **100MHz 0.1dB Gain Flatness (MAX4104/4105)**
- **1400V/µs Slew Rate (MAX4105/4305)**
- **-88dBc SFDR (5MHz, R_L = 100Ω) (MAX4104/4304)**
- **High Output Current Drive: ±70mA**
- **Low Differential Gain/Phase Error: 0.01%/0.01° (MAX4104/4304)**
- **Low ±1mV Input Offset Voltage**
- **Available in Space-Saving 5-Pin SOT23 Package**

Selector Guide

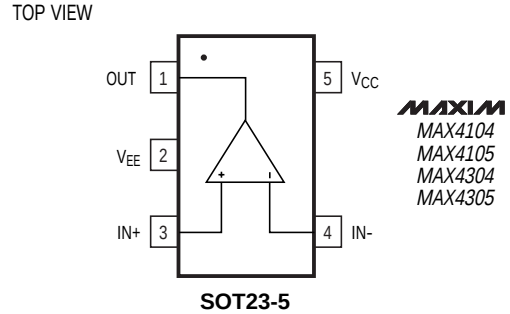
PART	MINIMUM STABLE GAIN (V/V)	BANDWIDTH (MHz)	PIN-PACKAGE
MAX4104	1	625	5-pin SOT23, 8-pin SO
MAX4304	2	740	5-pin SOT23, 8-pin SO
MAX4105	5	410	5-pin SOT23, 8-pin SO
MAX4305	10	340	5-pin SOT23, 8-pin SO

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE	SOT TOP MARK
MAX4104ESA	-40°C to +85°C	8 SO	—
MAX4104EUK-T	-40°C to +85°C	5 SOT23-5	ACCO

Ordering Information continued at end of data sheet.

Pin Configurations



SOT23-5

Pin Configurations continued at end of data sheet.

ABSOLUTE MAXIMUM RATINGS

Supply Voltage (V_{CC} to V_{EE}).....+12V
Voltage on Any Pin to Ground.....(V_{EE} - 0.3V) to (V_{CC} + 0.3V)
Short-Circuit Duration (V_{OUT} to GND).....Continuous
Continuous Power Dissipation (T_A = +70°C)
5-pin SOT23 (derate 7.1mW/°C above +70°C).....571mW
8-pin SO (derate 5.9mW/°C above +70°C).....471mW
Operating Temperature Range .....-40°C to +85°C
Storage Temperature Range .....-65°C to +150°C
Lead Temperature (soldering, 10sec) .....+300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = +5V, V_{EE} = -5V, V_{CM} = 0, R_L = 100kΩ, T_A = T_{MIN} to T_{MAX}, unless otherwise noted. Typical values are at T_A = +25°C.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Operating Supply Voltage Range	V_{CC}/V_{EE}	Guaranteed by PSRR test		±3.5	±5	±5.5	V
Input Offset Voltage	V_{OS}	V_{OUT} = 0	MAX4_0_ESA		1	6	mV
			MAX4_0_EUK		1	8	
Input Offset-Voltage Drift	TCV_{OS}				2.5		µV/°C
Input Bias Current	I_B				32	70	µA
Input Offset Current	I_{OS}				0.5	5.0	µA
Differential Input Resistance	R_{IN}	-0.8V ≤ V_{IN} ≤ 0.8V			6		kΩ
Common-Mode Input Resistance	R_{IN}	Either input			1.5		MΩ
Input Common-Mode Voltage Range	V_{CM}	Guaranteed by CMRR test		-2.8		+4.1	V
Common-Mode Rejection Ratio	CMRR	-2.8V ≤ V_{CM} ≤ 4.1V		80	95		dB
Positive Power-Supply Rejection Ratio	PSSR+	V_{CC} = 3.5V to 5.5V		75	85		dB
Negative Power-Supply Rejection Ratio	PSRR-	V_{EE} = -3.5V to -5.5V		55	65		dB
Quiescent Supply Current	I_S	V_{OUT} = 0			20	27	mA
Open-Loop Gain	A_{VOL}	-2.8V ≤ V_{OUT} ≤ 2.8V, R_L = 100Ω		55	65		dB
Output Voltage Swing	V_{OUT}	R_L = 100kΩ		±3.5	-3.7 to +3.8		V
		R_L = 100Ω		±3.0	-3.5 to +3.4		
Output Current Drive	I_{OUT}	R_L = 30Ω		±53	±70		mA
Short-Circuit Output Current	I_{SC}	R_L = short to ground			80		mA
Open-Loop Output Impedance	Z_{OUT}				9		Ω

AC ELECTRICAL CHARACTERISTICS

(V_{CC} = +5V, V_{EE} = -5V, V_{CM} = 0, R_L = 100Ω; A_V = +1V/V for MAX4104, +2V/V for MAX4304, +5V/V for MAX4105, +10V/V for MAX4305; T_A = +25°C; unless otherwise noted.)

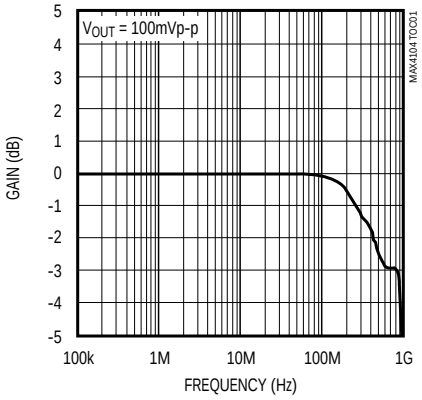
PARAMETER	SYMBOL	CONDITIONS			MIN	TYP	MAX	UNITS
-3dB Bandwidth	$BW_{(-3dB)}$	V_{OUT} = 100mVp-p		MAX4104		625		MHz
				MAX4304		740		
				MAX4105		410		
				MAX4305		340		
0.1dB Bandwidth	$BW_{(0.1)}$	V_{OUT} = 100mVp-p		MAX4104		100		MHz
				MAX4304		60		
				MAX4105		80		
				MAX4305		70		
Full-Power Bandwidth	FPBW	V_{OUT} = 2Vp-p		MAX4104		115		MHz
				MAX4304		285		
				MAX4105		370		
				MAX4305		320		
Slew Rate	SR	V_{OUT} = 2Vp-p		MAX4104		400		V/µs
				MAX4304		1000		
				MAX4105		1400		
				MAX4305		1400		
Settling Time to 0.1%	t_S	V_{OUT} = 2Vp-p		to 0.1%		20		ns
				to 0.01%		25		
Spurious-Free Dynamic Range	SFDR	V_{OUT} = 2Vp-p	MAX4104/ MAX4304	f_C = 5MHz		-88		dBc
				f_C = 20MHz		-67		
			MAX4105/ MAX4305	f_C = 5MHz		-74		
				f_C = 20MHz		-61		
Differential Gain Error	DG	NTSC, R_L = 150Ω		MAX4104/MAX4304		0.01		%
				MAX4105/MAX4305		0.02		
Differential Phase Error	DP	NTSC, R_L = 150Ω		MAX4104/MAX4304		0.01		degrees
				MAX4105/MAX4305		0.02		
Input Voltage Noise Density	e_n	f = 1MHz				2.1		nV/√Hz
Input Current Noise Density	i_n	f = 1MHz				3.1		pA/√Hz
Output Impedance	Z_{OUT}	f = 10MHz				1		Ω

740MHz, Low-Noise, Low-Distortion Op Amps in SOT23-5

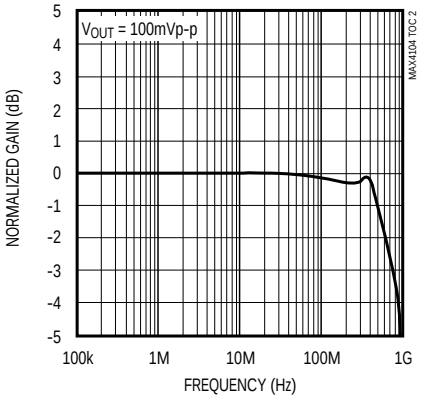
Typical Operating Characteristics

(V_{CC} = +5V, V_{EE} = -5V, R_F = 330Ω, R_L = 100Ω, T_A = +25°C, unless otherwise noted.)

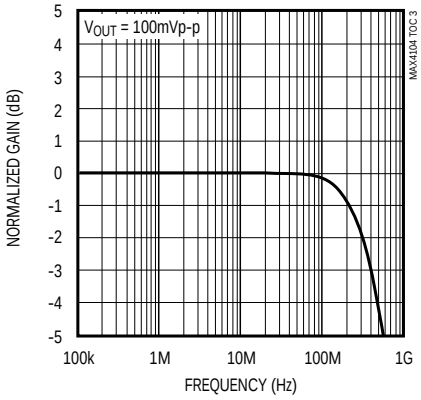
MAX4104
SMALL-SIGNAL GAIN
vs. FREQUENCY (A_{VCL} = +1)



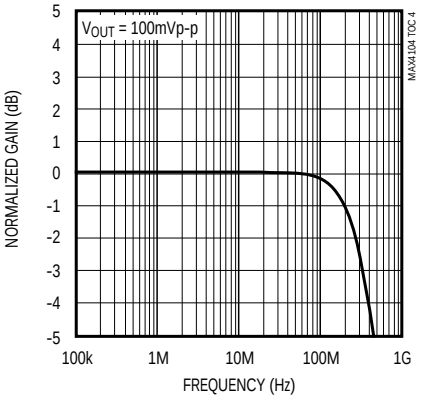
MAX4304
SMALL-SIGNAL GAIN
vs. FREQUENCY (A_{VCL} = +2)



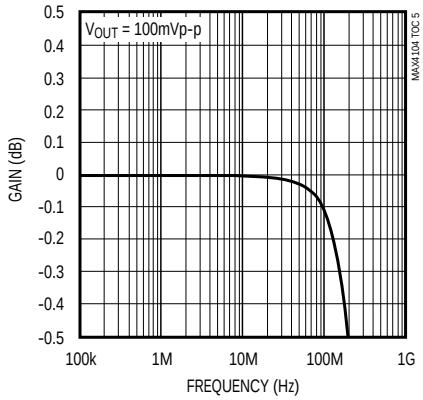
MAX4105
SMALL-SIGNAL GAIN
vs. FREQUENCY (A_{VCL} = +5)



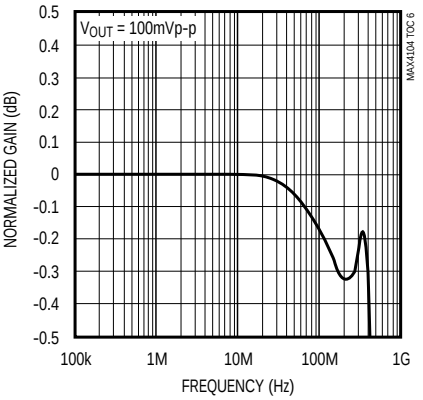
MAX4305
SMALL-SIGNAL GAIN
vs. FREQUENCY (A_{VCL} = +10)



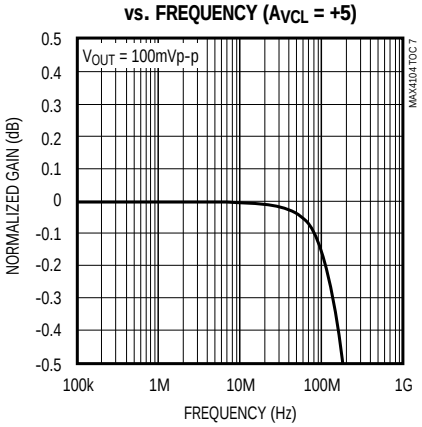
MAX4104
GAIN FLATNESS
vs. FREQUENCY (A_{VCL} = +1)



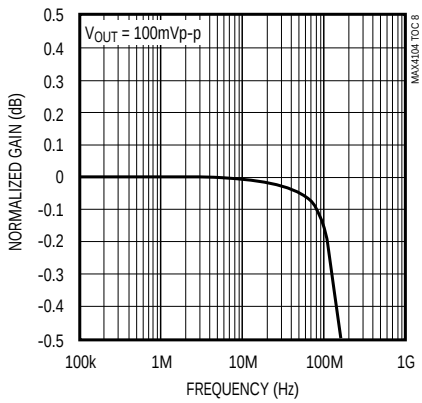
MAX4304
GAIN FLATNESS
vs. FREQUENCY (A_{VCL} = +2)



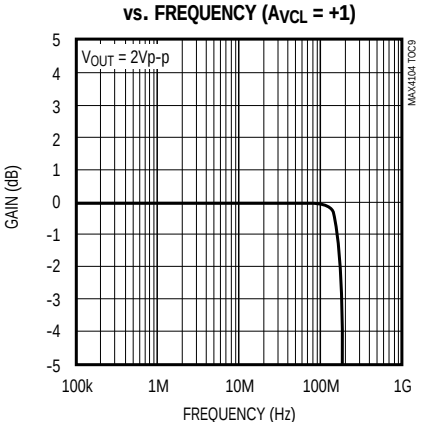
MAX4105
GAIN FLATNESS
vs. FREQUENCY (A_{VCL} = +5)



MAX4305
GAIN FLATNESS
vs. FREQUENCY (A_{VCL} = +10)



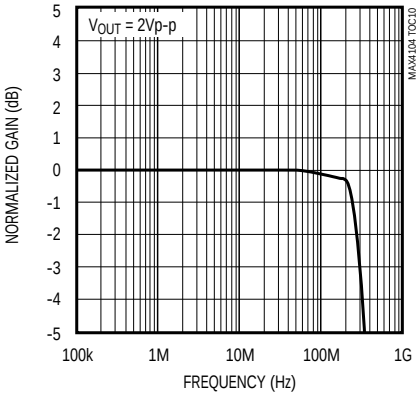
MAX4104
LARGE-SIGNAL GAIN
vs. FREQUENCY (A_{VCL} = +1)



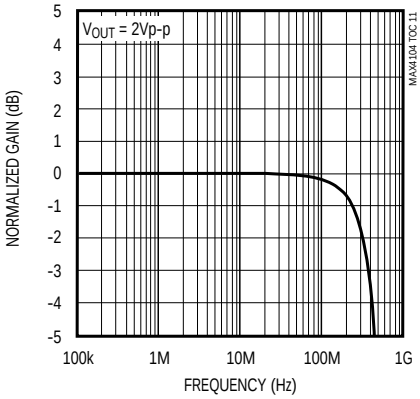
Typical Operating Characteristics (continued)

(V_{CC} = +5V, V_{EE} = -5V, R_F = 330Ω, R_L = 100Ω, T_A = +25°C, unless otherwise noted.)

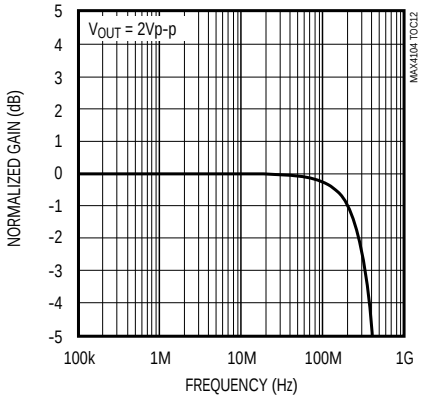
MAX4304
LARGE-SIGNAL GAIN
vs. FREQUENCY (A_{VCL} = +2)



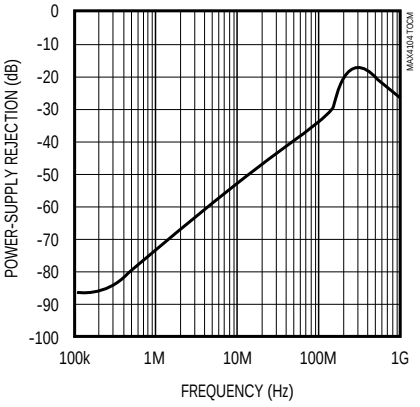
MAX4105
LARGE-SIGNAL GAIN
vs. FREQUENCY (A_{VCL} = +5)



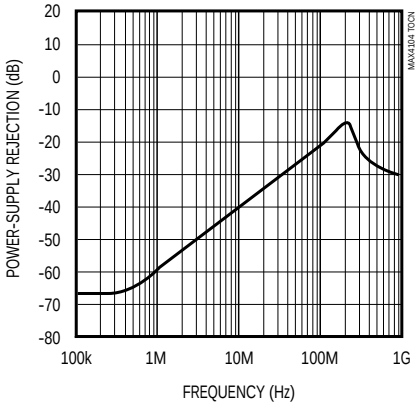
MAX4305
LARGE-SIGNAL GAIN
vs. FREQUENCY (A_{VCL} = +10)



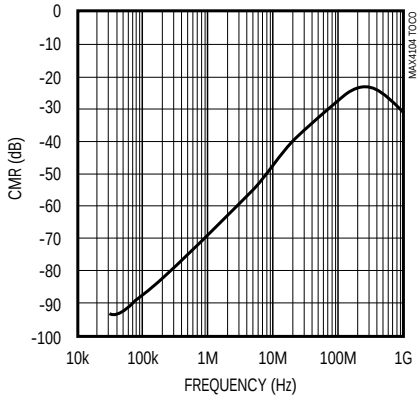
POSITIVE POWER-SUPPLY REJECTION
vs. FREQUENCY



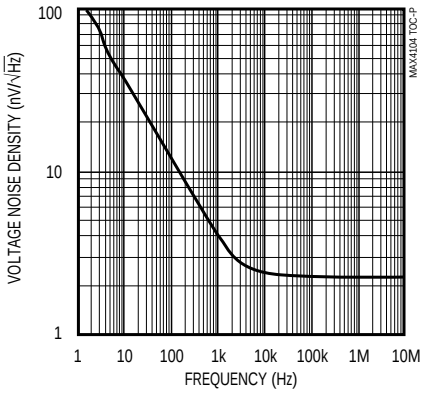
NEGATIVE POWER-SUPPLY REJECTION
vs. FREQUENCY



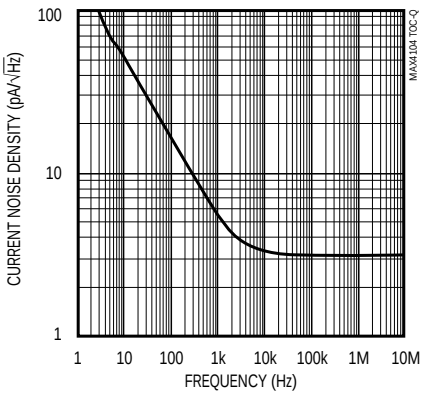
COMMON-MODE REJECTION
vs. FREQUENCY



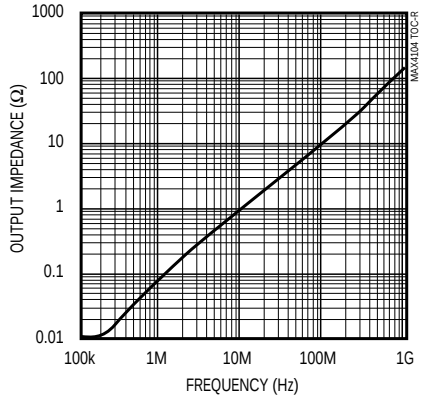
VOLTAGE NOISE DENSITY vs. FREQUENCY
(INPUT REFERRED)



CURRENT NOISE DENSITY vs. FREQUENCY
(INPUT REFERRED)



CLOSED-LOOP OUTPUT IMPEDANCE
vs. FREQUENCY

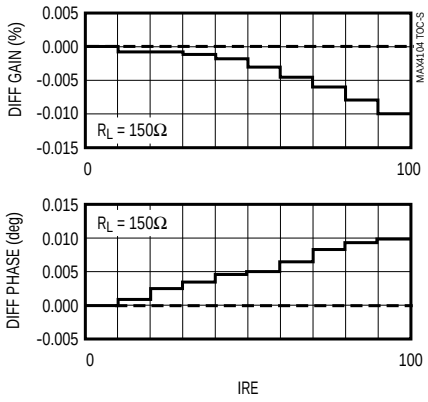


740MHz, Low-Noise, Low-Distortion Op Amps in SOT23-5

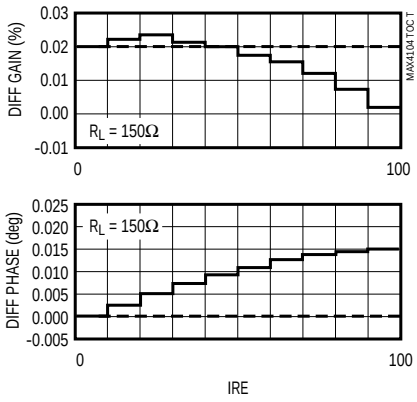
Typical Operating Characteristics (continued)

(V_{CC} = +5V, V_{EE} = -5V, R_F = 330Ω, R_L = 100Ω, T_A = +25°C, unless otherwise noted.)

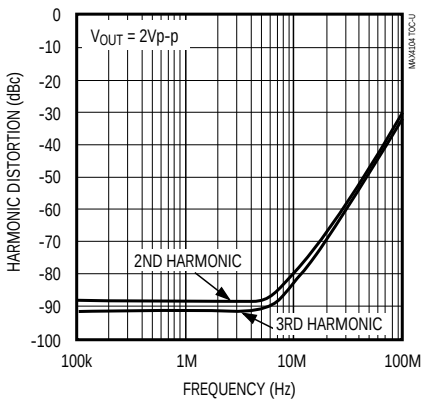
MAX4104/MAX4304
DIFFERENTIAL GAIN AND PHASE



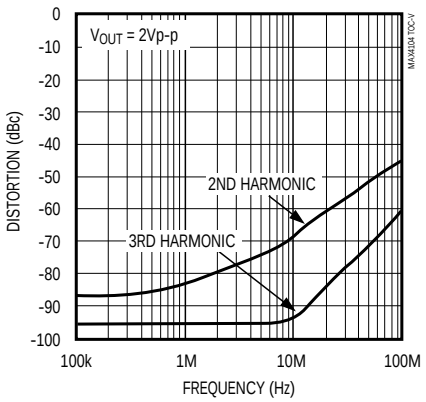
MAX4105/MAX4305
DIFFERENTIAL GAIN AND PHASE



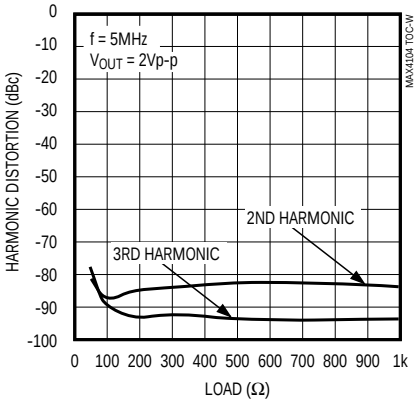
MAX4104/MAX4304
HARMONIC DISTORTION vs. FREQUENCY



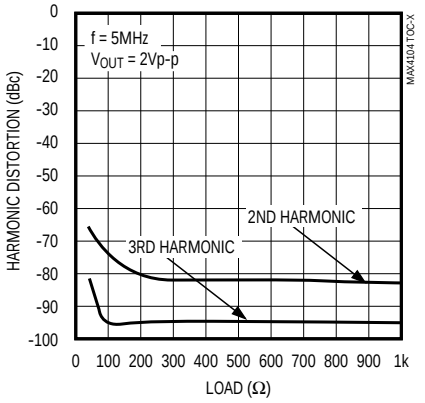
MAX4105/MAX4305
HARMONIC DISTORTION vs. FREQUENCY



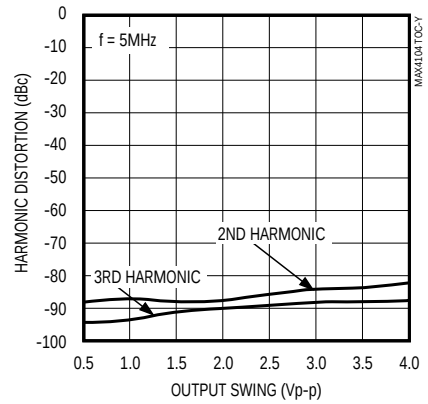
MAX4104/MAX4304
HARMONIC DISTORTION vs. LOAD



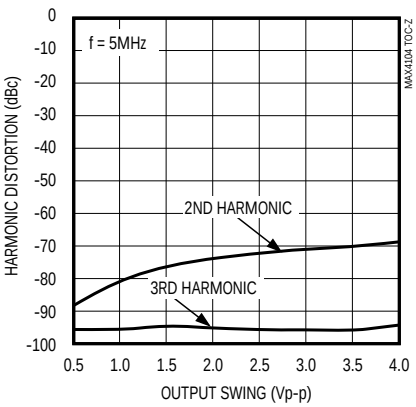
MAX4105/MAX4305
HARMONIC DISTORTION vs. LOAD



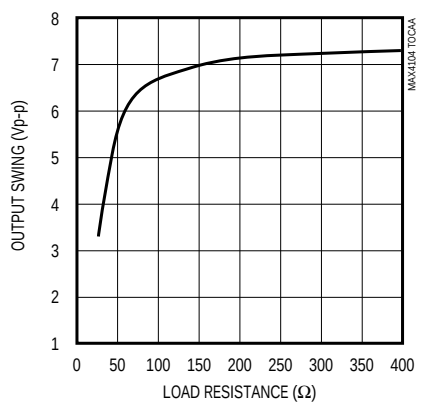
MAX4104/MAX4304
HARMONIC DISTORTION
vs. OUTPUT SWING



MAX4105/MAX4305
HARMONIC DISTORTION
vs. OUTPUT SWING



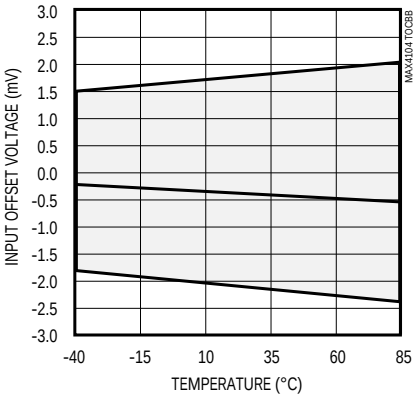
OUTPUT SWING vs. LOAD RESISTANCE



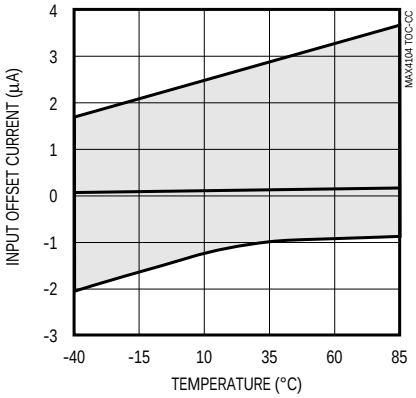
Typical Operating Characteristics (continued)

(V_{CC} = +5V, V_{EE} = -5V, R_F = 330Ω, R_L = 100Ω, T_A = +25°C, unless otherwise noted.)

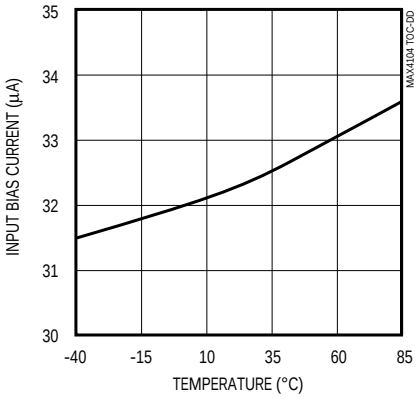
INPUT OFFSET VOLTAGE vs. TEMPERATURE



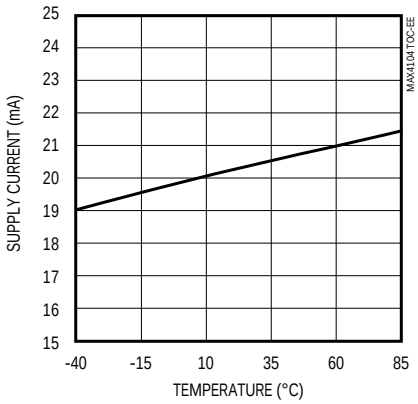
INPUT OFFSET CURRENT vs. TEMPERATURE



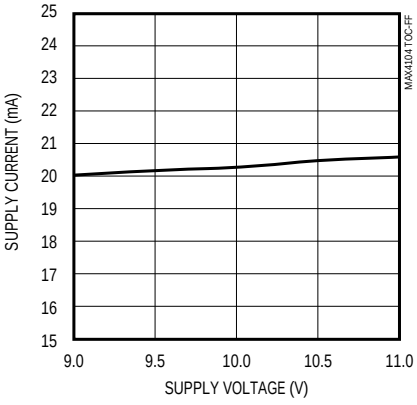
INPUT BIAS CURRENT vs. TEMPERATURE



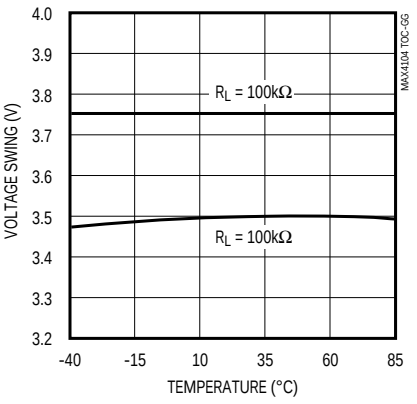
SUPPLY CURRENT vs. TEMPERATURE



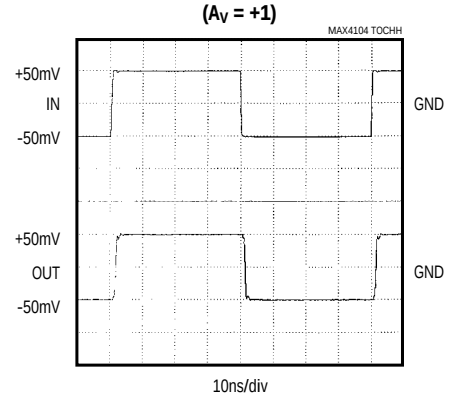
SUPPLY CURRENT vs. SUPPLY VOLTAGE



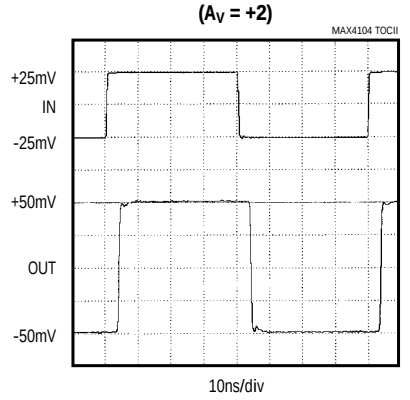
POSITIVE OUTPUT VOLTAGE SWING vs. TEMPERATURE



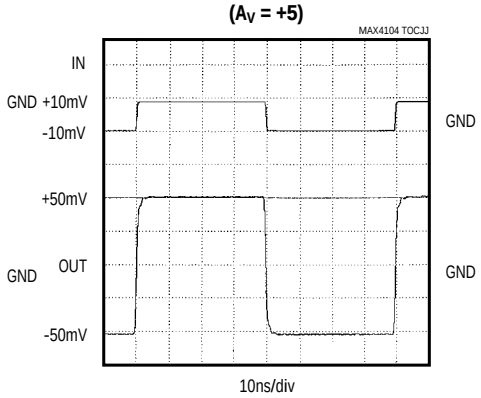
MAX4104 SMALL-SIGNAL PULSE RESPONSE (A_V = +1)



MAX4304 SMALL-SIGNAL PULSE RESPONSE (A_V = +2)



MAX4105 SMALL-SIGNAL PULSE RESPONSE (A_V = +5)

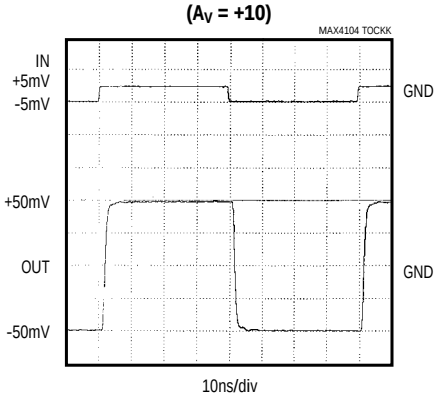


740MHz, Low-Noise, Low-Distortion Op Amps in SOT23-5

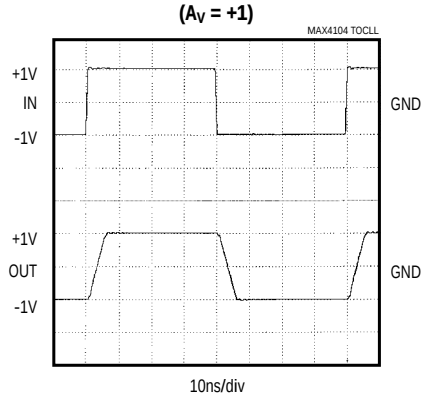
Typical Operating Characteristics (continued)

(V_{CC} = +5V, V_{EE} = -5V, R_F = 330Ω, R_L = 100Ω, T_A = +25°C, unless otherwise noted.)

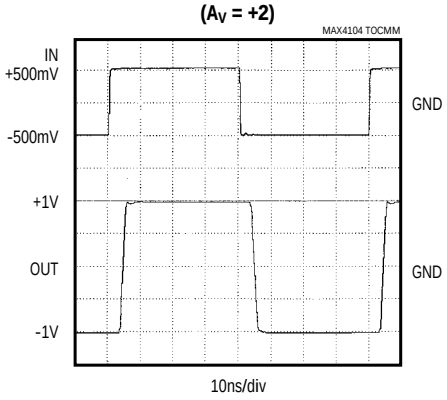
MAX4305
SMALL-SIGNAL PULSE RESPONSE
(A_V = +10)



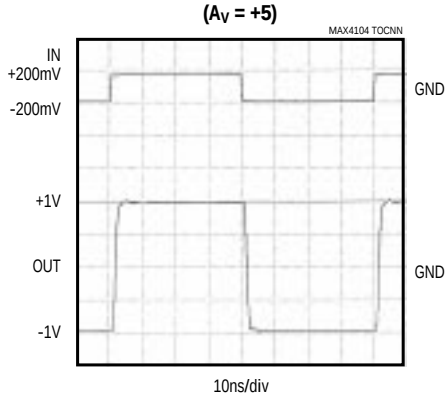
MAX4104
LARGE-SIGNAL PULSE RESPONSE
(A_V = +1)



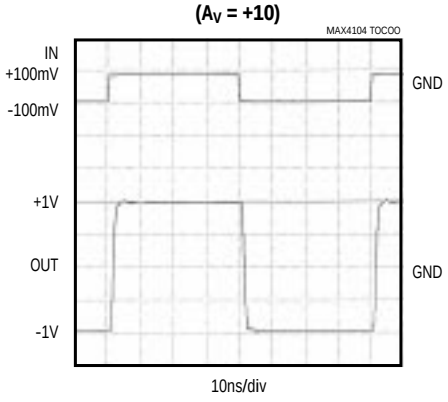
MAX4305
LARGE-SIGNAL PULSE RESPONSE
(A_V = +2)



MAX4105
LARGE-SIGNAL PULSE RESPONSE
(A_V = +5)



MAX4305
LARGE-SIGNAL PULSE RESPONSE
(A_V = +10)



Pin Description

PIN		NAME	FUNCTION
SOT23-5	SO		
—	1, 5, 8	N.C.	Not internally connected.
4	2	IN-	Amplifier Inverting Input
3	3	IN+	Amplifier Noninverting Input
2	4	V_{EE}	Negative Power Supply
1	6	OUT	Amplifier Output
5	7	V_{CC}	Positive Power Supply

Detailed Description

The MAX4104/MAX4304/MAX4105/MAX4305 are ultra-high-speed, low-noise amplifiers featuring -3dB bandwidths up to 880MHz, 0.1dB gain flatness up to 100MHz, and low differential gain and phase errors of 0.01% and 0.01°, respectively. These devices operate on dual power supplies ranging from ±3.5V to ±5.5V and require only 20mA of supply current.

The MAX4104/MAX4304/MAX4105/MAX4305 are optimized for minimum closed-loop gains of +1V/V, +2V/V, +5V/V and +10V/V (respectively) with corresponding -3dB bandwidths of 880MHz, 730MHz, 430MHz, and 350MHz. Each device in this family features a low input voltage noise density of only 2.1nV/√Hz (at 1MHz), an output current drive of ±70mA, and spurious-free dynamic range as low as -88dBc (5MHz, R_L = 100Ω).

Applications Information

Layout and Power-Supply Bypassing

The MAX4104/MAX4105/MAX4304/MAX4305 have an extremely high bandwidth, and consequently require careful board layout, including the possible use of constant-impedance microstrip or stripline techniques.

To realize the full AC performance of these high-speed amplifiers, pay careful attention to power-supply bypassing and board layout. The PC board should have at least two layers: a signal and power layer on one side, and a large, low-impedance ground plane on the other side. The ground plane should be as free of voids as possible. With multilayer boards, locate the ground plane on a layer that incorporates no signal or power traces.

Regardless of whether or not a constant-impedance board is used, it is best to observe the following guidelines when designing the board:

1) Do not use wire-wrapped boards (they are much too inductive) or breadboards (they are much too capacitive).
2) Do not use IC sockets. IC sockets increase reactances.
3) Keep signal lines as short and straight as possible. Do not make 90° turns; round all corners.
4) Observe high-frequency bypassing techniques to maintain the amplifier's accuracy and stability.
5) Bear in mind that, in general, surface-mount components have shorter bodies and lower parasitic reactance, resulting in greatly improved high-frequency performance over through-hole components.

The bypass capacitors should include 1nF and 0.1µF ceramic surface-mount capacitors between each supply pin and the ground plane, located as close to the package as possible. Optionally, place a 10µF tantalum capacitor at the power supply pins' point of entry to the PC board to ensure the integrity of incoming supplies. The power-supply trace should lead directly from the tantalum capacitor to the V_{CC} and V_{EE} pins. To minimize parasitic inductance, keep PC traces short and use surface-mount components.

Input termination resistors and output back-termination resistors, if used, should be surface-mount types, and should be placed as close to the IC pins as possible.

DC and Noise Errors

The MAX4104/MAX4105/MAX4304/MAX4305 output offset voltage, V_{OUT} (Figure 1), can be calculated with the following equation:

$$V_{OUT} = [V_{OS} + (I_{B+} \times R_S) + (I_{B-} \times (R_F \parallel R_G))] [1 + R_F / R_G]$$

where:

V_{OS} = input offset voltage (in volts)
$1 + R_F/R_G$ = amplifier closed-loop gain (dimensionless)
I_{B+} = noninverting input bias current (in amps)
I_{B-} = inverting input bias current (in amps)
R_G = gain-setting resistor (in ohms)
R_F = feedback resistor (in ohms)
R_S = source resistor at noninverting input (in ohms)

The following equation represents output noise density:

$$e_{n(OUT)} = \left[1+\frac{R_F}{R_G}\right] \sqrt{\left(i_n \times R_S\right)^2 + \left[i_n \times \left(R_F \parallel R_G\right)\right]^2 + e_n^2}$$

740MHz, Low-Noise, Low-Distortion Op Amps in SOT23-5

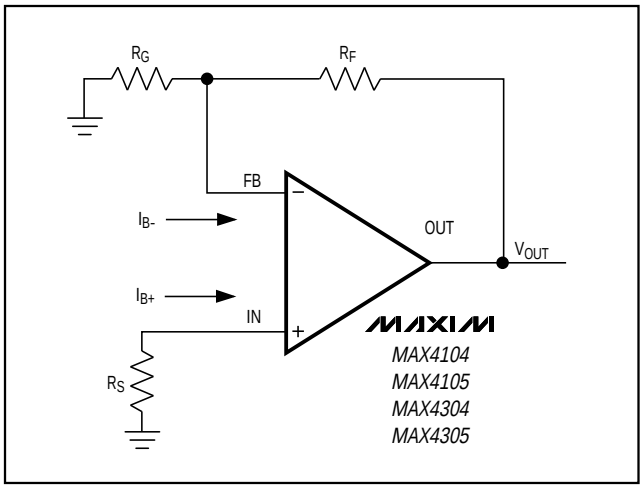


Figure 1. Output Offset Voltage

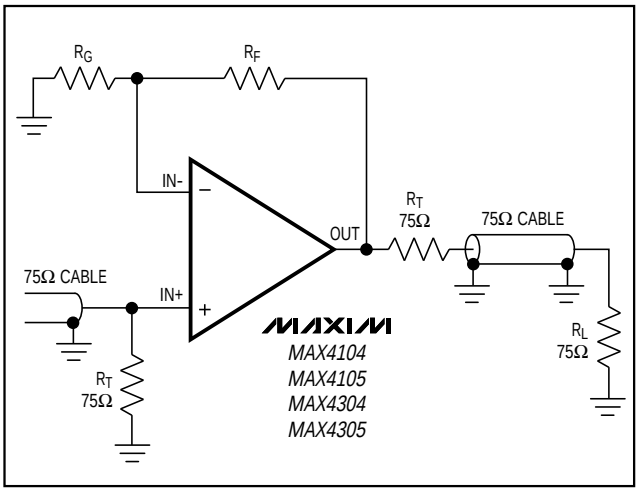


Figure 2. Video Line Driver

where:

i_n = input current noise density (in pA/√Hz)
e_n = input voltage noise density (in nV/√Hz)

The MAX4104/MAX4105/MAX4304/MAX4305 have a very low, 2.1nV/√Hz input voltage noise density and 3.1pA/√Hz input current noise density.

An example of DC-error calculations, using the MAX4304 typical data and the typical operating circuit with $R_F = R_G = 330\Omega$ ($R_F \parallel R_G = 165\Omega$) and $R_S = 50\Omega$ gives:

$$V_{OUT} = \left[\left(32 \times 10^{-6}\right)(50) + \left(32 \times 10^{-6}\right)(165\Omega) + 1 \times 10^{-3}\right][1 + 1]$$

$$V_{OUT} = 15.8mV$$

Calculating total output noise in a similar manner yields the following:

$$e_{n(OUT)} = [1+1]\sqrt{\left(3.1 \times 10^{-12} \times 50\right)^2 + \left(3.1 \times 10^{-12} \times 165\right)^2 + \left(2.1 \times 10^{-9}\right)^2}$$

$$e_{n(OUT)} = 4.3nV\sqrt{Hz}$$

With a 200MHz system bandwidth, this calculates to 60.8µV_{RMS} (approximately 365µVp-p, using the six-sigma calculation).

ADC Input Buffers

Input buffer amplifiers can be a source of significant error in high-speed ADC applications. The input buffer is usually required to rapidly charge and discharge the ADC's input, which is often capacitive. In addition, the input impedance of a high-speed ADC often changes very rapidly during the conversion cycle—a condition that demands an amplifier with very low output impedance at high frequencies to maintain measurement accuracy. The combination of high-speed, fast slew rate, low noise, and low-distortion available in the MAX4104/MAX4105/MAX4304/MAX4305 makes them ideally suited for use as buffer amplifiers in high-speed ADC applications.

Video Line Driver

The MAX4104/MAX4105/MAX4304/MAX4305 are optimized to drive coaxial transmission lines when the cable is terminated at both ends, as shown in Figure 2. To minimize reflections and maximize power transfer, select the termination resistors to match the characteristic impedance of the transmission line. Cable frequency response can cause variations in the flatness of the signal.

Driving Capacitive Loads

The MAX4104/MAX4105/MAX4304/MAX4305 provide maximum AC performance when driving no output load capacitance. This is the case when driving a correctly terminated transmission line (i.e., a back-terminated cable).

In most amplifier circuits, driving a large load capacitance increases the chance of oscillations occurring. The amplifier's output impedance and the load capacitor combine to add a pole and excess phase to the loop response. If the pole's frequency is low enough and phase margin is degraded sufficiently, oscillations may result.

A second concern when driving capacitive loads originates from the amplifier's output impedance, which

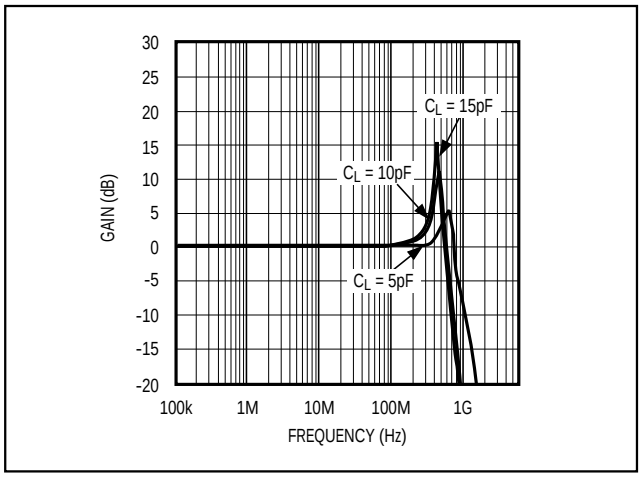


Figure 3a. MAX4104 Frequency Response with Capacitive Load and No Isolation Resistor

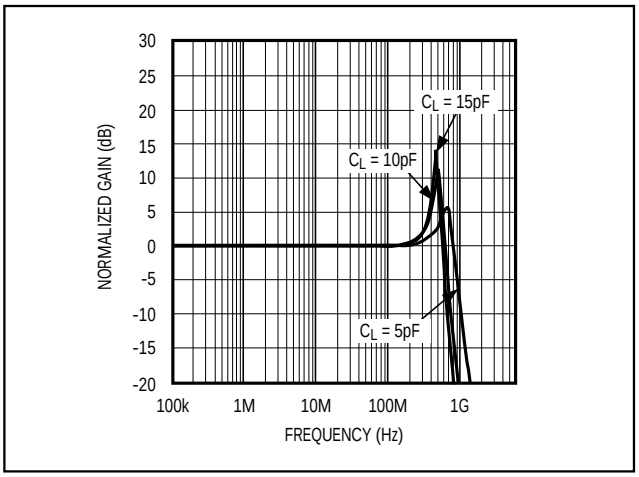


Figure 3b. MAX4304 Frequency Response with Capacitive Load and No Isolation Resistor

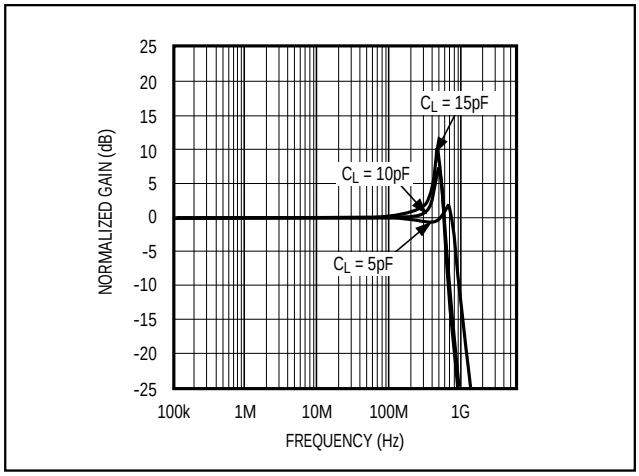


Figure 3c. MAX4105 Frequency Response with Capacitive Load and No Isolation Resistor

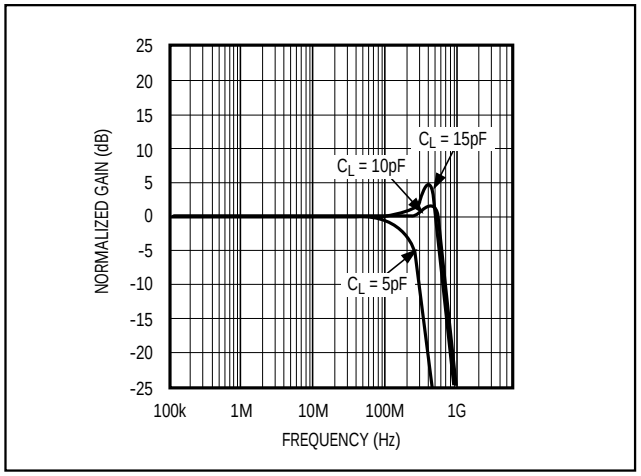


Figure 3d. MAX4305 Frequency Response with Capacitive Load and No Isolation Resistor

appears inductive at high frequencies. This inductance forms an L-C resonant circuit with the capacitive load, which causes peaking in the frequency response and degrades the amplifier's phase margin.

The MAX4104/MAX4105/MAX4304/MAX4305 drive capacitive loads up to 10pF without oscillation. However, some peaking may occur in the frequency domain (Figure 3). To drive larger capacitance loads or to reduce ringing, add an isolation resistor between the amplifier's output and the load (Figure 4).

The value of R_{ISO} depends on the circuit's gain and the capacitive load (Figure 5). Figure 6 shows the MAX4104/MAX4105/MAX4304/MAX4305 frequency response with the isolation resistor and a capacitive load. With higher capacitive values, bandwidth is dominated by the RC network formed by R_{ISO} and C_L; the bandwidth of the amplifier itself is much higher. Also note that the isolation resistor forms a divider that decreases the voltage delivered to the load.

Maxim's High-Speed Evaluation Boards

The MAX4104 evaluation kit manual shows a suggested layout for Maxim's high-speed, single-amplifier evaluation boards. This board was developed using the techniques described previously (*see Layout and Power-Supply Bypassing* section). The smallest available surface-mount resistors were used for the feedback and back-termination resistors to minimize the

740MHz, Low-Noise, Low-Distortion Op Amps in SOT23-5

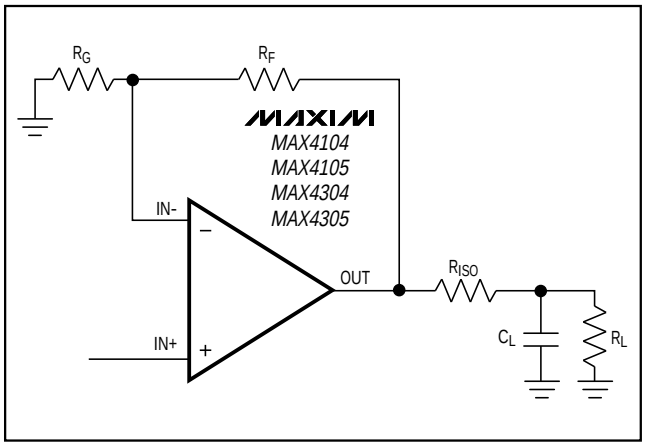


Figure 4. Using an Isolation Resistor (R_{ISO}) for High Capacitive Loads

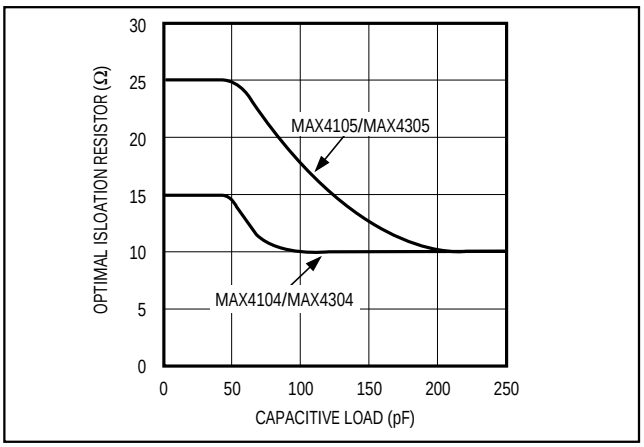
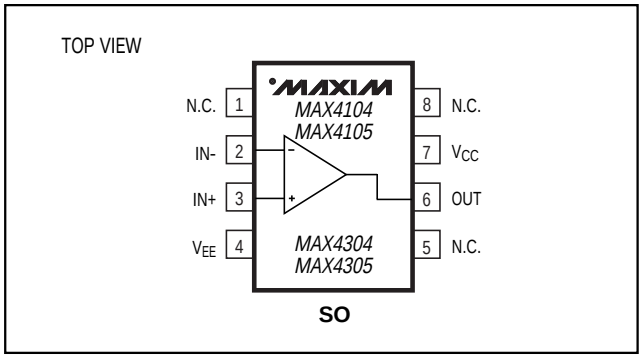


Figure 5. Optimal Isolation Resistor (R_{ISO}) vs. Capacitive Load

Pin Configurations (continued)



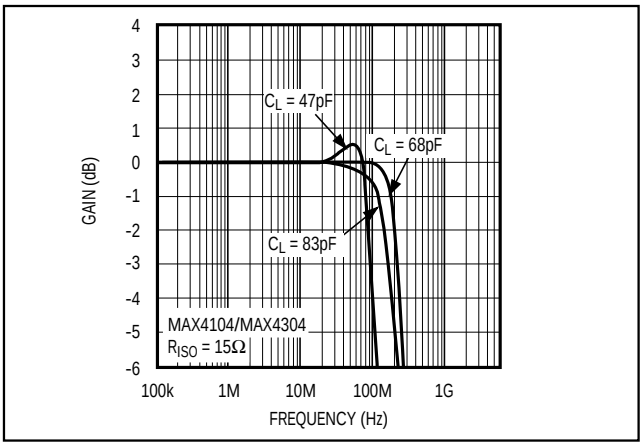


Figure 6. Frequency Responses vs. Capacitive Load with 15Ω Isolation Resistor

distance from the IC to these resistors, thus reducing the capacitance associated with longer lead lengths.

SMA connectors were used for best high-frequency performance. Because distances are extremely short, performance is unaffected by the fact that inputs and outputs do not match a 50Ω line. However, in applications that require lead lengths greater than 1/4 of the wavelength of the highest frequency of interest, constant-impedance traces should be used.

Fully assembled evaluation boards are available for the MAX4104 in an 8-pin SO package.

Ordering Information (continued)

PART	TEMP. RANGE	PIN-PACKAGE	SOT TOP MARK
MAX4105ESA	-40°C to +85°C	8 SO	—
MAX4105EUK-T	-40°C to +85°C	5 SOT23-5	ACCP
MAX4304ESA	-40°C to +85°C	8 SO	—
MAX4304EUK-T	-40°C to +85°C	5 SOT23-5	ACCQ
MAX4305ESA*	-40°C to +85°C	8 SO	—
MAX4305EUK-T	-40°C to +85°C	5 SOT23-5	ACCR

**Future product—contact factory for availability.*

Chip Information

TRANSISTOR COUNT: 44

SUBSTRATE CONNECTED TO V_{EE}

19-2234; Rev 1; 11/02

MAXIM

ECL/PECL Phase-Frequency Detectors

General Description

The MAX9382/MAX9383 are high-speed PECL/ECL phase-frequency detectors designed for use in high-bandwidth phase-locked loop (PLL) applications. The devices compare a single-ended reference (R) and a VCO (V) input and produce pulse streams on differential up (U) and down (D) outputs. When integrated, the difference of the output pulse streams provides a control voltage proportional to input phase or frequency difference. Guaranteed minimum short pulse duration completely eliminates minimum phase difference requirements during the lock condition, maximizing loop jitter performance.

The MAX9382/MAX9383 feature low propagation and reset delay, making them ideal for high-frequency clock synchronization use. The MAX9382 uses 100K logic levels, has a supply voltage range of V_{CC} - V_{EE} = 4.2V to 5.5V, and is pin compatible with Motorola's MCK12140. The MAX9383 uses 10H logic levels with a supply voltage range of V_{CC} - V_{EE} = 4.75V to 5.5V and is pin compatible with the MCH12140.

The MAX9382/MAX9383 are available in industry-standard 8-pin SO and space-saving 8-pin µMAX packages.

Applications

Precision Clock Distribution

Central Office

DSLAM

DLC

Base Station

ATE

Features

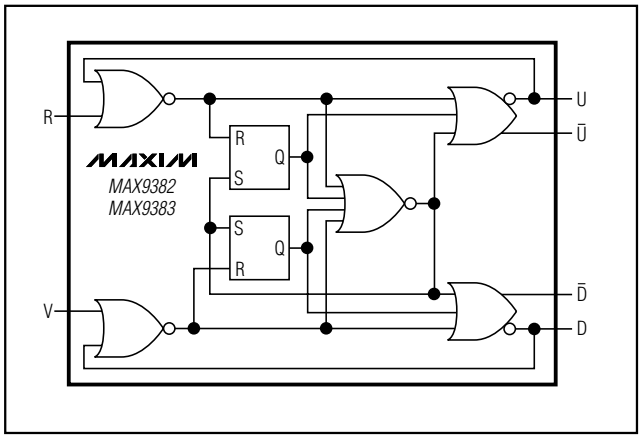
- **Guaranteed Minimum Pulse Width Eliminates Dead Band**
- **450MHz Typical Bandwidth with up to ±π Phase Detection**
- **75kΩ Internal Input Pulldown Resistors**
- **44mA Typical Supply Current**
- **±2kV ESD Protection (Human Body Model)**
- **Pin Compatible with MCK12140 and MCH12140**
- **Available in 8-Pin µMAX and SO Packages**

Ordering Information

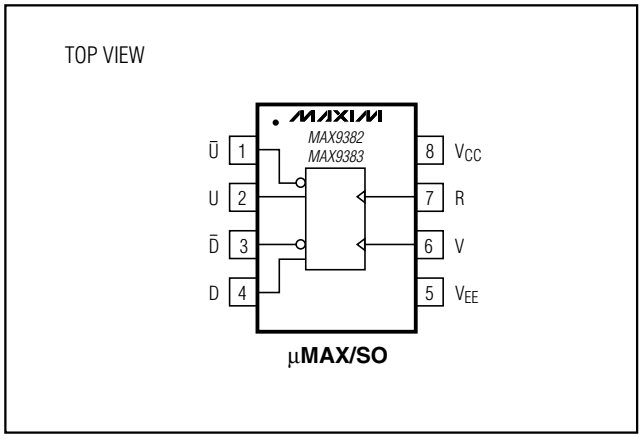
PART	TEMP RANGE	PIN-PACKAGE
MAX9382EUA*	-40°C to +85°C	8 µMAX
MAX9382ESA	-40°C to +85°C	8 SO
MAX9383EUA*	-40°C to +85°C	8 µMAX
MAX9383ESA	-40°C to +85°C	8 SO

**Future product—contact factory for availability.*

Functional Diagram



Pin Configuration



ECL/PECL Phase-Frequency Detectors

ABSOLUTE MAXIMUM RATINGS

V_{CC} - V_{EE} .....+6.0V
Inputs (R, V).....(V_{CC}) to (V_{EE} - 0.3V)
Continuous Output Current.....50mA
Surge Output Current.....100mA
Junction-to-Ambient Thermal Resistance in Still Air*
8-Pin µMAX.....+221°C/W
8-Pin SO.....+170°C/W
Junction-to-Ambient Thermal Resistance with*
500LFPM Airflow
8-Pin µMAX.....+155°C/W
8-Pin SO.....+99°C/W
Junction-to-Case Thermal Resistance
8-Pin µMAX.....+39°C/W
8-Pin SO.....+40°C/W
Operating Temperature Range.....-40°C to +85°C
Junction Temperature.....+150°C
Storage Temperature Range.....-65°C to +150°C
ESD Protection
Human Body Model (R, V, U, $\overline{U}$, D, $\overline{D}$).....±2kV
Soldering Temperature (10s)..... +300°C

*Ratings are for single-layer board.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

MAX9382 DC ELECTRICAL CHARACTERISTICS

(V_{CC} - V_{EE} = 4.2V to 5.5V. Outputs loaded with 50Ω ±1% to V_{CC} - 2V, unless otherwise noted. Typical values at V_{CC} - V_{EE} = 4.5V.) (Notes 1, 2, 3)

PARAMETER	SYMBOL	CONDITIONS	-40°C			+25°C			+85°C			UNITS
			MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
INPUTS (R, V)												
Input High Voltage	V_{IH}		V_{CC} - 1.165		V_{CC} - 0.880	V_{CC} - 1.165		V_{CC} - 0.880	V_{CC} - 1.165		V_{CC} - 0.880	V
Input Low Voltage	V_{IL}		V_{CC} - 1.810		V_{CC} - 1.475	V_{CC} - 1.810		V_{CC} - 1. 475	V_{CC} - 1.810		V_{CC} - 1.475	V
Input High Current	I_{IH}	$V_{IN} = V_{IHMAX}$			150			150			150	µA
Input Low Current	I_{IL}	$V_{IN} = V_{ILMIN}$	0.5			0.5			0.5			µA
OUTPUTS (U, $\overline{U}$, D, $\overline{D}$)												
Single-Ended Output High Voltage	V_{OH}	$V_{IN} = V_{IH}$ or V_{IL}	V_{CC} - 1.085	V_{CC} - 0.990	V_{CC} - 0.880	V_{CC} - 1.035	V_{CC} - 0.960	V_{CC} - 0.880	V_{CC} - 1.035	V_{CC} - 0.940	V_{CC} - 0.880	V
Single-Ended Output Low Voltage	V_{OL}	$V_{IN} = V_{IH}$ or V_{IL}	V_{CC} - 1.890	V_{CC} - 1.810	V_{CC} - 1.555	V_{CC} - 1.850	V_{CC} - 1.770	V_{CC} - 1.620	V_{CC} - 1.810	V_{CC} - 1.730	V_{CC} - 1.600	V
Differential Output Voltage	V_{OH} - V_{OL}	$V_{IN} = V_{IH}$ or V_{IL}	585	820		585	810		585	800		mV
POWER SUPPLY												
Supply Current	I_{EE}	(Note 4)		43	56		44	56		45	58	mA

MAX9383 DC ELECTRICAL CHARACTERISTICS

(V_{CC} - V_{EE} = 4.75V to 5.5V. Outputs loaded with 50Ω ±1% to V_{CC} - 2V, unless otherwise noted. Typical values at V_{CC} - V_{EE} = 5.2V.) (Notes 1, 2, 3)

PARAMETER	SYMBOL	CONDITIONS	-40°C			+25°C			+85°C			UNITS
			MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
INPUTS (R, V)												
Input High Voltage	V_{IH}		V_{CC} - 1.230		V_{CC} - 0.890	V_{CC} - 1.130		V_{CC} - 0.810	V_{CC} - 1.060		V_{CC} - 0.720	V
Input Low Voltage	V_{IL}		V_{CC} - 1.950		V_{CC} - 1.500	V_{CC} - 1.950		V_{CC} - 1. 480	V_{CC} - 1.950		V_{CC} - 1.480	V
Input High Current	I_{IH}	V_{IN} = V_{IHMAX}			150			150			150	µA
Input Low Current	I_{IL}	V_{IN} = V_{ILMIN}	0.5			0.5			0.5			µA
OUTPUTS (U, $\overline{U}$, D, $\overline{D}$)												
Single-Ended Output High Voltage	V_{OH}	V_{IN} = V_{IH} or V_{IL}	V_{CC} - 1.115	V_{CC} - 1.010	V_{CC} - 0.890	V_{CC} - 0.980	V_{CC} - 0.924	V_{CC} - 0.810	V_{CC} - 0.945	V_{CC} - 0.900	V_{CC} - 0.720	V
Single-Ended Output Low Voltage	V_{OL}	V_{IN} = V_{IH} or V_{IL}	V_{CC} - 1.990	V_{CC} - 1.832	V_{CC} - 1.650	V_{CC} - 1.950	V_{CC} - 1.740	V_{CC} - 1.630	V_{CC} - 1.950	V_{CC} - 1.700	V_{CC} - 1.595	V
Differential Output Voltage	V_{OH} - V_{OL}	V_{IN} = V_{IH} or V_{IL}	650	822		650	817		650	803		mV
POWER SUPPLY												
Supply Current	I_{EE}	(Note 4)		37	52		38	52		39	52	mA

MAX9382/MAX9383 AC ELECTRICAL CHARACTERISTICS

(Over specified DC input parameters, f = 100MHz, outputs loaded with 50Ω ±1% to V_{CC} - 2V, unless otherwise noted.) (Note 5)

PARAMETER	SYMBOL	CONDITIONS	-40°C			+25°C			+85°C			UNITS
			MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
R Input to U Output Delay	t_{PRU}	Figure 1	575	650	750	590	660	780	635	720	830	ps
V Input to D Output Delay	t_{PVD}	Figure 1	575	650	750	590	660	780	635	720	830	ps
R Input to D Output Delay	t_{PRD}	Figure 1	945	1120	1320	960	1110	1360	1005	1150	1360	ps
V Input to U Output Delay	t_{PVU}	Figure 1	945	1120	1320	960	1110	1360	1005	1150	1360	ps
Minimum Pulse Duration	t_{Pmin}	Figure 1	370	470		370	450		370	430		ps

MAX9382/MAX9383 AC ELECTRICAL CHARACTERISTICS (continued)

(Over specified DC input parameters, f = 100MHz, outputs loaded with 50Ω ±1% to V_{CC} - 2V, unless otherwise noted.) (Note 5)

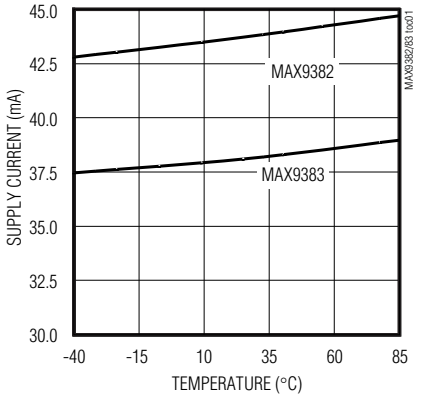
PARAMETER	SYMBOL	CONDITIONS	-40°C			+25°C			+85°C			UNITS
			MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
Maximum Operating Frequency	f_{MAX}	±π usable phase difference range	400	450		400	450		400	450		MHz
Phase Offset		V_{IN} = 200MHz, 50% duty cycle (Note 6)		30	70		28	60		28	60	ps
Added Random Jitter	t_{RJ}	V_{IN} = 400MHz, 50% duty cycle (Note 7)		0.2	1.0		0.2	1.0		0.2	1.0	ps (RMS)
Output Rise/ Fall Time	t_R, t_F	20% to 80%, Figure 2	80		160	100		180	110		190	ps

Note 1: Measurements are made with the device in thermal equilibrium.
Note 2: Current into a pin is defined as positive. Current out of a pin is defined as negative.
Note 3: DC parameters are production tested at +85°C. DC limits are guaranteed by design and characterization over the full operating temperature range.
Note 4: All pins open except V_{CC} and V_{EE}.
Note 5: Guaranteed by design and characterization. Limits are set to ±6 sigma.
Note 6: Phase offset is defined as the difference in propagation delay timing between the two input paths. It is measured between the U and D outputs at the differential crosspoint with a rising edge simultaneously applied at the R and V inputs.
Note 7: Device jitter added to the input signal.

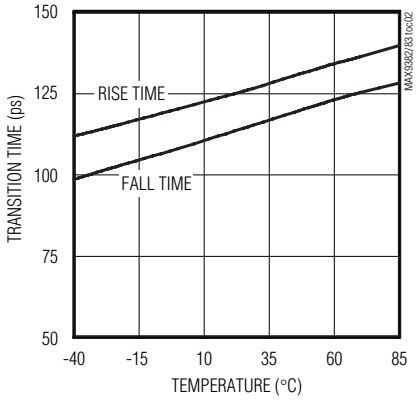
Typical Operating Characteristics

(V_{CC} - V_{EE} = +4.5V (MAX9382) or V_{CC} - V_{EE} = +5.2V (MAX9383), V_{IH} = V_{CC} - 1.00V, V_{IL} = V_{CC} - 1.60V, f_R = f_V = 100MHz, outputs loaded with 50Ω to V_{CC} - 2V, T_A = +25°C, unless otherwise noted.)

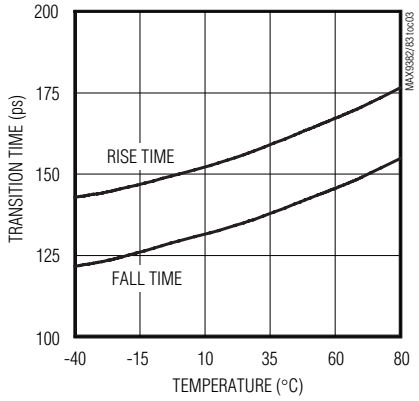
SUPPLY CURRENT vs. TEMPERATURE



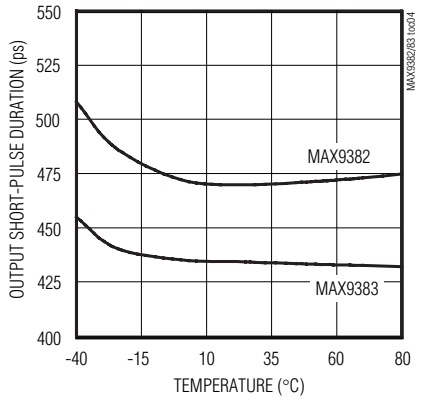
MAX9382 TRANSITION TIME vs. TEMPERATURE



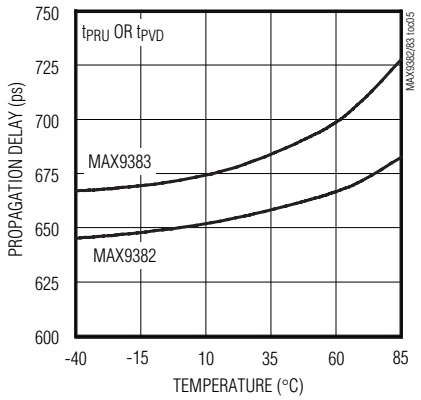
MAX9383 TRANSITION TIME vs. TEMPERATURE



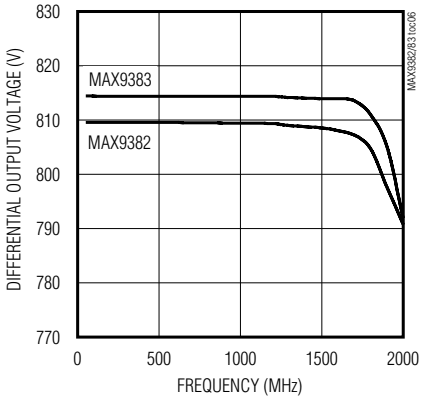
OUTPUT SHORT-PULSE DURATION vs. TEMPERATURE



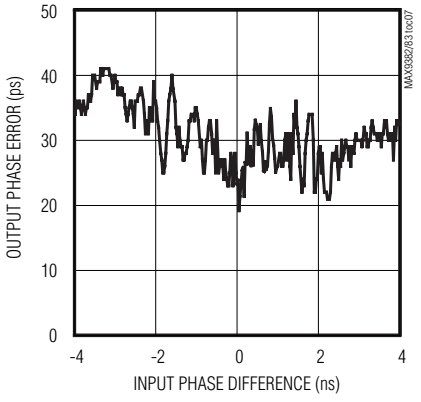
PROPAGATION DELAY vs. TEMPERATURE



DIFFERENTIAL OUTPUT VOLTAGE vs. FREQUENCY



OUTPUT PHASE ERROR vs. INPUT PHASE DIFFERENCE



ECL/PECL Phase-Frequency Detectors

Pin Description

PIN	NAME	FUNCTION
1	$\overline{U}$	Inverting Up Output. Pulse stream is generated at this pin when $f_R > f_V$ or V lags R in phase. Terminate with 50Ω resistor to V_{CC} - 2V or equivalent.
2	U	Noninverting Up Output. Pulse stream is generated at this pin when $f_R > f_V$ or V lags R in phase. Terminate with 50Ω resistor to V_{CC} - 2V or equivalent.
3	$\overline{D}$	Inverting Down Output. Pulse stream is generated at this pin when $f_V > f_R$ or R lags V in phase. Terminate with 50Ω resistor to V_{CC} - 2V or equivalent.
4	D	Noninverting Down Output. Pulse stream is generated at this pin when $f_V > f_R$ or R lags V in phase. Terminate with 50Ω resistor to V_{CC} - 2V or equivalent.
5	V_{EE}	Negative Supply
6	V	Single-Ended VCO Input
7	R	Single-Ended Reference Input
8	V_{CC}	Positive Supply. Bypass from V_{CC} to V_{EE} with 0.1µF and 0.01µF ceramic capacitors. Place the capacitors as close to the device as possible with the smaller value capacitor closest to the device.

Detailed Description

The MAX9382/MAX9383 are high-speed phase or frequency detectors. The MAX9382 is compatible with 100K logic and has a power-supply range of V_{CC} - V_{EE} = 4.2V to 5.5V. The MAX9383 is compatible with 10H logic with a power-supply range of V_{CC} - V_{EE} = 4.75V to 5.5V. Both devices are specified to function from -40°C to +85°C.

Each device is symmetrical; the R and V input functions may be swapped, together with the U and D output functions, and the inputs and outputs relabeled. Because of this device symmetry, a necessary condition for correct phase measurement operation is that the $\overline{U}$ and $\overline{D}$ outputs must both be high (state 0 condition) when the rising edge of the leading input is received. This condition is automatically generated when the two inputs are at different frequencies.

Phase Detection

The MAX9382/MAX9383 are intended for use in high-bandwidth PLL applications. These devices compare a single-ended VCO input (V) to a single-ended reference input (R) to determine the phase or frequency relationship between V and R. The device differential outputs U, $\overline{U}$ and D, $\overline{D}$ provide pulse trains with duty cycle proportional to the phase or frequency difference between R and V. These outputs are the up and down signals required to control the system VCO. Figure 1 shows typical waveforms when V leads R and V lags R. Subtracting and integrating these two outputs provide the necessary VCO control signal. Figure 3 shows the device transfer function obtained. The detector can detect phase differences up to ±2π. The application frequency and the characteristics of the device internal reset circuits determine the usable input phase difference range.

Frequency Detection

Figure 4 is the state diagram for the MAX9382/MAX9383. With the two inputs at the same frequency, and input R leading input V, the device toggles between states 0 and 2. Similarly, if input R lags input V, the device toggles between states 0 and 1. With the two inputs at different frequencies, the output becomes a function of the frequency difference. The normalized ideal transfer function is given by:

$$V_{OUT\_AVE} = 1 - \frac{f_R}{2f_V} \text{ for } f_V > f_R$$

and

$$V_{OUT\_AVE} = 1 - \frac{f_V}{2f_R} \text{ for } f_R > f_V$$

Output Pulses

When inputs R and V are at the same phase and frequency, outputs U, $\overline{U}$ and D, $\overline{D}$ produce a stream of minimum duration pulses that occur at the rising edges of the input waveforms. This is the lock condition. If either input starts to lead the other in phase, the width of pulses on the corresponding output (U for R input, D for V input) increases in proportion to the phase difference. In a PLL implementation, these outputs direct the

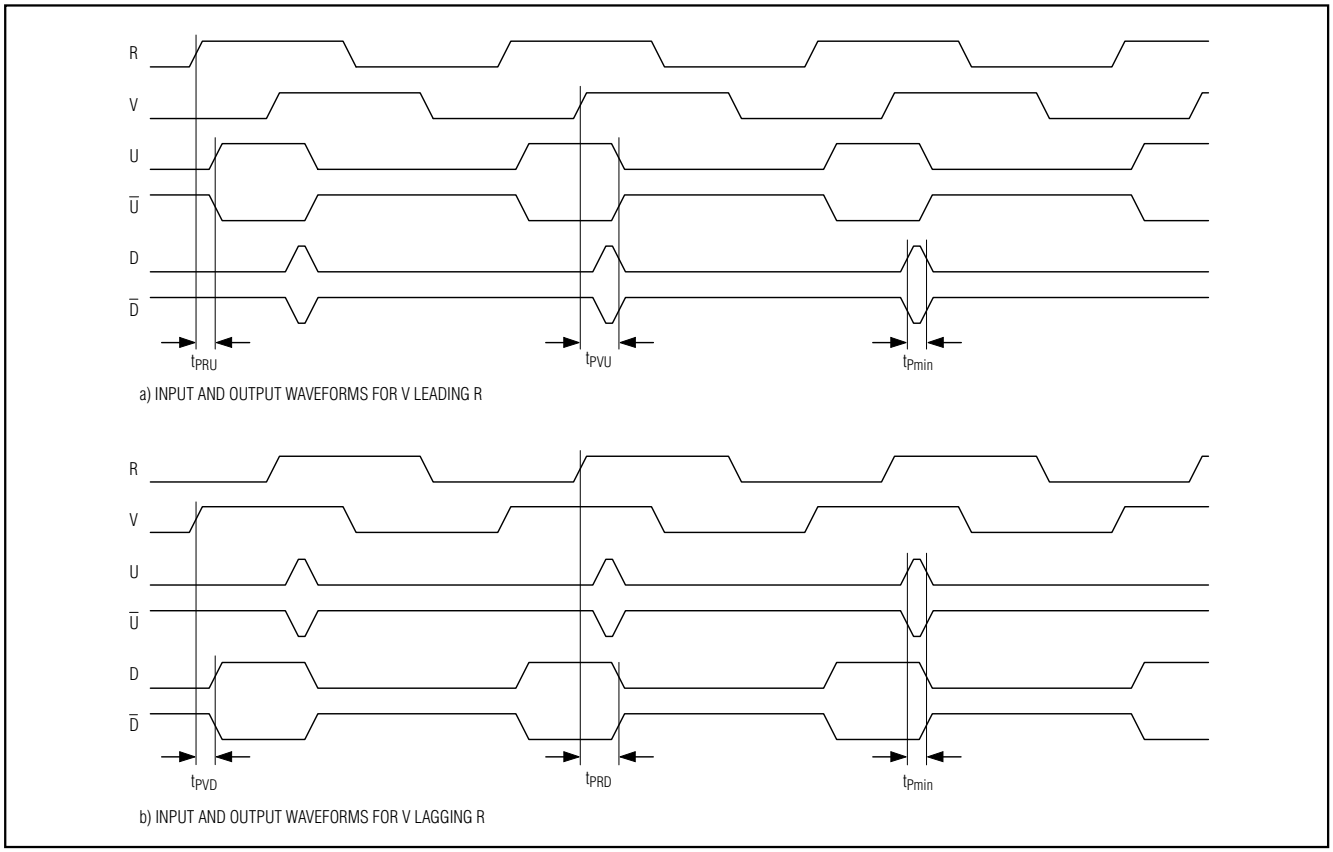


Figure 1. Typical Waveforms when $f_R = f_V$

system VCO to increase or decrease frequency to maintain the lock condition.

The minimum output pulse duration is an important parameter for the design of the signal processing functions, which follow the phase detector. When controlling a charge-pump integrator, a detector can produce a dead-zone characteristic at the lock condition if the minimum pulse width is too short. MAX9382/MAX9383 eliminate this dead-zone characteristic, and the resulting phase offset at lock, by providing a well-defined minimum output pulse width.

Applications Information

The MAX9382/MAX9383 input and output levels are defined to be relative to the positive supply voltage. In ECL systems, the positive supply voltage is conventionally chosen to be system ground. This arrangement produces the best noise immunity, since ground is normally a system-wide reference voltage. Operate the devices with V_{CC} connected to ground and V_{EE} connected to a negative supply for ECL systems. With

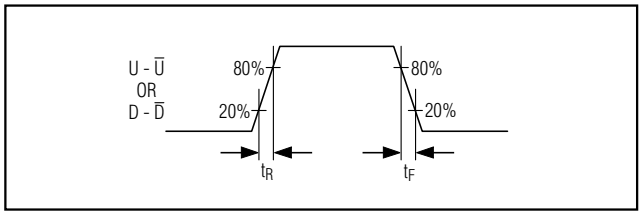


Figure 2. Output Transition Times

PECL systems, connect V_{CC} to a positive supply and V_{EE} to ground.

Power-Supply Bypassing

Adequate power-supply bypassing is necessary to maximize the performance and noise immunity of ECL devices. This is particularly true of a PECL system where the power-supply voltage is used as a reference. Bypass V_{CC} to V_{EE} with high-frequency surface-mount ceramic 0.1µF and 0.01µF capacitors in parallel and as close to the device as possible, with the 0.01µF capaci-

ECL/PECL Phase-Frequency Detectors

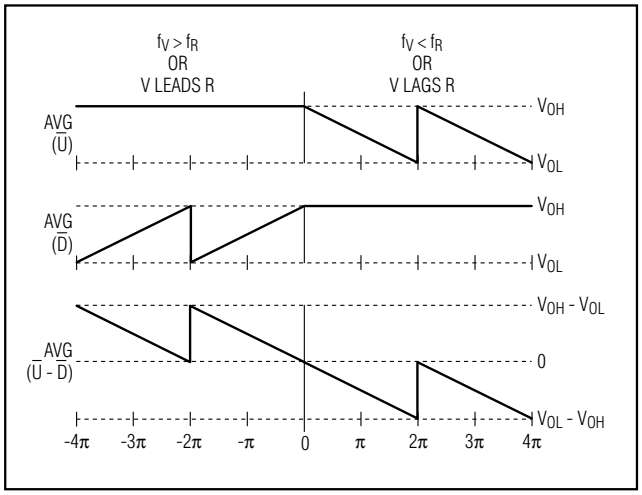


Figure 3. Average Output Voltage vs. Phase Difference

tor closest to the device pins. Use multiple parallel vias for ground plane connection to minimize inductance.

Circuit Board Traces

Input and output trace characteristics affect the performance of ECL/PECL devices. Connect each of the detector's inputs and outputs to a 50Ω characteristic impedance trace. Avoid impedance discontinuities, maintain the distance between differential traces, avoid sharp corners, and keep the electrical length of the differential traces matched. This maximizes common-mode noise rejection and reduces signal skew. Trace vias cause impedance discontinuities, so keep the number of vias in the 50Ω traces to a minimum. Reduce reflections by maintaining the 50Ω characteristic impedance through connectors and across cables.

Output Termination

Terminate outputs through 50Ω to V_{CC} - 2V or use an equivalent Thevenin termination. When a single-ended signal is taken from a differential output, terminate both outputs. For example, if the U output of the MAX9382 or MAX9383 is connected to a single-ended input, terminate both the U and $\overline{U}$ outputs.

Chip Information

TRANSISTOR COUNT: 706

PROCESS: Bipolar

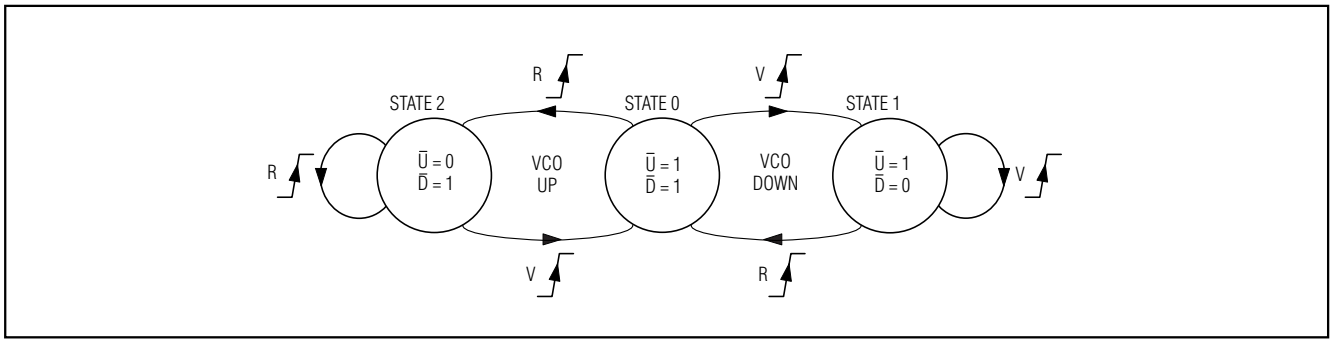
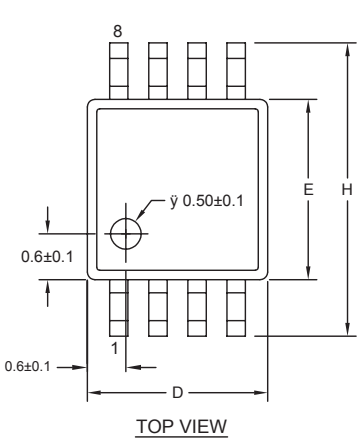


Figure 4. MAX9382/MAX9383 State Diagram

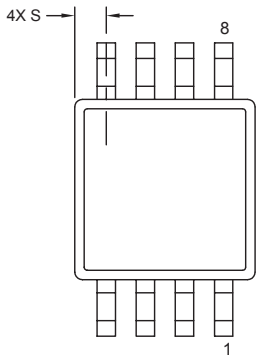
Package Information

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information, go to **www.maxim-ic.com/packages**.)

8LUMAXD.EPS

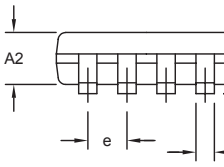


TOP VIEW

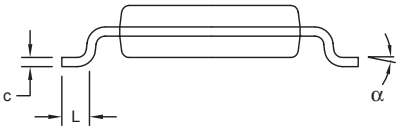


BOTTOM VIEW

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	-	0.043	-	1.10
A1	0.002	0.006	0.05	0.15
A2	0.030	0.037	0.75	0.95
b	0.010	0.014	0.25	0.36
c	0.005	0.007	0.13	0.18
D	0.116	0.120	2.95	3.05
e	0.0256 BSC		0.65 BSC	
E	0.116	0.120	2.95	3.05
H	0.188	0.198	4.78	5.03
L	0.016	0.026	0.41	0.66
α	0∞	6∞	0∞	6∞
S	0.0207 BSC		0.5250 BSC	



FRONT VIEW



SIDE VIEW

NOTES:
1. D&E DO NOT INCLUDE MOLD FLASH.
2. MOLD FLASH OR PROTRUSIONS NOT TO EXCEED 0.15MM (.006").
3. CONTROLLING DIMENSION: MILLIMETERS.
4. MEETS JEDEC MO-187C-AA.

DALLAS SEMICONDUCTOR MAXIM

PROPRIETARY INFORMATION

TITLE: PACKAGE OUTLINE, 8L uMAX/uSOP

APPROVAL	DOCUMENT CONTROL NO.	REV.	
	21-0036	J	1/1

ECL/PECL Phase-Frequency Detectors

Package Information (continued)

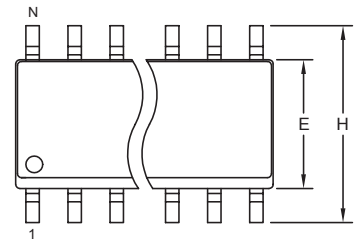
(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information, go to **www.maxim-ic.com/packages**.)

SOICN .EPS

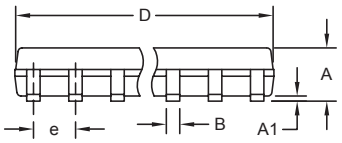
	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.053	0.069	1.35	1.75
A1	0.004	0.010	0.10	0.25
B	0.014	0.019	0.35	0.49
C	0.007	0.010	0.19	0.25
e	0.050 BSC		1.27 BSC	
E	0.150	0.157	3.80	4.00
H	0.228	0.244	5.80	6.20
L	0.016	0.050	0.40	1.27

VARIATIONS:

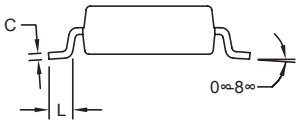
	INCHES		MILLIMETERS			
DIM	MIN	MAX	MIN	MAX	N	MS012
D	0.189	0.197	4.80	5.00	8	AA
D	0.337	0.344	8.55	8.75	14	AB
D	0.386	0.394	9.80	10.00	16	AC



TOP VIEW



FRONT VIEW



SIDE VIEW

NOTES:
1. D&E DO NOT INCLUDE MOLD FLASH.
2. MOLD FLASH OR PROTRUSIONS NOT TO EXCEED 0.15mm (.006").
3. LEADS TO BE COPLANAR WITHIN 0.10mm (.004").
4. CONTROLLING DIMENSION: MILLIMETERS.
5. MEETS JEDEC MS012.
6. N = NUMBER OF PINS.

DALLAS SEMICONDUCTOR MAXIM

PROPRIETARY INFORMATION

TITLE: PACKAGE OUTLINE, .150" SOIC

APPROVAL	DOCUMENT CONTROL NO.	REV.	
	21-0041	B	1/1

Maxim cannot assume responsibility for use of any circuitry other than circuitry entirely embodied in a Maxim product. No circuit patent licenses are implied. Maxim reserves the right to change the circuitry and specifications without notice at any time.