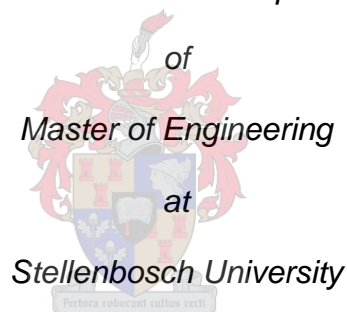


The Design and Development of a Co-pulse IFM Receiver

by

Kobus Myburgh

Thesis presented in fulfilment of the requirements for the degree



Supervisor: Prof. J.B. de Swardt

Department Electrical and Electronic Engineering

December 2015

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature:

K. Myburgh

Date:

ABSTRACT

The instantaneous frequency measurement (IFM) receiver is normally a subunit used in electronic warfare receivers. Its purpose is to measure the frequency of pulse and continuous wave signals over a very wide bandwidth and power dynamic range. It should be noted that the band of interest can range from 500 MHz to 40 GHz. This is an extremely wide bandwidth. These receivers have a very high resolution (typically 1 MHz) over this very wide bandwidth and the error of these receivers is typically a few megahertz. The IFM receiver should reveal the presence of signals. It is not generally known in advance whether signals are present, even if they are, nor their frequency and other characteristics.

An inherent disadvantage of all traditional IFM receivers is that overlapping pulses within the same power range corrupt the discriminator outputs to the extent that frequency measurements appear random. This is particularly bothersome, as high duty cycle and continuous wave emitters reduce the frequency measurements to little value.

This thesis entails the design and development of an IFM receiver (co-pulse IFM receiver) that can distinguish overlapping pulses within the same power range.

Different acquisition receivers were researched to make sure that the IFM receiver is still the best option, taking bandwidth, frequency resolution/accuracy and throughput time (latency) into consideration. The traditional IFM receiver was researched and simulated in MATLAB.

While researching the traditional IFM receiver, literature was discovered that identified Prony's method as a solution to distinguish overlapping pulses (simultaneous signals). Literature proposing Prony's method that had been submitted from as early as 1989 was found. However, no evidence of the use of the method in current commercial IFM receivers was found, indicating that the implementation of Prony's method is the difficult part. Prony's method was simulated in MATLAB, then implemented, simulated and tested in hardware.

The results of the hardware implementation were documented and demonstrated a solution to distinguish overlapping pulses.

OPSOMMING

Die oomblikfrekwensiemeting (OFM)-ontvanger is gewoonlik 'n sub-eenheid wat gebruik word in elektroniese oorlog-ontvangers. Die doel daarvan is om die frekwensie van gepulsde en kontinue seine te meet oor 'n baie wye bandwydte en wye drywing dinamiese bereik. Daar moet in ag geneem word dat die bandwydte van belang vanaf 500 MHz tot 40 GHz kan strek. Dit is 'n uiters wye bandwydte. Hierdie ontvangers het 'n baie hoë resolusie (tipies 1 MHz) oor die baie wye bandwydte en die meetfout van hierdie ontvangers is tipies 'n paar megahertz. Die OFM-ontvanger moet die teenwoordigheid van seine uitwys. Dit is gewoonlik nie eers seker of daar wel seine is nie, nog minder wat die moontlike frekwensie en ander eienskappe is.

'n Nadeel van alle tradisionele OFM-ontvangers is dat oorvleuelende pulse wat naby dieselfde drywings bereik is, die diskrimineerders se uitsette verkeerd beïnvloed, sodat die frekwensiemetings verkeerd is. Dit is 'n groot probleem want hoë diensiklus- en kontinue sein-uitsenders verlaag die frekwensiemeting se betroubaarheid.

Die doel van hierdie tesis is om 'n OFM-ontvanger (tweepuls-OFM-ontvanger) te ontwikkel wat oorvleuelende pulse wat naby dieselfde drywingsbereik is, te kan onderskei.

Verskillende sein-optelontvangers is nagevors om seker te maak dat die OFM-ontvanger nog steeds die beste opsie is as bandwydte, frekwensieresolusie, akkuraatheid en vertragingstyd (latensie) in ag geneem word. Die tradisionele OFM-ontvanger is nagevors en is in MATLAB gesimuleer.

Terwyl die tradisionele OFM-ontvanger nagevors is, is literatuur gevind wat Prony se metode voorstel as oplossing vir die probleem van oorvleuelende pulse (gelyktydige seine). Literatuur wat so vroeg as 1989 gepubliseer is, is gevind wat Prony se metode voorstel. Bewys van die gebruik daarvan in huidige kommersiële OFM-ontvangers is egter nie gekry nie, wat aandui dat die implementering daarvan baie moeilik is. Prony se metode is gesimuleer in MATLAB. Daarna is dit geïmplementeer, gesimuleer en in hardeware getoets.

Die resultate van die hardeware-implementering is gedokumenteer en dit demonstreer 'n oplossing om oorvleuelende pulse te onderskei.

ACKNOWLEDGEMENTS

My sincerest gratitude to:

- Prof. Johann de Swardt (Stellenbosch University) for his support and guidance as study leader.
- Dr Derik Minnaar (CSIR, DPSS Unit, REW Competence Area) for his support and guidance as study mentor. I appreciate all the encouragement you gave me over the years.
- Dr André Niemann (CSIR, DPSS Unit, REW Competence Area) for his insight and interest. Thank you for always being willing to help.
- Mr Leon Staphorst (CSIR, Meraka Unit) for giving me the opportunity to enrol for my master's degree.
- Mr Marcel Gouws (CSIR, DPSS Unit, REW Competence Area) for his patience, insight and sharing his knowledge of Xilinx System Generator and Digital IFM receivers.
- Mr René Broich (CSIR, DPSS Unit, REW Competence Area) for sharing his knowledge of Xilinx System Generator.
- Mr Llewellyn Potgieter (Saab Grintek) for sharing his knowledge of MATLAB and the remote control of synthesisers.
- The CSIR for financial assistance during my studies.
- Thank you to my family for the encouragement throughout the project.

CONTENTS

| | |
|--|-------------|
| Abstract | iii |
| Opsomming | iv |
| Acknowledgements | v |
| List of Figures | ix |
| List of Tables | xii |
| Nomenclature | xiii |
| 1 Introduction | 1 |
| 1.1 Background | 1 |
| 1.2 Problem Statement | 2 |
| 1.3 Thesis Layout | 2 |
| 2 Literature | 3 |
| 2.1 Different Signal Acquisition Receivers | 3 |
| 2.1.1 Crystal Video Receiver | 3 |
| 2.1.2 Scanning Superheterodyne Receivers | 3 |
| 2.1.3 Compressive or Microscan Receivers | 3 |
| 2.1.4 IFM Receivers | 4 |
| 2.1.5 Channelised Receivers | 4 |
| 2.1.6 Acousto-optic Receivers | 5 |
| 2.1.7 Digital Receivers – Fast Fourier Transform on a Digital Signal Processing Platform | 5 |
| 2.1.8 Summary of Different Receivers | 5 |
| 2.2 Basic IFM Receiver Principles | 6 |
| 2.3 Basic IFM Receiver Components | 7 |
| 2.3.1 Limiting Amplifiers | 7 |
| 2.3.2 Frequency Discriminator | 7 |
| 2.3.3 Polar Display | 12 |
| 2.4 Digital IFM Receiver | 13 |
| 2.5 Simultaneous Signals | 14 |
| 3 Simulation of a Typical Digital IFM (DIFM) Receiver | 16 |
| 3.1 Single Delay Line | 16 |
| 3.1.1 Single Input Signal with a Frequency of 200 MHz | 17 |
| 3.1.2 Single Input Signal with a Frequency of 1000 MHz | 17 |
| 3.1.3 Two Input Signals with Frequencies of 200 MHz and 1000 MHz | 18 |
| 3.2 Multiple Delay Lines | 19 |
| 3.2.1 Single Input Signal with a Frequency of 200 MHz | 20 |
| 3.2.2 Single Input Signal with a Frequency of 1000 MHz | 21 |
| 3.2.3 Two Input Signals with Frequencies of 200 MHz and 1000 MHz | 21 |

| | | |
|-------------------|---|-----------|
| 3.3 | Typical Frequency Error Caused by Simultaneous Signals | 22 |
| 3.4 | Effect of Phase Modulation on the IFM | 25 |
| 4 | Investigation of a possible solution | 28 |
| 4.1 | Prony's Method | 28 |
| 4.1.1 | Discriminator Outputs and Simultaneous Signal Condition | 28 |
| 4.1.2 | Two Simultaneous Signals | 29 |
| 4.1.3 | Three Simultaneous Signals | 31 |
| 4.1.4 | Four Simultaneous Signals | 32 |
| 4.1.5 | N Simultaneous Signals | 33 |
| 4.1.6 | Alternative choices of discriminator lags | 34 |
| 5 | Simulation of a possible solution | 36 |
| 5.1 | Prony's Method | 36 |
| 5.1.1 | Four Delay Lines/Two Simultaneous Signals | 36 |
| 5.1.2 | Six Delay Lines/Three Simultaneous Signals | 39 |
| 5.1.3 | Eight Delay Lines/Four Simultaneous Signals | 42 |
| 5.1.4 | Summary of Simulation Results | 45 |
| 6 | Hardware Implementation and Simulation | 46 |
| 6.1 | Background on the DRFM Platform | 46 |
| 6.2 | Implementation Methodology | 48 |
| 6.3 | High-level block diagram of the hardware implementation | 49 |
| 6.4 | Analogue-to-digital Convertor | 49 |
| 6.5 | Deserialiser | 50 |
| 6.6 | Hilbert Transform | 50 |
| 6.7 | Delays | 50 |
| 6.8 | Phase Compute/Frequency Discriminators | 51 |
| 6.9 | Frequency Resolving | 53 |
| 6.9.1 | Outputs from the Frequency Discriminators | 53 |
| 6.9.2 | Constants $a_1 .. a_N$ | 53 |
| 6.9.3 | Determination of the Roots | 54 |
| 6.9.4 | Determination of the Frequency Values | 56 |
| 6.10 | Hardware Simulation Results | 56 |
| 7 | Hardware Results achieved | 61 |
| 7.1 | Hardware Test Setup | 61 |
| 7.2 | Hardware Implementation Results | 64 |
| 8 | Conclusion and suggestions for future work | 67 |
| | References | 69 |
| Appendix A | Single Delay Line Simulation MATLAB Code | 71 |
| Appendix B | Multiple Delay Line Simulation MATLAB Code | 73 |

| | | |
|-------------------|---|-----------|
| Appendix C | Prony's Method Simulation MATLAB Code | 76 |
| C.1 | Two Simultaneous Signals | 76 |
| C.2 | Three Simultaneous Signals | 79 |
| C.3 | Four Simultaneous Signals | 82 |
| Appendix D | Co-pulse DIFM System Generator Top Level | 87 |
| Appendix E | Co-pulse IFM in Top Level of Firmware Implementation | 88 |
| Appendix F | Roots of a Cubic Equation | 89 |
| Appendix G | MATLAB Post-processing | 90 |

LIST OF FIGURES

| | | |
|------------|--|----|
| Figure 1: | Receiver revealing the presence of various signals | 1 |
| Figure 2: | Basic IFM receiver | 6 |
| Figure 3: | Basic IFM receiver components | 7 |
| Figure 4: | Delay Line Discriminator | 8 |
| Figure 5: | Basic frequency discriminator | 10 |
| Figure 6: | Standard FD component symbols | 11 |
| Figure 7: | Most frequently described FD configuration | 12 |
| Figure 8: | Digital IFM receiver | 13 |
| Figure 9: | Characteristic dip during simultaneous signals | 14 |
| Figure 10: | Measured 3 GHz CW source | 15 |
| Figure 11: | Measured 17 GHz CW source | 15 |
| Figure 12: | Measured 3 GHz and 17 GHz CW sources | 15 |
| Figure 13: | Single-line DIFM receiver | 16 |
| Figure 14: | Frequency Output = 200 MHz | 17 |
| Figure 15: | Frequency output = 1000 MHz | 18 |
| Figure 16: | Frequency output = 45 MHz | 18 |
| Figure 17: | Multiple delay line DIFM receiver | 19 |
| Figure 18: | Frequency output = 200 MHz | 20 |
| Figure 19: | Frequency output = 1000 MHz | 21 |
| Figure 20: | Angle resolving between input signals and delayed versions | 22 |
| Figure 21: | Measured frequency (f1 = 200 MHz, f2 Sweeping) | 23 |
| Figure 22: | Measured frequency (f1 = 200 MHz, f2 = 300 MHz) | 23 |
| Figure 23: | Measured frequency (f1 = 200 MHz, f2 = 800 MHz) | 24 |
| Figure 24: | Measured frequency vs power difference | 24 |
| Figure 25: | Phase measurement of a chirp signal | 25 |
| Figure 26: | Frequency measurement of a chirp signal | 26 |
| Figure 27: | Phase measurement during a phase jump | 26 |
| Figure 28: | Frequency result during a phase jump | 27 |
| Figure 29: | DIFM receiver using Prony's resolving method | 36 |

| | | |
|------------|--|----|
| Figure 30: | Frequency output = 200 MHz/1000 MHz | 37 |
| Figure 31: | Frequency output = 200 MHz/1000 MHz, SNR = 40 dB | 38 |
| Figure 32: | Frequency output = 500 MHz/sweeping | 38 |
| Figure 33: | Frequency output = 200 MHz/1000 MHz | 39 |
| Figure 34: | DIFM receiver using Prony's resolving method – six delay lines | 40 |
| Figure 35: | Frequency output = 200 MHz/600 MHz/1000 MHz | 41 |
| Figure 36: | Frequency output = 200 MHz/600 MHz/1000 MHz, SNR = 40 dB | 41 |
| Figure 37: | Frequency output = 400 MHz/700 MHz/Sweeping | 42 |
| Figure 38: | DIFM receiver using Prony's resolving method – eight delay lines | 43 |
| Figure 39: | Frequency output = 100 MHz/400 MHz/700MHz/1000 MHz | 44 |
| Figure 40: | Frequency output = 100 MHz/400 MHz/700MHz/1000 MHz, SNR = 40 dB | 44 |
| Figure 41: | Frequency output = 400 MHz/600 MHz/800MHz/sweeping | 45 |
| Figure 42: | Block diagram of fourth generation DRFM hardware architecture | 46 |
| Figure 43: | Fourth generation DRFM developed by CSIR | 47 |
| Figure 44: | Fifth generation DRFM developed by the CSIR | 48 |
| Figure 45: | Co-pulse IFM receiver hardware implementation | 49 |
| Figure 46: | Deserialiser | 50 |
| Figure 47: | One sample (τ) delay | 51 |
| Figure 48: | Phase compute/frequency discriminator | 51 |
| Figure 49: | Frequency discriminator implementation | 52 |
| Figure 50: | Hardware simulation of the co-pulse IFM receiver | 57 |
| Figure 51: | Hardware simulation of the co-pulse IFM receiver (frequency crossing) | 57 |
| Figure 52: | Hardware simulation of the co-pulse IFM receiver (frequency crossing with LPF) | 58 |
| Figure 53: | Hardware simulation of the co-pulse IFM receiver (frequency crossing with determinant check) | 58 |
| Figure 54: | Hardware simulation of the co-pulse IFM receiver (LPF, determinant check with traditional IFM to resolve frequency crossing) | 59 |
| Figure 55: | Hardware simulation of the co-pulse IFM receiver (30 dB power difference) | 59 |
| Figure 56: | Hardware simulation of the co-pulse IFM receiver (30 dB power difference with determinant check) | 60 |
| Figure 57: | Block diagram of the test setup | 61 |

| | |
|--|----|
| Figure 58: Hardware test setup | 62 |
| Figure 59: DRFM in the hardware test setup | 63 |
| Figure 60: Combined signal generators | 64 |
| Figure 61: Hardware result of the co-pulse IFM receiver | 65 |
| Figure 62: Hardware result of the co-pulse IFM receiver (frequency crossing) | 65 |
| Figure 63: Hardware result of the co-pulse IFM receiver (30 dB power difference) | 66 |

LIST OF TABLES

| | | |
|----------|--|----|
| Table 1: | Comparison of signal acquisition receivers (including IFM) | 6 |
| Table 2: | Fourth generation DRFM performance specifications | 47 |

NOMENCLATURE

Abbreviations

| | |
|--------|--|
| AC | Alternating Current |
| ADC | Analogue-to-digital Converter |
| CSIR | Council for Scientific and Industrial Research |
| CVR | Crystal Video Receiver |
| CW | Continuous Wave |
| DAC | Digital-to-analogue Converter |
| dB | Decibel |
| DC | Direct Current |
| DLD | Delay Line Discriminator |
| DPM | Data-processing Module |
| DRFM | Digital Radio Frequency Memory |
| DSP | Digital Signal Processing |
| FD | Frequency Discriminator |
| FFT | Fast Fourier Transform |
| FPGA | Field-programmable Gate Array |
| GHz | Gigahertz |
| GUI | Graphical User Interface |
| Hz | Hertz |
| IBW | Instantaneous Bandwidth |
| IF | Intermediate Frequency |
| IFM | Instantaneous Frequency Measurement |
| IP | Internet Protocol |
| LAN | Local Area Network |
| LPF | Low-pass Filter |
| MATLAB | Matrix Laboratory (Mathworks, Inc.) |
| MHz | Megahertz |
| PD | Phase Discriminator |
| POI | Probability of Intercept |

| | |
|-----|-------------------------------|
| PW | Pulse Width |
| RF | Radio Frequency |
| SSD | Simultaneous Signal Detection |
| SNR | Signal-to-noise Ratio |
| VCO | Voltage-controlled oscillator |
| VME | Versa Module European |
| YIG | Yttrium Iron Garnet |

Constants

$$c = 3 \times 10^8 \text{ m/s}$$

1 Introduction

1.1 Background

The instantaneous frequency measurement (IFM) receiver is normally a subunit used in electronic warfare receivers. Its purpose is to measure the frequency of pulse and continuous wave (CW) signals over a very wide bandwidth and power dynamic range.

Firstly it should be noted that the band of interest can range from 500 MHz to 40 GHz. This is an extremely large bandwidth and a single receiver generally covers only a portion of this frequency range. Techniques can be used to extend cover to the whole band. Nevertheless it must be realised that very large bandwidths are involved.

Secondly, these receivers have a very high resolution (typically 1 MHz) over this very large bandwidth and the error of these receivers is typically a few Megahertz.

Next, it must be realised that the receivers are meant to reveal the presence of signals. It is not generally known in advance whether signals are present, even if they are, nor their frequency or other characteristics. Detection of the most complex and fleeting signals should be aimed at, and then obviously the simple signals will also be detected. It should also be borne in mind that several signals may be present at any given time interval and may perhaps even be active simultaneously.

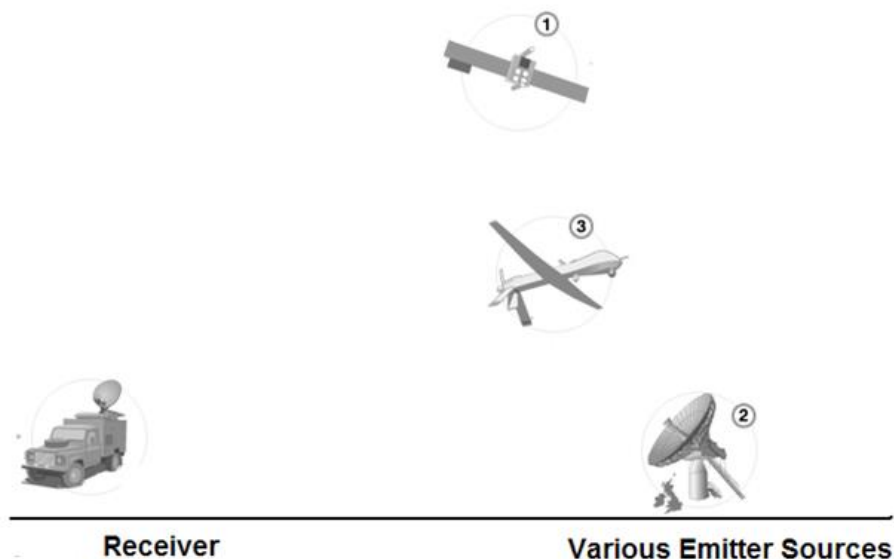


Figure 1: Receiver revealing the presence of various signals

A large proportion of the signals in this frequency band are emitted by radar units. The simplest radar signal is CW, but most are pulsed and have very short pulse durations. Typical receivers should be designed to intercept pulses as short as 50 ns in duration. Receivers should therefore aim at very high probability of

pulse intercept. Radars can also vary their frequency within each pulse and/or from pulse to pulse. They may also vary their pulse repetition frequency or may emit several beams at different frequencies.

Signals may also vary very widely in amplitude so that a wide dynamic range is desirable [1],[4],[19],[20],[21].

1.2 Problem Statement

An inherent disadvantage of all traditional IFM receivers is that overlapping pulses within the same power range corrupt the discriminator outputs to the extent that frequency measurements appear random. This is particularly bothersome, as high duty cycle and continuous wave emitters reduce the frequency measurements to little value under these conditions. Lately the trend has been to extend IFM receivers from very low frequency to well within the millimetre band, which demands reliable measurement over very wide bandwidths and very high emitter densities [1],[4],[5],[6],[8],[17],[18].

1.3 Thesis Layout

This thesis starts with an introduction, which gives an overview of IFM receivers and the project scope. In Chapter 2 various signal acquisition receivers are mentioned and compared, focussing mainly on the IFM receiver and explaining the inherent disadvantage of traditional IFM receivers. In Chapter 3 a typical digital IFM receiver is simulated in MATLAB and the results are discussed. In Chapter 4 a possible solution will be discussed to solve the inherent disadvantage of the IFM receiver. Chapter 5 will be about the simulation of the solution. Chapter 6 will explain how the solution was implemented on the digital radio frequency memory (DRFM) platform, which consist of a 2 Gbps analogue-to-digital converter (ADC) and a Xilinx Virtex5 field-programmable gate array (FPGA). Chapter 7 will document the results achieved, with the solution implemented on the DRFM platform. The final chapter is a conclusion and suggests some areas for future work.

2 Literature

2.1 Different Signal Acquisition Receivers

This section briefly describes a few different acquisition receivers and then compares them. Most of the information presented in this section is a summary of acquisition receivers explained in Polar Frequency Discriminators [4].

2.1.1 Crystal Video Receiver

The crystal video receiver (CVR) is the most widely used wideband receiver because of its simplicity, small size and low cost. In its simplest form it consists of a low noise radio frequency (RF) pre-amplifier, a square law video detector and a log video amplifier. This provides 100 percent probability of intercept (POI), a broad bandwidth and signal amplitude information. Its major drawback is that it provides no frequency information beyond the signal falling within the pass band of the receiver.

2.1.2 Scanning Superheterodyne Receivers

This receiver covers an octave band in steps (typically 5 MHz) but could equally well be swept. An yttrium iron garnet (YIG) preselector is used to obtain a wide dynamic range and image rejection with one frequency conversion. A 5 MHz intermediate frequency (IF) limits the minimum detectable pulse width to 200 ns. Increasing the IF bandwidth reduces the signal acquisition time and the minimum detectable pulse width but also reduces the sensitivity.

This receiver's dynamic range is its main advantage, typically 50 dB or greater. The accuracy of frequency information depends on oscillator stability and linearity and also the resolution of the IF bandwidth. A YIG oscillator typically gives 0.2 per cent accuracy or 5 to 8 MHz in the 2 to 4 GHz band.

The receiver exhibits excellent signal separation capability because of the frequency selectivity and a low false alarm rate. Its greatest disadvantage is its very low pulse POI. A given pulse will only be detected if the voltage-controlled oscillator (VCO) frequency is within an IF bandwidth (5 MHz for example) of the carrier frequency plus IF frequency while the pulse is present. If the VCO sweep period is much greater than the pulse length, the probability of interception may be very low.

2.1.3 Compressive or Microscan Receivers

The compressive receiver is a logical step up from the scanning superheterodyne receiver. It attempts to improve the scanning superheterodyne receiver's POI by using a very fast sweep time, ideally faster than the shortest pulse duration.

The input signal is mixed with a sweeping oscillator and this produces a "chirped" or frequency-modulated signal. This signal is applied to a pulse compression filter with a slope opposite to the swept oscillator. The output is a compressed signal whose position in time is a function of the frequency of the input signal.

This receiver provides a near unity POI while retaining the ability to separate simultaneous signals closely spaced in frequency.

In theory there is no limit to the bandwidth of these receivers. Practical limitations on bandwidth are imposed by local oscillator scan speed and linearity and also compressive filter delay linearity. Wide bandwidth, fast tuning and very narrow output pulses mean that very high data rates have to be processed and that processor costs are high. Another limitation is that the time-of-arrival measurements are quantised into intervals equal to the scan time.

2.1.4 IFM Receivers

This section serves only as a very brief introduction to IFM receivers to facilitate comparison with other receivers. IFM receivers will be covered in more detail later.

The IFM receiver achieves a near unity pulse POI by having a 'wide-open' front-end requiring no tuning. Octave bandwidths are typical and 2 – 18 GHz versions are easily available. Sensitivity and dynamic range are good, provided a limiting amplifier is used at the input to the frequency discriminator (FD). The FD's function is to accept an RF input and produce two video signals. If these two signals are applied to the vertical and horizontal plates of a polar display, the angular component can be scaled to read the frequency.

Since IFM receivers have no frequency dispersive components, they cannot separate two or more simultaneous signals. If two signals are present the display will show the vector sum of the two signals and thus will read neither signal correctly.

This inherent problem of IFM receivers is the focus of this thesis and will be discussed in depth in subsequent sections.

2.1.5 Channelised Receivers

Channelised receivers consist of banks of contiguous filters covering the frequency band of interest. Detectors following each filter show the presence of a signal within its associated filter. It offers the high POI of CVRs and IFM receivers and the signal separation ability of the superheterodyne receiver. Channelised receivers are often referred to as instantaneous Fourier transform receivers because of the way incoming signals are instantly broken up into frequency cells with a resolution of half the filter bandwidths.

If a 10 GHz bandwidth is to be covered and each filter is 10 MHz wide, 1000 filters are required. However, if a "folding" technique is used, the number of filters required can be greatly reduced, but every time folding takes place, noise is summed into the IF channel, greatly reducing the receiver sensitivity.

Another difficulty with channelised receivers is that the frequency accuracy and resolution required are often at odds with the filter bandwidth necessary to allow measurements of the narrowest pulses expected.

The large amounts of hardware required by channelised receivers have limited their use in the past but interest in them has been revived with the introduction of surface acoustic wave devices.

2.1.6 Acousto-optic Receivers

An alternative form of “instantaneous Fourier transform” or channelised receiver makes use of the interaction between monochromatic light and sound waves to synthesise the equivalent of a large bank of filters. The acousto-optic interaction takes place in a Bragg cell.

The Bragg cell is a combination of two transducers. The electro-mechanical transducer converts the RF signal into a mechanical vibration. The acousto-optic device uses the mechanical vibration to establish a periodic variation in density of materials. This acts as a diffracting grating, which refracts light passing through it. The amount of refraction is dependent on the wavelength of the mechanical vibration generated by the RF signal.

Acousto-optic receivers promise the bandwidth and accuracy of IFM receivers together with the ability to separate simultaneous signals. Their disadvantages include a relatively small dynamic range, complex processor interface hardware requirements for accurate time-of-arrival measurements and bulky packaging required for the optical components.

2.1.7 Digital Receivers – Fast Fourier Transform on a Digital Signal Processing Platform

The signal is transferred into time discrete amplitude samples with an ADC. A fast Fourier transform (FFT) algorithm on a digital signal processing (DSP) platform is then used to calculate the frequency of the signal. Other processes can also be used to determine the pulse repetition interval, pulse width, scan pattern and modulation type.

FFT works well for very narrow frequency bandwidths and is highly recommended for narrowband receivers. When bandwidths become too wide, FFT becomes very processing-intensive and is not instantaneous. History of data must be stored and continuous calculations must be done [1],[14].

Currently an ADC with a wide bandwidth (DC – 18 GHz, 3 bit) is available, but the processing on this wide bandwidth will be enormous.

2.1.8 Summary of Different Receivers

Several different types of acquisition receivers have been discussed in this section. No single receiver completely outshines the rest for every situation. Commonly then, a broadband receiver is used for initial signal detection and the steering of a narrowband receiver to the correct frequency for more complete signal analysis. Attempts have been made to rate each receiver qualitatively in Table 1. The figure of merit derived constitutes only a very rough guide. The table does indicate the prominence of IFM receivers. This also strengthens the argument for further research into IFM receivers and for their use as the broadband steering component of combination receivers. The FFT running on a DSP device is a good candidate for the narrowband receiver. The issue is the disadvantage that overlapping pulses, which corrupt the discriminator outputs to the extent that frequency measurements appear random in IFM receivers, can cause the IFM receiver to steer the narrowband receiver to the incorrect frequency.

Table 1: Comparison of signal acquisition receivers (including IFM)

| Qualitative Key Excellent = 1.00 Good = 0.67 Poor = 0.33 | Crystal-video | Superhet | Microscan | IFM | Channelised | Acousto-optical | FFT on DSP Platform |
|---|---------------|-----------|-----------|-----------|-------------|-----------------|---------------------|
| Figure of Merit | 0.78 | 0.61 | 0.73 | 0.89 | 0.61 | 0.89 | 0.83 |
| Instantaneous Bandwidth | Excellent | Poor | Good | Excellent | Good | Good | Good |
| Frequency Resolution | Poor | Excellent | Good | Excellent | Good | Excellent | Excellent |
| Throughput Time | Excellent | Poor | Excellent | Excellent | Good | Good | Poor |
| Simultaneous Signals | Poor | Poor | Good | Poor | Excellent | Excellent | Excellent |
| Size/Weight | Excellent | Excellent | Good | Excellent | Poor | Excellent | Excellent |
| Cost | Excellent | Good | Good | Excellent | Poor | Excellent | Excellent |

2.2 Basic IFM Receiver Principles

IFM receivers will be covered in more detail in the following sections.

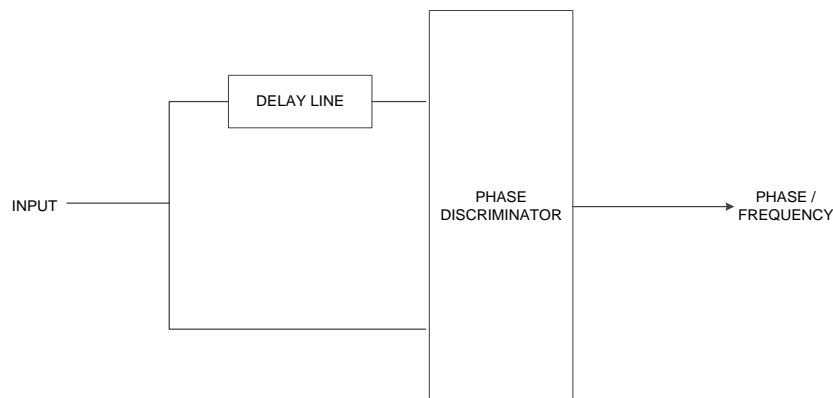


Figure 2: Basic IFM receiver

The basic principle of frequency measurement in an IFM receiver is comparing the direct and the delayed versions of a signal using a phase discriminator (PD). The phase difference is directly proportional to frequency. Assuming no phase modulation, this method approximates the true instantaneous frequency as the delay approaches zero, hence the name IFM [1],[2],[3],[4],[22].

2.3 Basic IFM Receiver Components

Figure 3 shows the components of a basic IFM receiver [4].

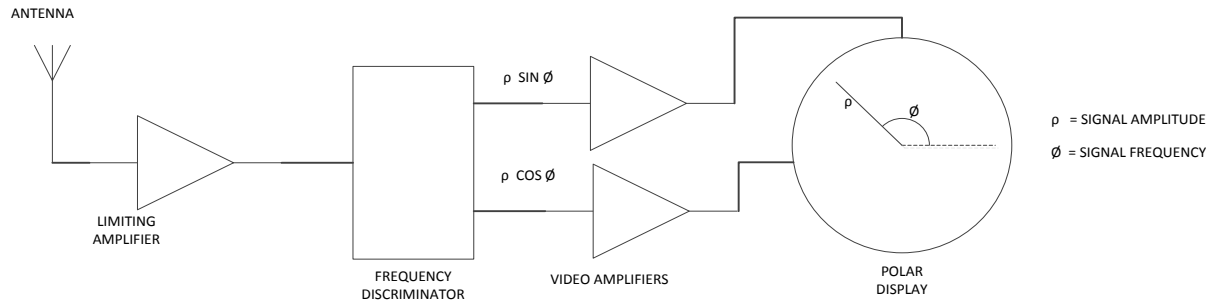


Figure 3: Basic IFM receiver components

2.3.1 Limiting Amplifiers

Limiting amplifiers are the answer to both dynamic range and sensitivity problems. Low power input signals can then be considerably amplified, whereas high power signals drive the amplifier to limiting so that one's FDs are not overdriven. The requirements placed on these limiting amplifiers are harsh, making them a very expensive module in an IFM receiver.

2.3.2 Frequency Discriminator

The function of the FD is to accept an RF input signal and produce two video signals, $\rho \sin \phi$ and $\rho \cos \phi$. ρ is a measure of the signal amplitude and ϕ is directly proportional to frequency.

The FD block shown in Figure 3 can be broken up into a power splitter, delay line and PD, as shown in Figure 4. In this form it is often called a delay line discriminator (DLD). It makes use of the well-established principle of measuring frequency in terms of phase delay when a signal is propagated down a transmission line of known length. Almost all IFM receivers reported in this literature make use of DLDs and were pioneered at Mullard Research Laboratories in England approximately 60 years ago [4]. The next subsections will discuss the DLD in more detail.

2.3.2.1 Basic Principles of the Delay Line Frequency Discriminator

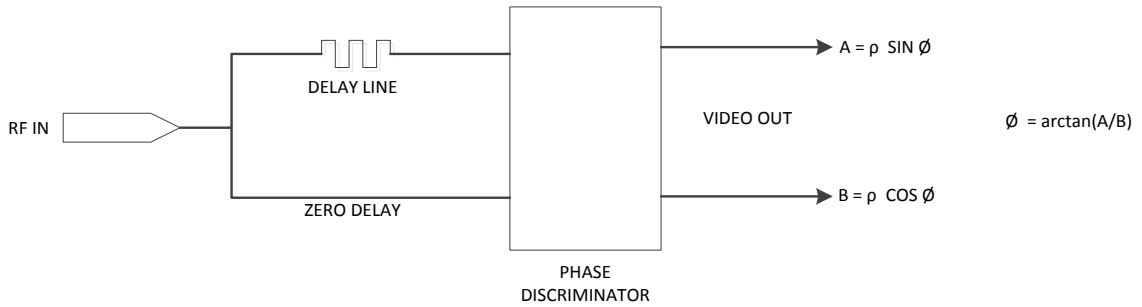


Figure 4: Delay Line Discriminator

If a delay line of length L and phase constant β is used, the phase delay φ is given by

$$\varphi = \beta L \quad (1)$$

If the signal wavelength is λ , then

$$\beta = \frac{2\pi}{\lambda} \quad (2)$$

If v is the velocity of propagation and f is the signal frequency, then

$$\lambda = \frac{v}{f} \quad (3)$$

Substituting (3) into (2)

$$\beta = \frac{2\pi}{v} f \quad (4)$$

Substituting (4) into (1)

$$\varphi = \frac{2\pi}{v} f L \quad (5)$$

Alternatively

$$\varphi = k f \quad (6)$$

Where k is a constant

$$k = \frac{2\pi}{v} L \quad (7)$$

Hence phase delay is directly proportional to signal frequency and can be used as a direct measure of frequency.

If f_1 and f_2 are the edge frequencies of the DLD's pass band, and $f_2 > f_1$, then from (6)

$$\varphi_1 = kf_1 \quad (8)$$

$$\varphi_2 = kf_2 \quad (9)$$

so that

$$\varphi_2 - \varphi_1 = k(f_2 - f_1) \quad (10)$$

The requirement for no ambiguity is $\varphi_2 - \varphi_1 < 2\pi$

(11)

If f_{BW} is the bandwidth of the DLD. i.e. $f_{BW} = f_2 - f_1$, then from (10)

$$f_{BW} = \frac{\varphi_2 - \varphi_1}{k} \quad (12)$$

The maximum unambiguous bandwidth is

$$f_{BW_{MAX}} = \frac{2\pi}{k} \quad (13)$$

Substituting for k from (7) one gets

$$f_{BW_{MAX}} = \frac{v}{L} \quad (14)$$

Equation (14) shows that the shorter the delay line length, the larger the unambiguous bandwidth. The frequency accuracy of the DLD then depends on the accuracy of the PD. If phase inaccuracies could be reduced without limit there would be no conflict between frequency accuracy and unambiguous bandwidth. In practice, however, an accuracy of even two degrees would be very difficult and expensive to realise.

From (5), an indication of the frequency accuracy is given by

$$\frac{d\varphi}{df} = \frac{2\pi}{v} L \quad (15)$$

Equation (15) shows that the longer the delay line, the more accurate the frequency measurement. This is in direct conflict with unambiguous bandwidth requirements and for a receiver of the type a trade-off would have to be chosen.

Work on building a broadband “wide-open” or non-tuning FD began at Mullard Research Laboratories in 1954. First results were not very successful, with the relationship between frequency and any other derived parameter being too non-linear and unrepeatably for accurate measurement.

The break-through and birth of modern delay lines FDs came in 1957 when the identity $\cos^2\left(\frac{\varphi}{2}\right) - \sin^2\left(\frac{\varphi}{2}\right) = \cos \varphi$ was used to measure the phase delay φ down a delay line of known length,

this being directly related to frequency. A second such operation employing an additional delay of $\frac{\pi}{2}$ radians at all frequencies provided the orthogonal $\sin \varphi$ component.

Figure 5 shows the basic components of a DLD. The delay line causes a delay τ and the delayed signal is multiplied with a direct signal and also with a signal shifted a fixed $\frac{\pi}{2}$ radians to give the sine and cosine outputs after low-pass filtering, as shown.

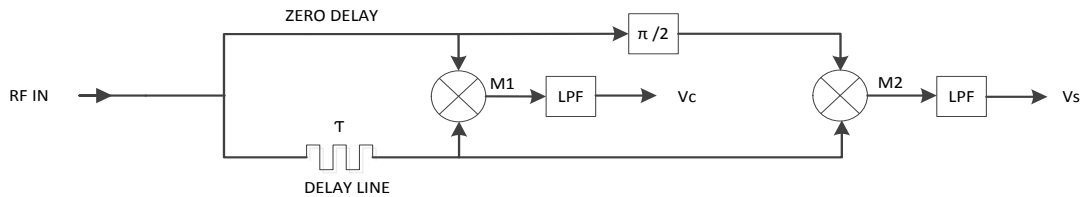


Figure 5: Basic frequency discriminator

Consider a fixed frequency, constant amplitude, sinusoidal input signal $V \cos(2\pi ft)$ where f is the signal frequency and V its amplitude. The output of the delay line is then $V \cos(2\pi ft - \varphi)$ where $\varphi = 2\pi f\tau$.

The mixing product M_1 is given by

$$M_1 = V \cos(2\pi ft) \times V \cos(2\pi ft - \varphi) \quad (16)$$

By using the product and sum trigonometric formulas, M_1 is also

$$M_1 = \frac{V^2}{2} \cos(\varphi) + \frac{V^2}{2} \cos(4\pi ft - \varphi) \quad (17)$$

Low-pass filtering rejects the sum frequency so that the output V_C is given by

$$V_C = \frac{V^2}{2} \cos(\varphi) \quad (18)$$

The inputs to the second mixer are $V \cos\left(2\pi ft - \frac{\pi}{2}\right)$ and $V \cos(2\pi ft - \varphi)$. The mixing product M_2 is then

$$M_2 = V \cos\left(2\pi ft - \frac{\pi}{2}\right) \times V \cos(2\pi ft - \varphi) \quad (19)$$

or

$$M_2 = \frac{V^2}{2} \cos\left(-\frac{\pi}{2} + \varphi\right) + \frac{V^2}{2} \cos\left(4\pi ft - \frac{\pi}{2} - \varphi\right) \quad (20)$$

Low-pass filtering and the use of identity $\cos\left(-\frac{\pi}{2} + \varphi\right) = \sin \varphi$ gives V_s as

$$V_s = \frac{V^2}{2} \sin(\varphi) \quad (21)$$

Besides the delay line itself, FDs are made up of combinations of hybrid quadrature couplers, in-phase power splitters/combiners and square-law detectors. Figure 6 shows the symbols used to represent these components.

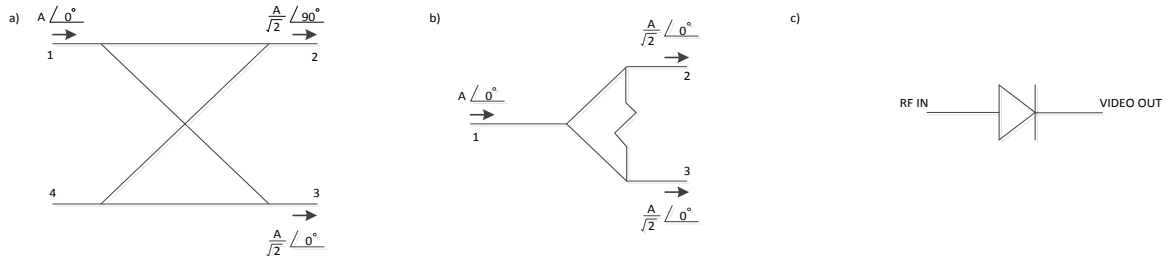


Figure 6: Standard FD component symbols

For the purpose of this part of the literature study, a quadrature coupler is a four-port device, as shown in Figure 6 (a). A signal incident at port 1 is evenly divided between ports 2 and 3, with a 90 degree phase difference between the two outgoing signals. No signal appears at port 4, as it is isolated from port 1. Similarly, a signal incident at port 4 is evenly split between ports 2 and 3, while port 1 is isolated. A signal incident at port 2 or 3 is divided equally between ports 1 and 4, ports 2 and 3 being isolated from each other. The two divided or outgoing signals always have a 90 degree phase difference.

Figure 6 (b) shows an in-phase power splitter/combiner in which a signal incident at port 1 is split equally between ports 2 and 3. In contrast with the quadrature coupler, the outgoing signals are in phase with each other. The form shown here is that of a single-stage Wilkinson coupler. The resistor shown causes port 2 to

be isolated from port 3. Since the device is passive it is reversible. Signals injected simultaneously at ports 2 and 3 are multiplied by the same $\frac{1}{\sqrt{2}}$ voltage factor and then summed vectorially at port 1. Isolation between ports 2 and 3 is maintained.

Figure 6 (c) shows the symbol used for a square-law detector. If an RF signal $A \cos(2\pi ft)$ is incident at the RF port then the squared result is $2KA^2 \cos^2(2\pi ft)$, which can be written as $K(A^2 + A^2 \cos 4\pi ft)$. $2K$ is the detector sensitivity. The detector output incorporates a low-pass filter (LPF) so that the output signal is KA^2 , which is a DC voltage if the input is CW and has no amplitude modulation.

Figure 7 shows the FD circuit configuration most frequently described in literature. It consists of a PD fed by delayed and undelayed outputs of an in-phase power splitter. The PD section is composed of three quadrature couplers and an in-phase power splitter. These feed four detectors, as shown.

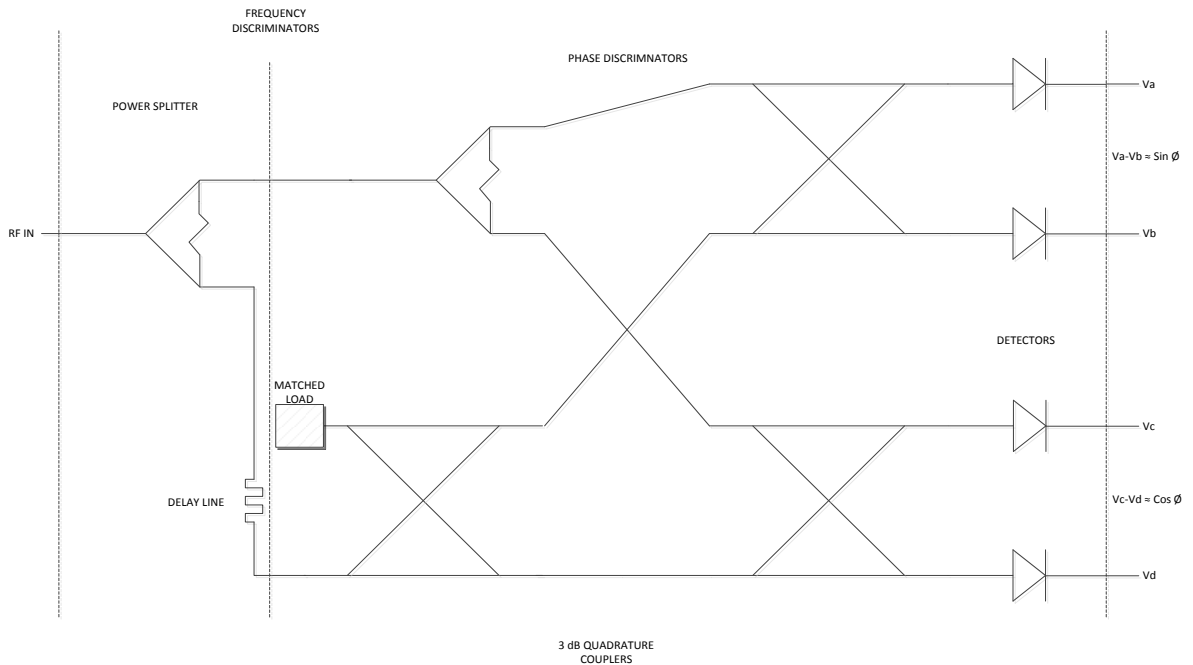


Figure 7: Most frequently described FD configuration

2.3.3 Polar Display

If the outputs from the FD, $\rho \sin \phi$ and $\rho \cos \phi$, are applied to the vertical and horizontal plates of a polar display, ρ is the radial component and the angular component ϕ can be scaled to read frequency directly. The FD is often referred to as a polar frequency discriminator because of the sine and cosine terms. Any amplitude or frequency modulation that is present will show up on the display as radial and angular fluctuations respectively.

It should be noted that although the polar display concept sounds relatively easy, it is a very complex task to take the outputs from the FD(s) and produce the correct frequency value. This effort should not be underestimated.

2.4 Digital IFM Receiver

Figure 8 is a block diagram of a typical digital IFM (DIFM) receiver. It shows the main functional elements. Most traditional IFM receivers will be of this form.

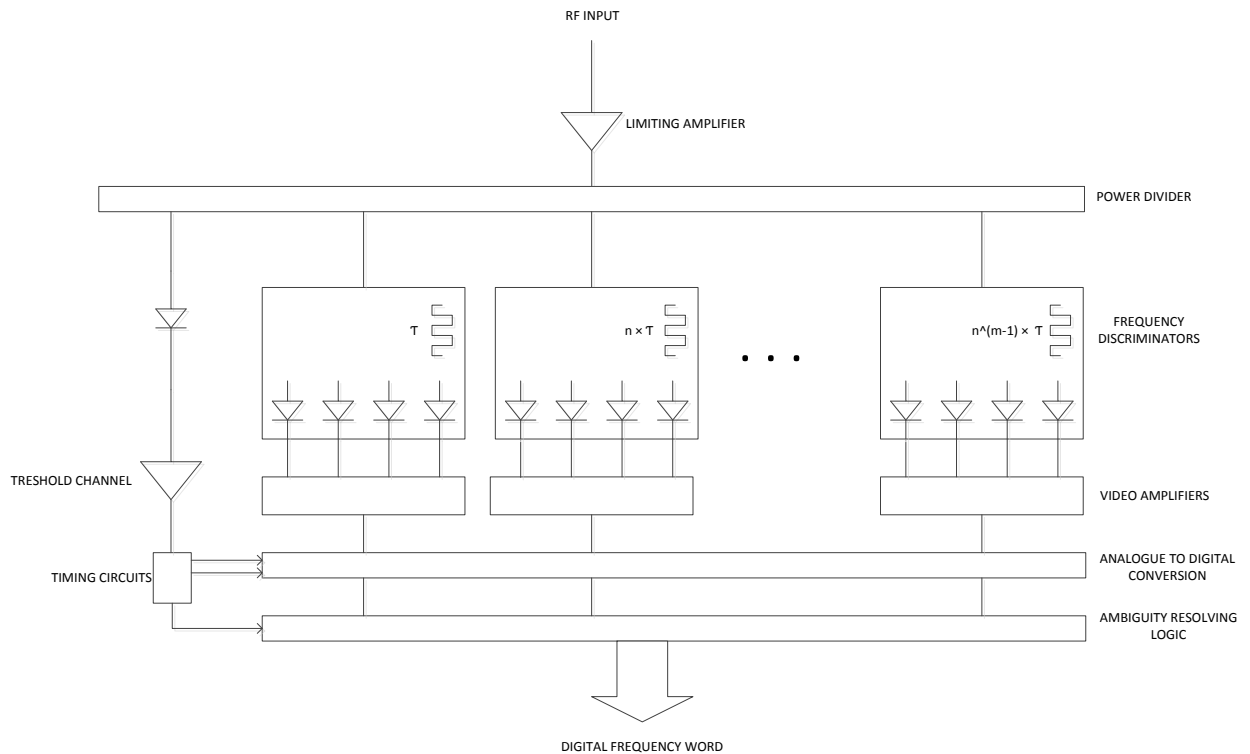


Figure 8: Digital IFM receiver

The filter coarsely defines the operating band of the RF input. The RF amplifier generally has relatively poor gain roll-off characteristics and the main function of the filter is to prevent the generation of spurious signals within the RF amplifier by high-power signals outside the operating band.

Typically 60 dB of RF gain may be required for maximum sensitivity. The amplifier should exhibit good pulse fidelity, low harmonic levels and good limiting at high signal levels. The need for limiting was discussed in 2.3.1.

The broadband power divider ensures that the signal is evenly split to the discriminators and the threshold channel. The video amplifiers may be AC-coupled, DC-restored or DC amplifier types. AC-coupled amplifiers are the simplest, but are insensitive to CW and high duty-cycle signals. DC types respond to CW but are affected by RF amplifier noise levels and may be blocked when CW signals are present.

The threshold channel detects the presence or arrival of signals and co-ordinates the analogue-to-digital conversion and logic circuitry sampling. The logic network combines the digital codes from the discriminator

quantisers to provide an accurate unambiguous code representing signal frequency. Some form of error correction or temperature compensation may be included here, but processing must be fast to allow a high pulse handling rate. The typical processing time for 10 to 12 bit frequency data is 150 ns [4],[6].

It was found that the term DIFM is ambiguous. Literature was found which also explains a DIFM as an all-digital implementation of the IFM in a FPGA [7], which is a not exactly as explained above.

2.5 Simultaneous Signals

Current IFM receivers offer no way of separating signals of different frequencies present in the FDs at the same time. The discussion of simultaneous signals is simplified by considering the case of just two signals present. Consider a receiver of the type described in the previous sections. A single signal produces sine and cosine terms at the inputs of the polar display. A second signal produces its own sine and cosine terms so that the display plots the sum of the two sine terms against the sum of the two cosine terms giving, in general, a result reflective of neither input. In fact, the result will be weighted towards the larger of the two signals and it is this signal that is usually of most interest.

The probability of this occurring when a CW signal is present may be quite high. If both signals are pulsed, then the probability of them being simultaneous is small. Nevertheless, it may be important to know the integrity of every measurement, as it is made so that doubtful results will be discarded and not acted upon. Many IFM receiver manufacturers include simultaneous signal detection (SSD) circuitry, which sets a flag whenever readings are determined to be in doubt.

Figure 9 shows the transition period with its characteristic dip, which some SSD circuitry uses to set a flag when in doubt. The reason for this dip is because of the mixing effect of the limiting amplifier. The non-linear characteristic of the limiting amplifier generates harmonics which reduce the power levels of the original frequencies. Sometime during the rise time of the stronger pulse the most harmonics will be generated by the limiting amplifier, because the most harmonics are generated when the power level of the signals are approximately the same [4]. Note that other methods is also used for SSD, this is not the only method used.

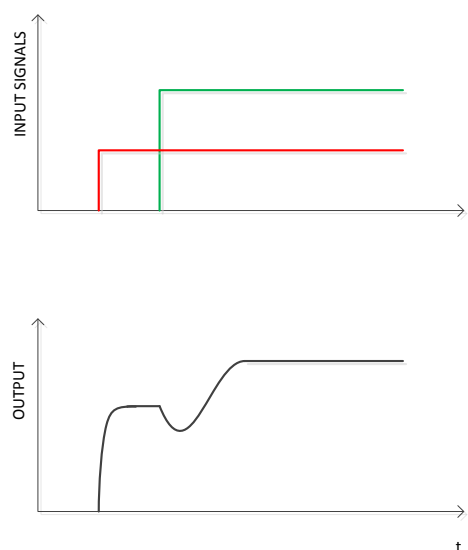


Figure 9: Characteristic dip during simultaneous signals

To demonstrate that this is still true for current IFM receivers, the following screenshots were captured from a test software application, which was developed to test an IFM receiver. The screenshots will clearly indicate how simultaneous signals resulted in a frequency measure that was wrong. The IFM receiver that was tested was from a leading IFM receiver manufacturer in Europe. This IFM receiver had a bandwidth from 2 to 18 GHz and a power dynamic range from -5dBm to -60 dBm.

This IFM receiver had SSD circuitry and to work around this feature, continuous wave signals were used.

Firstly a 3 GHz CW signal at a power level of -20 dBm was injected into the IFM receiver. Figure 10 shows that the IFM receiver measured the frequency of the signal correctly.

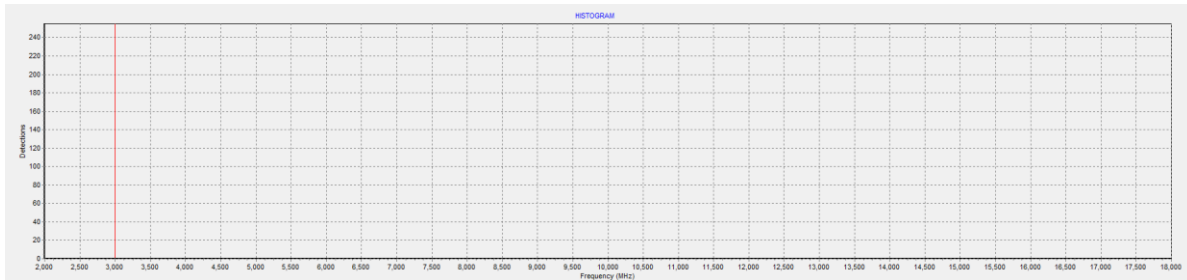


Figure 10: Measured 3 GHz CW source

Then a 17 GHz CW signal at a power level of -20 dBm was injected into the IFM receiver. Figure 11 shows that the IFM receiver measured the frequency of the signal correctly.

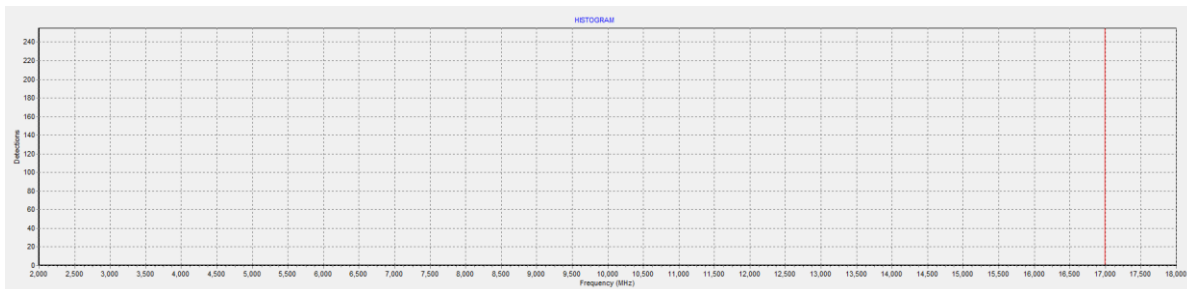


Figure 11: Measured 17 GHz CW source

Then a combined 3 GHz CW signal and 17 GHz CW signal (both at -20 dBm) were injected into the IFM receiver. Figure 12 shows that the IFM receiver measured that a signal was present at 2.6 GHz, which is incorrect. In this case the IFM receiver will steer the narrowband receiver towards 2.6 GHz, and then when the narrowband receiver does further analysis at 2.6 GHz, it will find no signals.

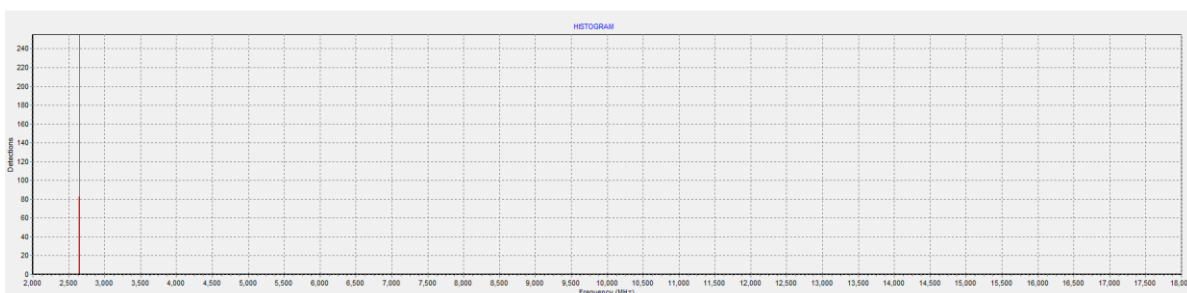


Figure 12: Measured 3 GHz and 17 GHz CW sources

3 Simulation of a Typical Digital IFM (DIFM) Receiver

3.1 Single Delay Line

A MATLAB simulation was implemented to simulate a typical (traditional) single-line DIFM receiver. Figure 13 shows a block diagram of a typical single-line DIFM receiver.

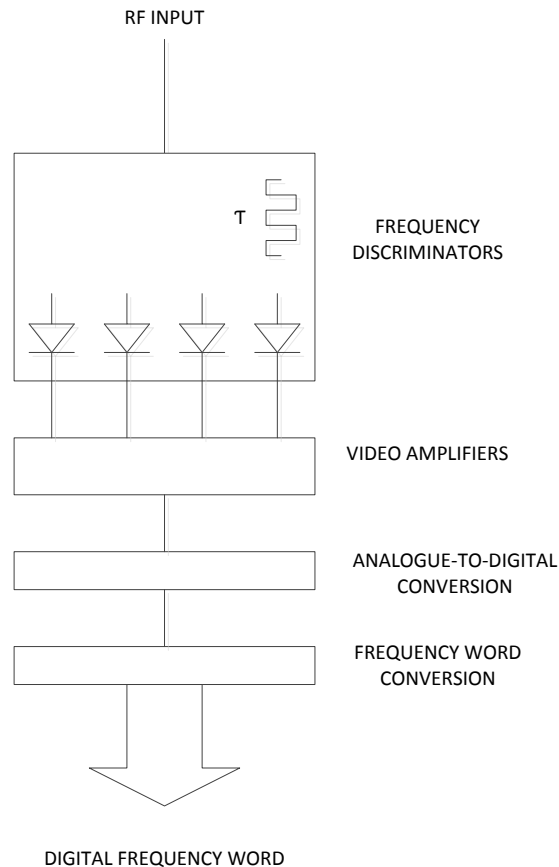


Figure 13: Single-line DIFM receiver

The simulation was implemented as follows:

- Generate time discrete amplitude samples for the input signal.
- Pass the time discrete signal through a Hilbert filter. This produces a complex time discrete signal.
- Choose/Calculate the length τ for an unambiguous phase difference of 2π for the desired input frequencies.
- Generate a delayed complex time discrete signal with delay τ .
- Multiply the non-delayed complex time discrete signal with the conjugate delayed complex time discrete signal to get rid of the sum frequency.
- Determine the angle between the real and imaginary part of the complex number.

- Convert the angle into a frequency value.

As mentioned the Hilbert filter is used to produce a complex time discrete signal from the real time discrete signal. Firstly a complex time discrete signal must be used to be able to calculate the outputs from the FD, $\rho \sin \varphi$ and $\rho \cos \varphi$. Also if a complex time discrete signal is used and the non-delayed complex time discrete signal is multiplied with the conjugate delayed complex time discrete signal the sum frequency shown in equation (17) and (20) is cancelled out. Therefore the need for a LPF is then not necessary.

See Appendix A for the source code of the MATLAB simulation.

3.1.1 Single Input Signal with a Frequency of 200 MHz

Firstly a single CW signal with a frequency of 200 MHz was simulated. Figure 14 shows the phase vs frequency plot. It is clear that the IFM receiver measured the frequency of the input signal at 200 MHz correctly.

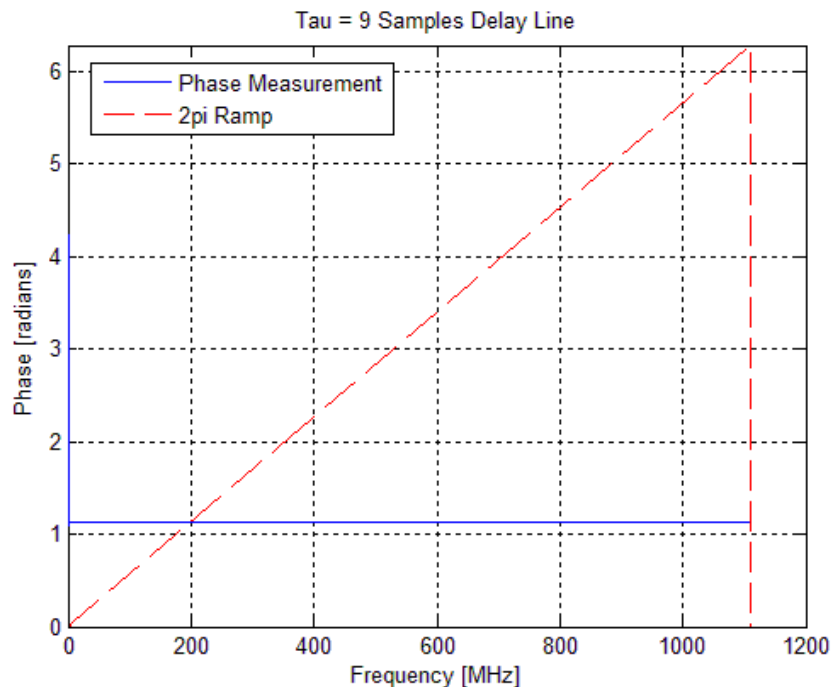


Figure 14: Frequency Output = 200 MHz

3.1.2 Single Input Signal with a Frequency of 1000 MHz

Then a single CW signal with a frequency of 1000 MHz was simulated. Figure 15 shows the phase vs frequency plot. It is clear that the IFM receiver measured the frequency of the input signal at 1000 MHz correctly.

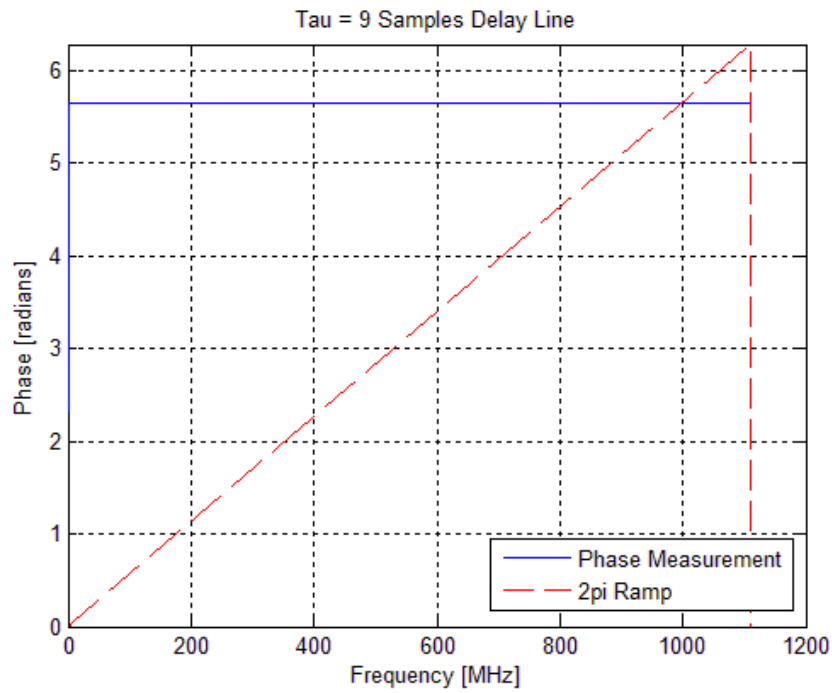


Figure 15: Frequency output = 1000 MHz

3.1.3 Two Input Signals with Frequencies of 200 MHz and 1000 MHz

Then two combined CW signals with frequencies of 200 MHz and 1000 MHz were simulated. Figure 16 shows the phase vs frequency plot. It is clear that the IFM receiver measured the frequency of the input signal at 45 MHz incorrectly.

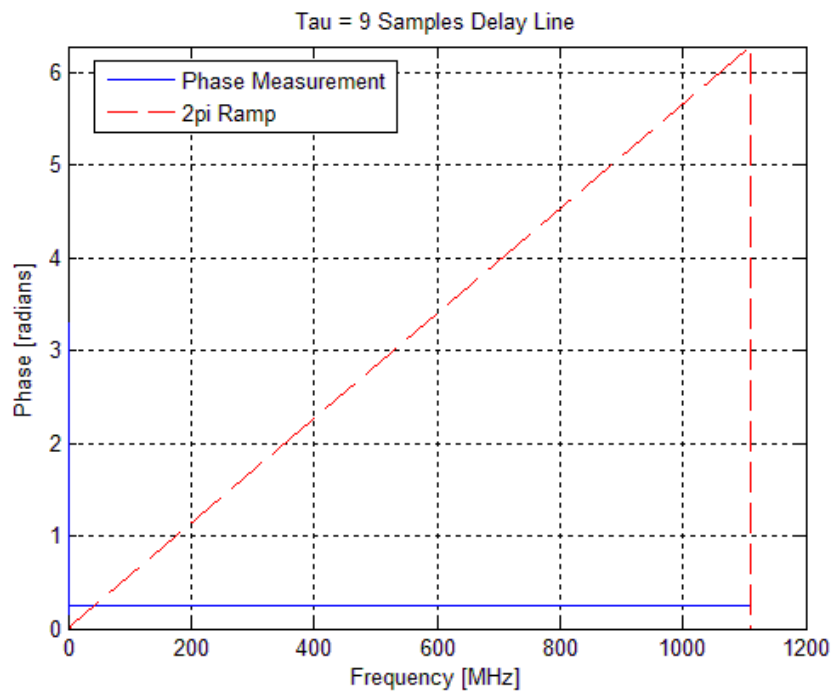


Figure 16: Frequency output = 45 MHz

3.2 Multiple Delay Lines

As explained in section 2.3.2.1, a shorter delay line length will give a larger unambiguous bandwidth and a longer delay line will give a more accurate frequency measurement. By using multiple delay lines, one can therefore have the best of both. One should use a short delay line to get a large unambiguous bandwidth together with a longer line to have more accurate frequency measurement [4].

With the long line one gets multiple answers with very accurate frequency measurement. Then one uses the shorter line to determine which one of the multiple answers is the correct one. This is called ambiguity resolving.

More delay lines (with lengths between longest and shortest) can be used to resolve ambiguities where the longest and the shortest line ratio is too high for the shortest line to resolve the longest line.

A MATLAB simulation was implemented to simulate a multiple delay line digital IFM receiver. Figure 17 shows a block diagram of a typical (traditional) multiple-line DIFM receiver.

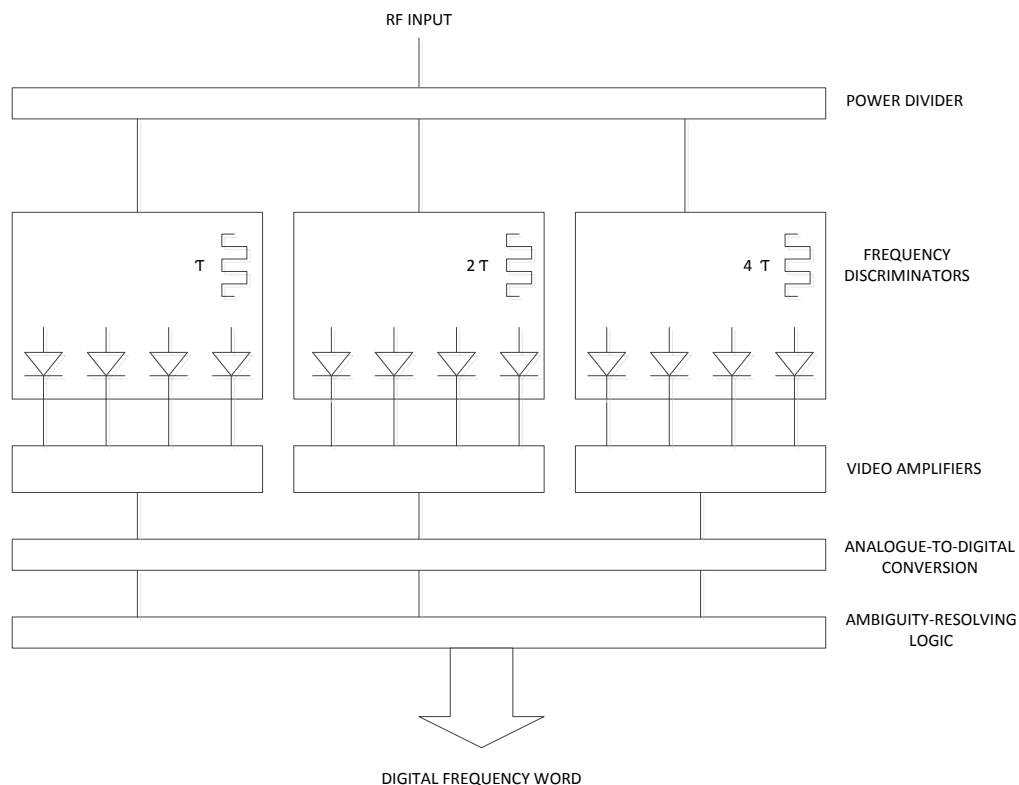


Figure 17: Multiple delay line DIFM receiver

The simulation was implemented as follows:

- Generate time discrete amplitude samples for the input signal.
- Pass the time discrete signal through a Hilbert filter. This produces a complex time discrete signal.
- Choose/Calculate the length τ for an unambiguous phase difference of 2π for the desired input frequencies.
- Generate three delayed complex time discrete signals with delays τ , 2τ and 4τ .

- Multiply the non-delayed complex time discrete signal with the three conjugate delayed complex time discrete signals to get rid of the sum frequency to get rid of the sum frequency.
- Determine the angles between the real and imaginary part of the complex numbers.
- Convert the angles to frequency values.

See Appendix B for the source code of the MATLAB simulation.

3.2.1 Single Input Signal with a Frequency of 200 MHz

Firstly a single CW signal with a frequency of 200 MHz was simulated. Figure 18 shows the phase vs frequency plots. In the 4τ plot (more accurate frequency measurement) 4 frequencies (200, 480, 755 and 1035 MHz) are possible solutions. In the 2τ plot 2 frequencies (200 and 755 MHz) are possible solutions. Therefore 480 and 1035 MHz can be ignored from the 4τ plot. In the τ plot (large unambiguous bandwidth) it can be seen that the frequency of 200 MHz is the only possible solution. Therefore the 755 MHz can also be ignored and so the IFM receiver measured the frequency of the input signal at 200 MHz correctly.

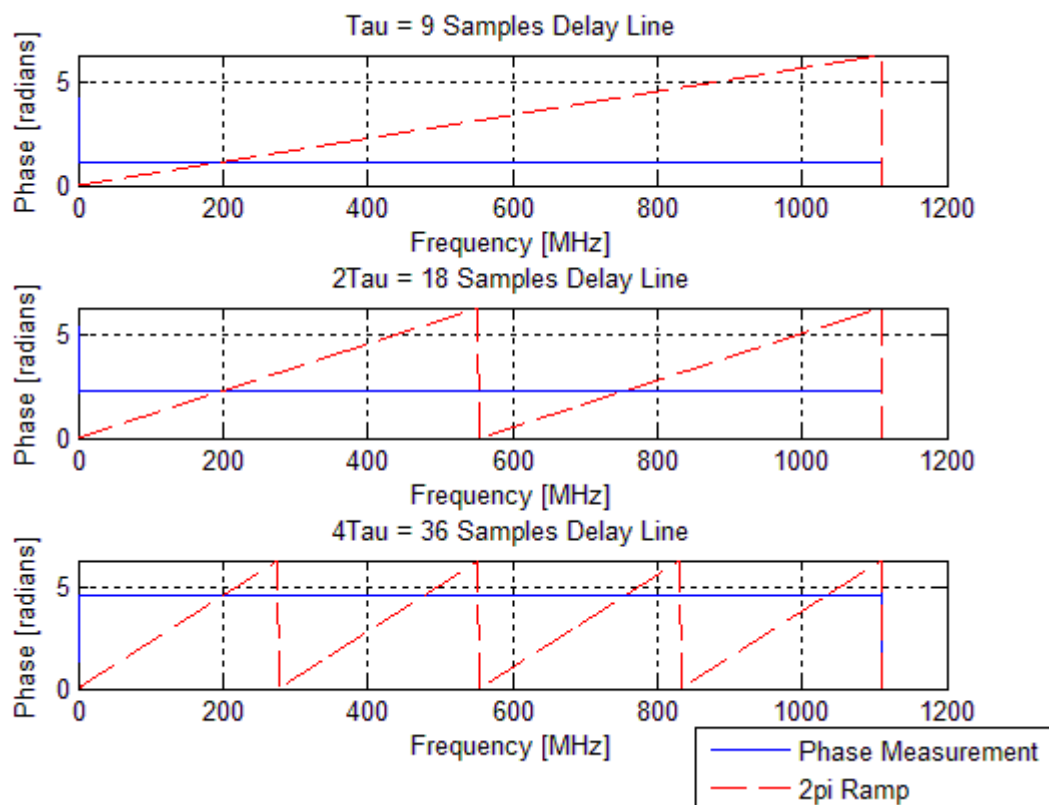


Figure 18: Frequency output = 200 MHz

3.2.2 Single Input Signal with a Frequency of 1000 MHz

Then a single CW signal with a frequency of 1000 MHz was simulated. Figure 19 shows the phase vs frequency plots. In the 4τ plot (more accurate frequency measurement) 4 frequencies (170, 445, 720 and 1000 MHz) are possible solutions. In the 2τ plot 2 frequencies (445 and 1000 MHz) are possible solutions. Therefore 170 and 720 MHz can be ignored from the 4τ plot. In the τ plot (large unambiguous bandwidth) it can be seen that the frequency of 1000 MHz is the only possible solution. Therefore 445 MHz can also be ignored and so the IFM receiver measured the frequency of the input signal at 1000 MHz correctly.

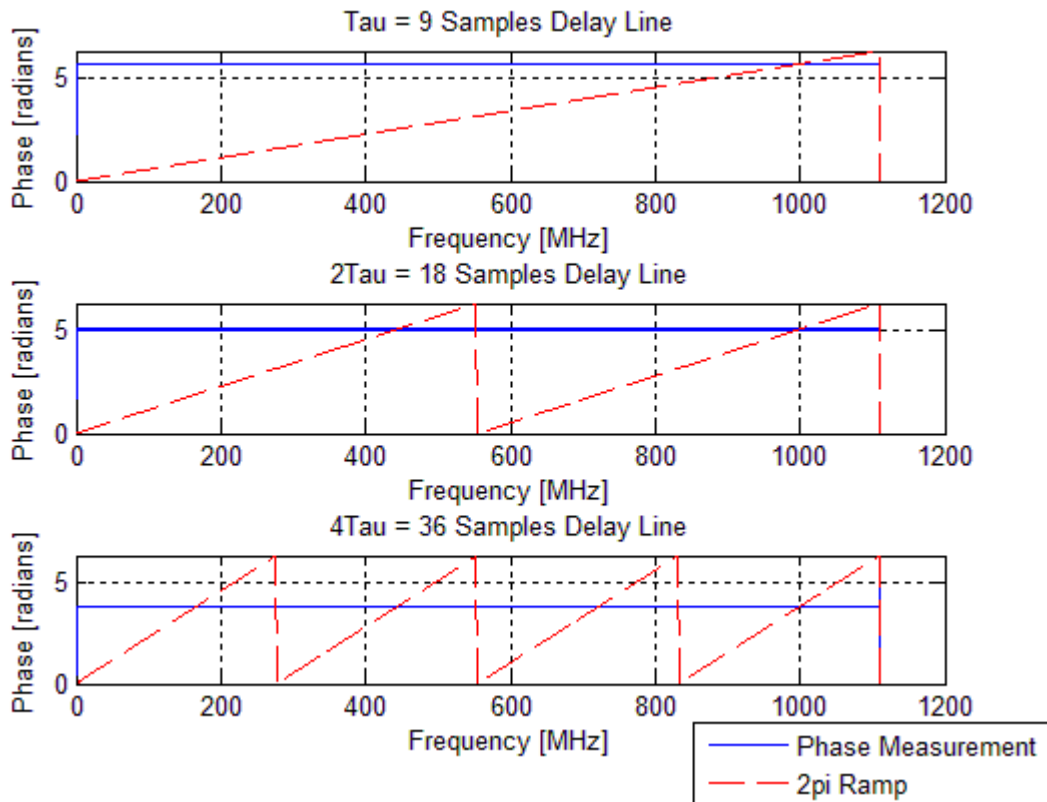


Figure 19: Frequency output = 1000 MHz

3.2.3 Two Input Signals with Frequencies of 200 MHz and 1000 MHz

Then an input signal consisting of two CW signals with frequencies at 200 MHz and 1000 MHz was simulated. Figure 20 shows the phase vs frequency plot. In the 4τ plot 4 frequencies (180, 460, 740 and 1020 MHz) are possible solutions. In the 2τ plot 2 frequencies (320 and 880 MHz) are possible solutions. In the τ plot it can be seen that the frequency of 40 MHz is the only possible solution. It is clear that the IFM receiver cannot resolve the frequency of the input signal correctly, giving non matching solutions for the different delay lines.

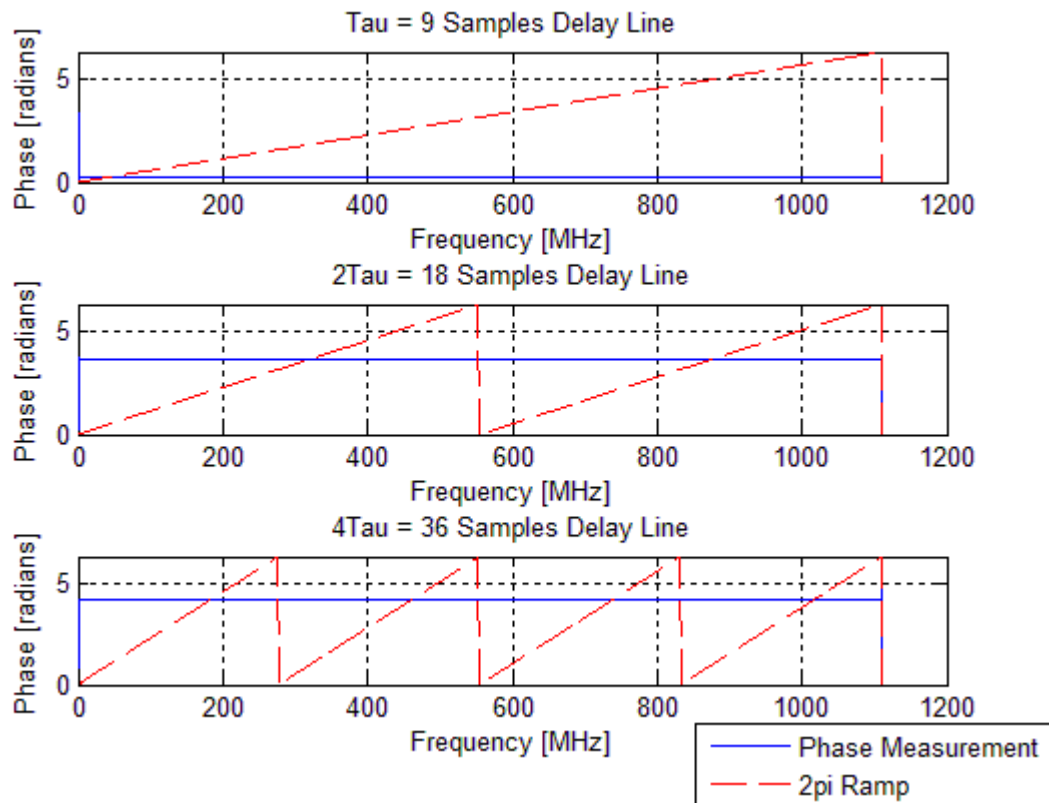


Figure 20: Angle resolving between input signals and delayed versions

3.3 Typical Frequency Error Caused by Simultaneous Signals

To gain better understanding of the frequency error caused by simultaneous signals, the simulations described below were done.

The single delay line IFM was used. Two combined signals were injected. One signal was at a fixed frequency of 200 MHz and the other signal at a sweeping frequency from 200 – 1000 MHz. Both signals were at the same power level.

As seen in Figure 21, when the two input frequencies is within 250 MHz from each other the measured frequency is in the centre of to the actual two input frequencies. Also refer to Figure 22. When the two input frequencies are moving further apart from each other, there is a point where the measured frequency jumps between two frequency values (maximum and minimum). Also refer to Figure 23 to understand the jump between the two measured frequency values. Note that both these frequency values are not close to the actual input frequencies. If the input frequencies are moved further apart, a point is reached where the measured frequency stops jumping, but is not close to the actual input frequencies.

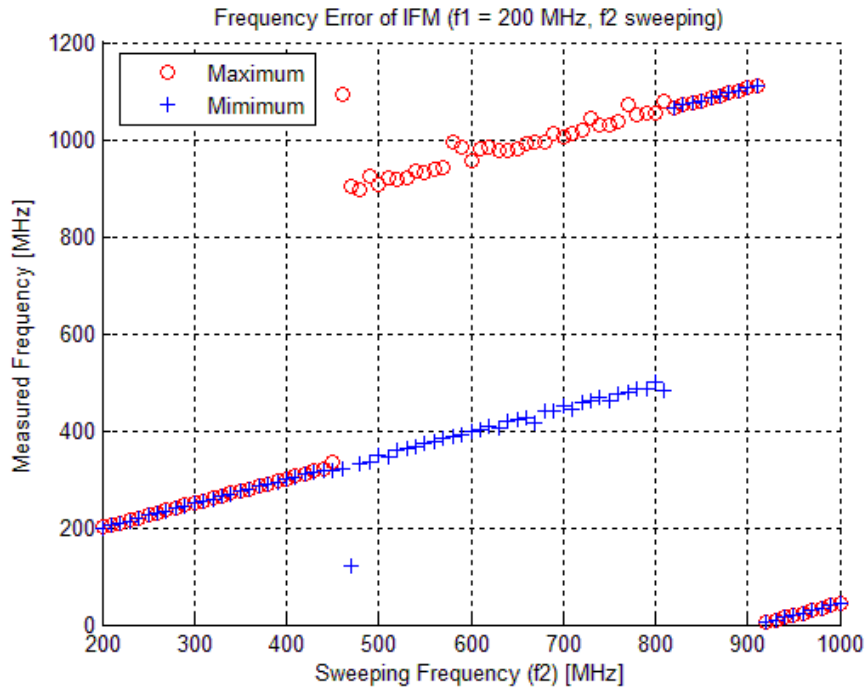


Figure 21: Measured frequency ($f_1 = 200$ MHz, f_2 Sweeping)

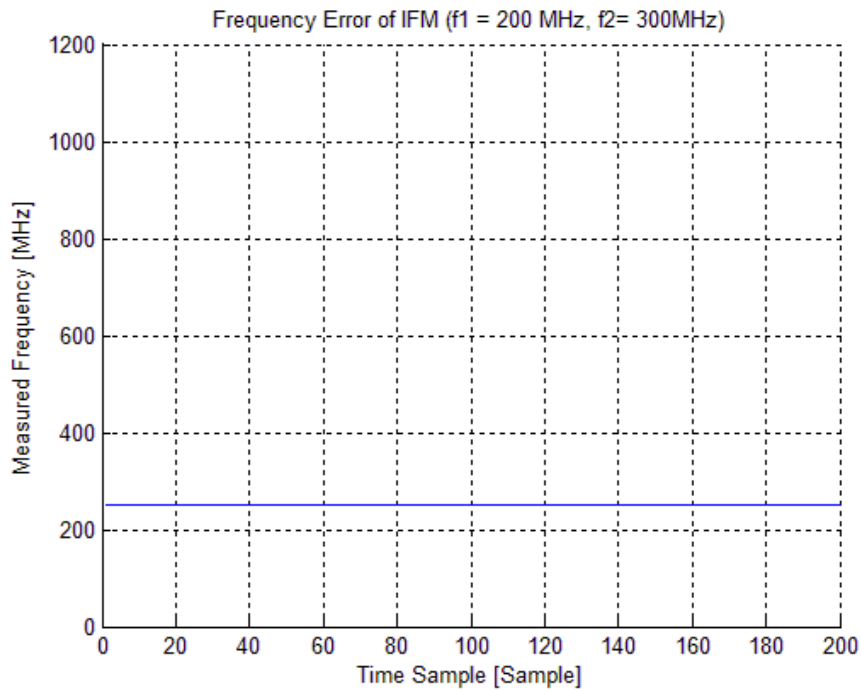


Figure 22: Measured frequency ($f_1 = 200$ MHz, $f_2 = 300$ MHz)

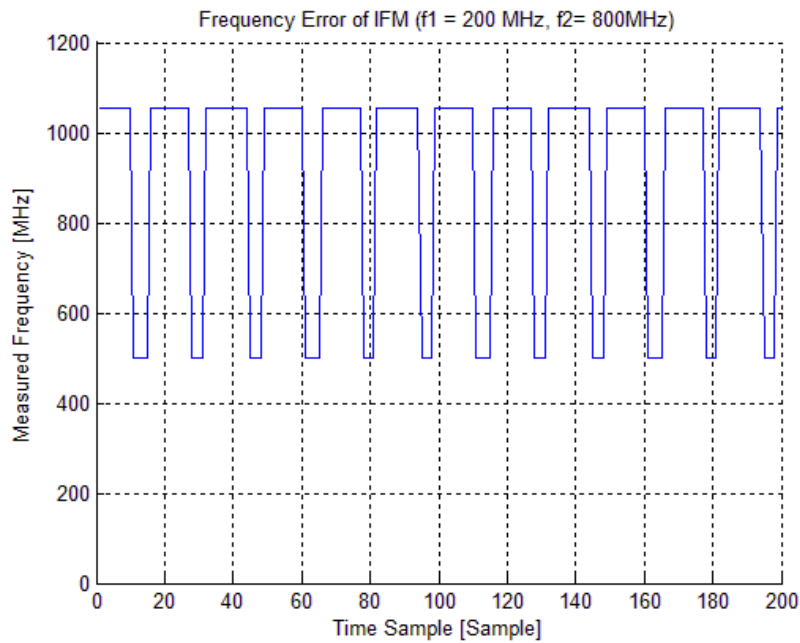


Figure 23: Measured frequency (f1 = 200 MHz, f2 = 800 MHz)

To demonstrate the effect that a difference in power level has, the following simulation was done: Two combined signals were injected. One signal was at a fixed frequency of 200 MHz and the other signal at a fixed frequency of 800 MHz. The power difference between the two signals was then changed, starting with the same power level and then decreasing the power level of the 800 MHz input signal.

Figure 24 clearly indicates that if the power level is the same, the frequency error is big. As the power difference becomes bigger, the measured frequency gets closer to 200 MHz. Note that the measured frequency also jumps between two frequency values, causing a maximum and minimum frequency measurement.

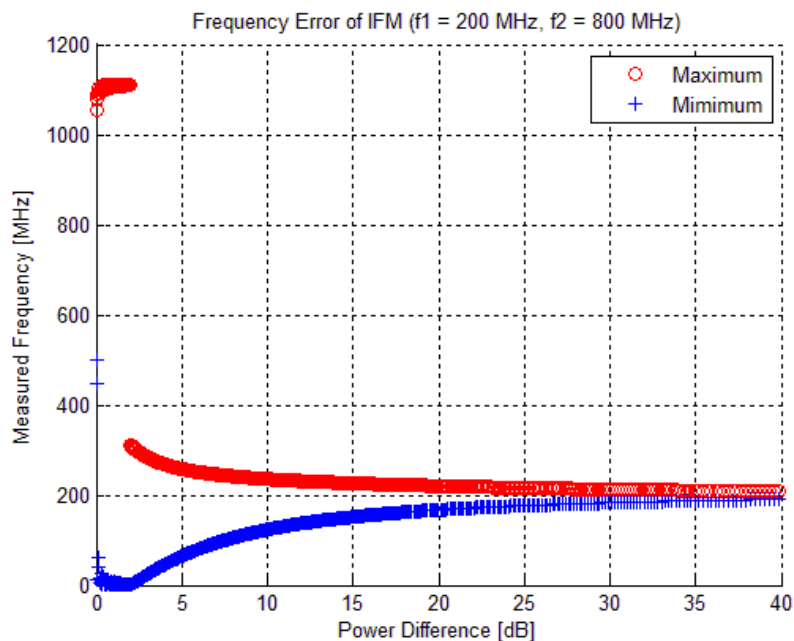


Figure 24: Measured frequency vs power difference

Note that these results are for this specific single delay line IFM. A different configuration will give different frequency values.

3.4 Effect of Phase Modulation on the IFM

To gain better understanding of the influence which phase modulations have on the IFM, the simulations described below were done. The single delay line IFM was used.

In the first simulation an input signal with a frequency of 200 MHz of which the phase was continuously increased, which resulted in a chirp signal from 200 MHz to 1000 MHz, was used.

In Figure 25 it can be seen that the phase difference between the non-delayed and delayed signal is measured correctly, which resulted in a correct frequency measurement as seen in Figure 26.

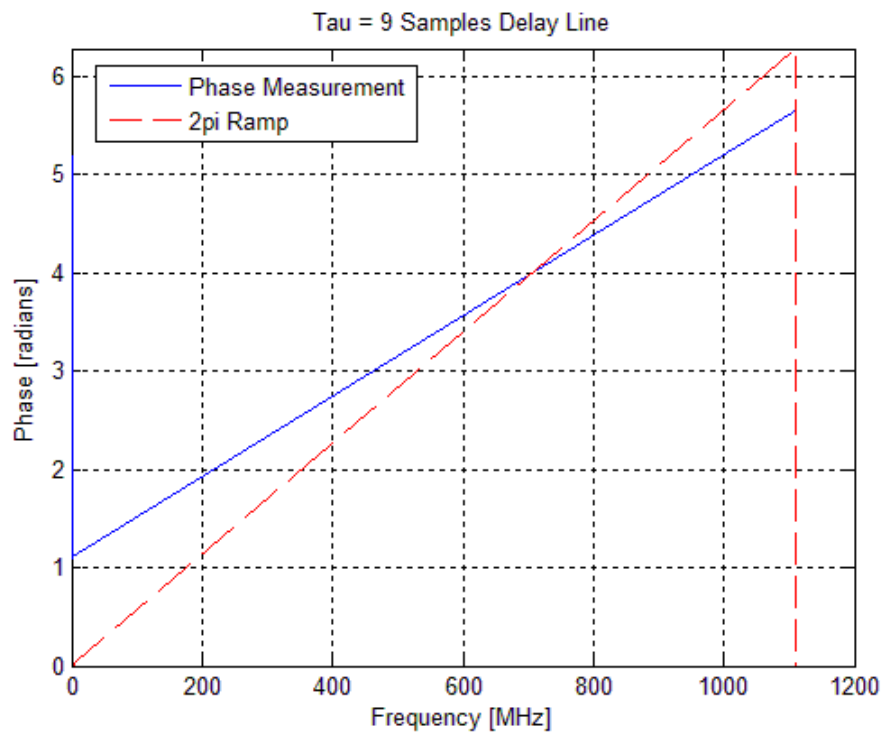


Figure 25: Phase measurement of a chirp signal

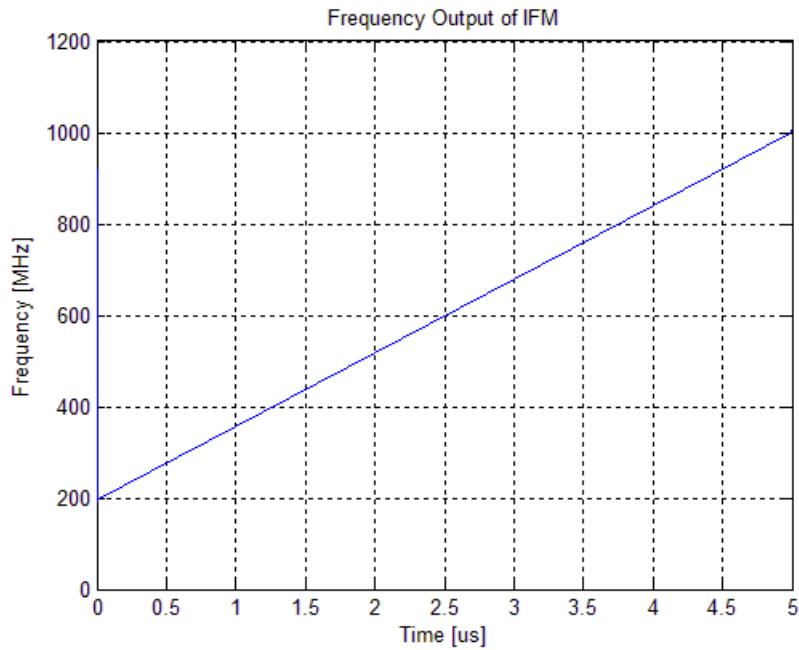


Figure 26: Frequency measurement of a chirp signal

Secondly an input signal with a frequency of 1000 MHz which contained two 180° phase jumps was used. This would be similar as a phase jumps used in barker or frank code modulations.

In Figure 27 it can be seen that the 180° phase jumps caused jumps in the phase difference between the non-delayed and delayed signal which resulted in an incorrect frequency measurement for about 2 ns as seen in Figure 28. The reason for this is because there is short time which the non-delayed signal is after the 180° and the delayed signal is still the before the 180° jump.

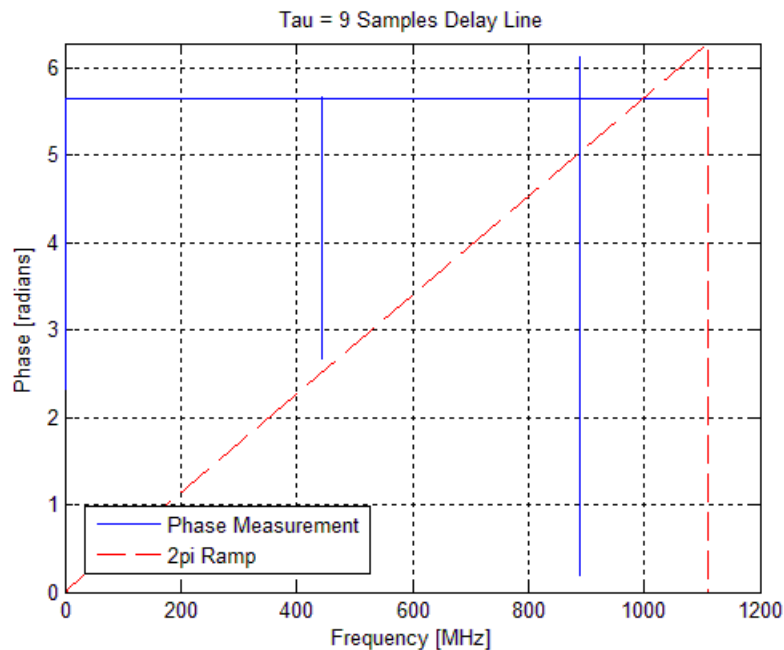


Figure 27: Phase measurement during a phase jump

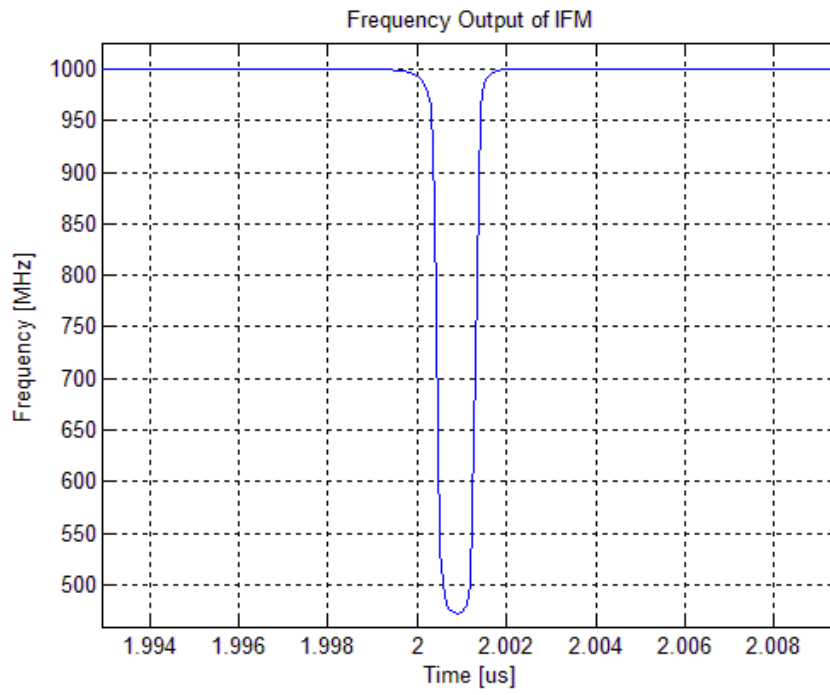


Figure 28: Frequency result during a phase jump

4 Investigation of a possible solution

During the literature study research, literature was discovered that proposed Prony's method as a solution to solve the issue of simultaneous signals [5],[6]. Literature proposing Prony's method submitted as early as 1989 was found [5]. However, no evidence of the use of this method could be found in current commercial IFM receivers, indicating that the implementation of Prony's method is the difficult part.

For this method the FDs are exactly the same as in the traditional IFM receiver. The differences are in the choice of delay lines and the ambiguity-resolving (frequency resolving) part.

4.1 Prony's Method

4.1.1 Discriminator Outputs and Simultaneous Signal Condition

As explained in 2.3, the FD in an IFM receiver consists of power splitters, delay lines, couplers and video detectors. The output of the discriminator with delay time τ and input signal $A \sin(\omega t)$ where A is the amplitude and ω is the angular frequency can be written as

$$R_r(\tau) = \frac{A^2}{2} \cos(\omega\tau)$$

$$R_i(\tau) = \frac{A^2}{2} \sin(\omega\tau)$$

The output of the discriminator can also be written in complex form as

$$R(\tau) = R_r(\tau) + jR_i(\tau) = P \exp(j\omega\tau) \text{ where } j \text{ represents the imaginary unit, which satisfies the equation } j^2 = -1.$$

The input frequency can be found as

$$\omega = (1/\tau) \tan^{-1}(R_i / R_r) \text{ as explained in the case of the typical IFM receiver.}$$

If one considers the case where two signals are present $V_1 \cos(2\pi f_1 t)$ and $V_2 \cos(2\pi f_2 t)$, where f_1 is the signal frequency and V_1 is the amplitude of the one signal and f_2 is the signal frequency and V_2 is the amplitude of the other signal.

Then equation (17) can be written as

$$\begin{aligned}
 M_1 = & \left(\frac{V_1^2}{2} + \frac{V_2^2}{2} \right) \cos(\varphi) + \frac{V_1^2}{2} \cos(4\pi f_1 t - \varphi) + V_1 V_2 \cos(2\pi(f_1 + f_2)t - \varphi) + \\
 & V_1 V_2 \cos(2\pi(f_1 - f_2)t + \varphi) + \frac{V_2^2}{2} \cos(4\pi f_2 t - \varphi)
 \end{aligned} \tag{22}$$

and equation (20) as

$$\begin{aligned}
 M_2 = & \left(\frac{V_1^2}{2} + \frac{V_2^2}{2} \right) \cos\left(-\frac{\pi}{2} + \varphi\right) + \frac{V_1^2}{2} \cos\left(4\pi f_1 t - \frac{\pi}{2} - \varphi\right) + V_1 V_2 \cos\left(2\pi(f_1 + f_2)t - \frac{\pi}{2} - \varphi\right) + \\
 & V_1 V_2 \cos\left(2\pi(f_1 - f_2)t - \frac{\pi}{2} + \varphi\right) + \frac{V_2^2}{2} \cos\left(4\pi f_2 t - \frac{\pi}{2} - \varphi\right)
 \end{aligned} \tag{23}$$

If the frequency separation of the two input signals is far apart, such that the difference frequency $(f_1 - f_2)$ will also be filtered out by the low-pass video filter in the discriminator, the outputs from the discriminator are

$$R_r(\tau) = P_1 \cos(\omega_1 \tau) + P_2 \cos(\omega_2 \tau)$$

$$R_i(\tau) = P_1 \sin(\omega_1 \tau) + P_2 \sin(\omega_2 \tau)$$

If equation $R(\tau) = R_r(\tau) + jR_i(\tau) = P \exp(j\omega\tau)$ is used to find the frequency, the result will be erroneous.

4.1.2 Two Simultaneous Signals

A solution for the simultaneous signals problem in traditional IFM receivers can be found in Prony's method. In order to solve the simultaneous problem, more discriminators with different delays are needed. For four discriminators with different delays τ , 2τ , 3τ and 4τ , the outputs from the discriminators can be written as:

$$R(\tau) = P_1 \exp(j\omega_1 \tau) + P_2 \exp(j\omega_2 \tau)$$

$$R(2\tau) = P_1 \exp(j\omega_1 2\tau) + P_2 \exp(j\omega_2 2\tau)$$

$$R(3\tau) = P_1 \exp(j\omega_1 3\tau) + P_2 \exp(j\omega_2 3\tau)$$

$$R(4\tau) = P_1 \exp(j\omega_1 4\tau) + P_2 \exp(j\omega_2 4\tau)$$

(24)

Linear prediction can be used to predict future values of time-discrete signals. If linear prediction is used, one can define two constants, a_1 and a_2 , such that they satisfy the following equations:

$$\begin{aligned} -R(4\tau) &= a_1 R(3\tau) + a_2 R(2\tau) \\ -R(3\tau) &= a_1 R(2\tau) + a_2 R(\tau) \end{aligned} \quad (25)$$

From the above equation, Cramer's rule can be used to solve a_1 and a_2 . By substituting equation (24) into equation (25), it can be shown that

$$\begin{aligned} a_1 &= -[\exp(jw_1\tau) + \exp(jw_2\tau)] = -(z_1 + z_2) \\ a_2 &= \exp[j(w_1 + w_2)\tau] = z_1 z_2 \end{aligned} \quad (26)$$

where

$$z_i = \exp(jw_i\tau) \quad (27)$$

and $i = 1, 2$. If z_i can be solved, the frequencies of w_1 and w_2 can be obtained.

From equation (26), it can be seen that z will satisfy the following function as

$$\begin{aligned} (z - z_1)(z - z_2) &= z^2 - (z_1 + z_2)z + z_1 z_2 \\ &= z^2 + a_1 z + a_2 = 0 \end{aligned} \quad (28)$$

Therefore one can find z by solving the above equations.

One can summarise the steps of solving two simultaneous signals as follows:

- Four discriminators with proper delays are needed.
- The values of a_i can be calculated.
- From equation (28), the values of z can be calculated and the individual frequency can be calculated from equation (27).

In this case where two simultaneous signals are expected, but only one signal is present, the problem can be solved from:

$$\det(R) = \begin{vmatrix} R(3\tau) & R(2\tau) \\ R(2\tau) & R(\tau) \end{vmatrix} = 0 \quad (29)$$

By doing this check, it can be determined if there is one signal or two signals. Noise in the system will cause the check to be very small but not equal to 0. For this condition, the check should be tested against a predetermined value.

4.1.3 Three Simultaneous Signals

In the case of three simultaneous signals, one can choose six discriminators with different delays τ , 2τ , 3τ , 4τ , 5τ and 6τ . The outputs from the discriminators can be written as

$$\begin{aligned}
 R(\tau) &= P_1 \exp(jw_1\tau) + P_2 \exp(jw_2\tau) + P_3 \exp(jw_3\tau) \\
 R(2\tau) &= P_1 \exp(j2w_1\tau) + P_2 \exp(j2w_2\tau) + P_3 \exp(j2w_3\tau) \\
 R(3\tau) &= P_1 \exp(j3w_1\tau) + P_2 \exp(j3w_2\tau) + P_3 \exp(j3w_3\tau) \\
 R(4\tau) &= P_1 \exp(j4w_1\tau) + P_2 \exp(j4w_2\tau) + P_3 \exp(j4w_3\tau) \\
 R(5\tau) &= P_1 \exp(j5w_1\tau) + P_2 \exp(j5w_2\tau) + P_3 \exp(j5w_3\tau) \\
 R(6\tau) &= P_1 \exp(j6w_1\tau) + P_2 \exp(j6w_2\tau) + P_3 \exp(j6w_3\tau)
 \end{aligned} \tag{30}$$

If linear prediction is used, one can define three constants (a_1 , a_2 and a_3) such that they satisfy the following equations

$$\begin{aligned}
 -R(6\tau) &= a_1 R(5\tau) + a_2 R(4\tau) + a_3 R(3\tau) \\
 -R(5\tau) &= a_1 R(4\tau) + a_2 R(3\tau) + a_3 R(2\tau) \\
 -R(4\tau) &= a_1 R(3\tau) + a_2 R(2\tau) + a_3 R(1\tau)
 \end{aligned} \tag{31}$$

From the above equation, Cramer's rule can be used to solve a_1 , a_2 and a_3 .

By substituting equation (30) into equation (31), it can be shown that

$$\begin{aligned}
 a_1 &= -(z_1 + z_2) \\
 a_2 &= z_1 z_2 \\
 a_3 &= -z_1 z_2 z_3
 \end{aligned} \tag{32}$$

where

$$z_i = \exp(jw_i\tau) \tag{33}$$

and $i = 1, 2, 3$. If z_i can be solved, the frequencies of w_1 , w_2 and w_3 can be obtained.

From equation (32), it can be seen that z will satisfy the following function as

$$(z - z_1)(z - z_2)(z - z_3) = z^3 + a_1 z^2 + a_2 z + a_3 = 0 \tag{34}$$

Therefore one can find z by solving the above equations.

One can summarise the steps of solving two simultaneous signals as follows:

- Six discriminators with proper delays are needed.
- The values of a_i can be calculated.
- From equation (34), the values of z can be calculated and the individual frequency can be calculated from equation (33).

In this case where three simultaneous signals are expected, but only two signals are present, the problem can be solved from:

$$\det(R) = \begin{vmatrix} R(5\tau) & R(4\tau) & R(3\tau) \\ R(4\tau) & R(3\tau) & R(2\tau) \\ R(3\tau) & R(2\tau) & R(\tau) \end{vmatrix} = 0 \quad (35)$$

By doing this check, it can be determined if there is three signals present. Noise in the system will cause the check to be very small but not equal to 0. For this condition, the check should be tested against a predetermined value.

4.1.4 Four Simultaneous Signals

In the case of four simultaneous signals, one can choose eight discriminators with different delays τ , 2τ , 3τ , 4τ , 5τ , 6τ , 7τ and 8τ .

If linear prediction is used, one can define four constants (a_1 , a_2 , a_3 and a_4) such that they satisfy the following equations:

$$\begin{aligned} -R(8\tau) &= a_1R(7\tau) + a_2R(6\tau) + a_3R(5\tau) + a_4R(4\tau) \\ -R(7\tau) &= a_1R(6\tau) + a_2R(5\tau) + a_3R(4\tau) + a_4R(3\tau) \\ -R(6\tau) &= a_1R(5\tau) + a_2R(4\tau) + a_3R(3\tau) + a_4R(2\tau) \\ -R(5\tau) &= a_1R(4\tau) + a_2R(3\tau) + a_3R(2\tau) + a_4R(\tau) \end{aligned} \quad (36)$$

From the above equation, Cramer's rule can be used to solve a_1 , a_2 , a_3 and a_4

It can then be shown that

$$\begin{aligned}
 a_1 &= -(z_1 + z_2) \\
 a_2 &= z_1 z_2 \\
 a_3 &= -z_1 z_2 z_3 \\
 a_4 &= z_1 z_2 z_3 z_4
 \end{aligned} \tag{37}$$

where

$$z_i = \exp(jw_i\tau) \tag{38}$$

and $i = 1, 2, 3, 4$. If z_i can be solved, the frequencies of w_1 , w_2 , w_3 and w_4 can be obtained.

From equation (37), it can be seen that z will satisfy the following function as

$$(z - z_1)(z - z_2)(z - z_3)(z - z_4) = z^4 + a_1 z^3 + a_2 z^2 + a_3 z + a_4 = 0 \tag{39}$$

Therefore one can find z by solving the above equations.

One can summarise the steps of solving two simultaneous signals as follows:

- Eight discriminators with proper delays are needed.
- The values of a_i can be calculated.
- From equation (39), the values of z can be calculated and the individual frequency can be calculated from equation (38).

In this case where four simultaneous signals are expected, but only three signals are present, the problem can be solved from:

$$\det(R) = \begin{vmatrix} R(7\tau) & R(6\tau) & R(5\tau) & R(4\tau) \\ R(6\tau) & R(5\tau) & R(4\tau) & R(3\tau) \\ R(5\tau) & R(4\tau) & R(3\tau) & R(2\tau) \\ R(4\tau) & R(3\tau) & R(2\tau) & R(\tau) \end{vmatrix} = 0 \tag{40}$$

By doing this check, it can be determined if there is four signals present. Noise in the system will cause the check to be very small but not equal to 0. For this condition, the check should be tested against a predetermined value.

4.1.5 N Simultaneous Signals

For N signals, the approach explained above can be generalised. $2N$ discriminators with delays from τ to $2N\tau$ in steps of τ are required. The constant a_i can be found from the following equations:

$$\begin{aligned}
 -R(2N\tau) &= a_1R[(2N-1)\tau] + a_2R[(2N-2)\tau] + \dots + a_NR(N\tau) \\
 -R[(2N-1)\tau] &= a_1R[(2N-2)\tau] + a_2R[(2N-3)\tau] + \dots + a_NR[(N-1)\tau] \\
 &\vdots \\
 -R[(N+1)\tau] &= a_1R(N\tau) + a_2R[(N-1)\tau] + \dots + a_NR(\tau)
 \end{aligned}$$

where a_i can be written as

$$\begin{aligned}
 a_1 &= -\sum_{i=1}^N \exp(jw_i\tau) \\
 a_2 &= \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \exp(jw_i\tau) \exp(jw_k\tau) \quad i = k \\
 &\vdots \\
 a_N &= (-1)^N \prod_{i=1}^N \exp(jw_i\tau)
 \end{aligned}$$

The corresponding z_i (where $i = 1, 2, \dots, N$) can be found from

$$z^N + a_1z^{N-1} + \dots + a_N = 0$$

It is important to note that the closed-form solutions of the above polynomial equations can only be found up to the fourth order. For equations with a higher order, numerical methods must be used to solve the equations.

4.1.6 Alternative choices of discriminator lags

As explained above, in order to solve N frequencies, $2N$ FDs are needed.

However, an FD generates the real and the imaginary part of the discriminator separately. The outputs from the discriminators can be combined differently [5].

As explained in section 4.1.1,

$$R(\tau) = R_r(\tau) + jR_i(\tau) = P \exp(j\omega\tau)$$

where

$$R_r(\tau) = \frac{A^2}{2} \cos(\omega\tau)$$

$$R_i(\tau) = \frac{A^2}{2} \sin(\omega\tau)$$

Note that

$$\begin{aligned}
R(-\tau) &= R_r(-\tau) + jR_i(-\tau) \\
&= P \cos(-w\tau) + P \sin(-w\tau) \\
&= P \cos(w\tau) - j \sin(w\tau)
\end{aligned}$$

This shows that one can obtain $R(\tau)$ and $R(-\tau)$ from the same discriminator with delay τ and also $R(n\tau)$ and $R(-n\tau)$ from the same discriminator with delay $n\tau$.

Using this idea, one can choose three discriminators with delays 0, τ and 2τ to obtain $R(0)$, $R(\tau)$, $R(-\tau)$ and $R(2\tau)$.

Then the two constants, a_1 and a_2 , can be found from linear prediction as

$$-R(2\tau) = a_1 R(\tau) + a_2 R(0)$$

$$-R(\tau) = a_1 R(0) + a_2 R(-\tau)$$

Because of the effect noise has on a discriminator with zero delay, this is not proposed.

However, if a different arrangement is used with delays $\tau/2$ and $3\tau/2$, the linear equations for solving a_1 and a_2 can be written as

$$-R(3\tau/2) = a_1 R(\tau/2) + a_2 R(-\tau/2)$$

$$-R(\tau/2) = a_1 R(-\tau/2) + a_2 R(-3\tau/2)$$

In this configuration only two discriminators are required, which saves hardware [5].

5 Simulation of a possible solution

5.1 Prony's Method

Figure 29 shows a block diagram of a four delay line DIFM receiver using Prony's method.

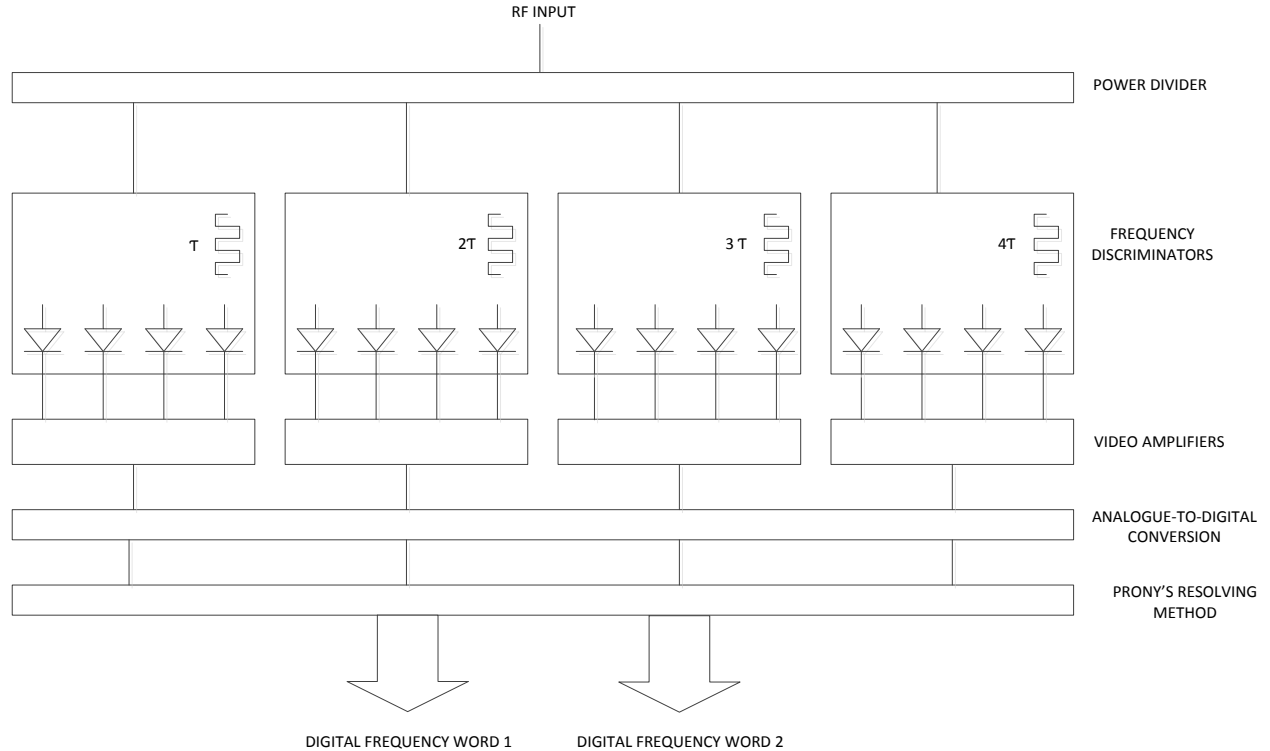


Figure 29: DIFM receiver using Prony's resolving method

In comparing Figure 17 with Figure 29, it can be seen that only the ambiguity-resolving logic block is replaced with Prony's resolving method block and that the delay line ratios are different. For the traditional multiple delay line DIFM receiver the delay line ratio is τ , 2τ , 4τ etc., but for Prony's method the ratios are τ , 2τ , 3τ , 4τ etc.

5.1.1 Four Delay Lines/Two Simultaneous Signals

In the case of two simultaneous signals, one can choose four discriminators with different with lags (or delay time) τ , 2τ , 3τ and 4τ . Refer to Figure 29.

A MATLAB simulation was implemented to simulate a four delay line DIFM receiver using Prony's method.

The simulation was implemented as follows:

- Generate time discrete amplitude samples for the input signal.
- Pass the time discrete signal through a Hilbert filter. This produces a complex time discrete signal.
- Generate four delayed complex time discrete signals with delays τ , 2τ , 3τ and 4τ .

- Multiply the non-delayed complex time discrete signal with the four conjugate delayed complex time discrete signals.
- Pass the results through an LPF (equiripple FIR filter, order = 100, passband frequency = 1 MHz, stopband frequency = 200 MHz) to get rid of the non-DC components. It should be noted that this is a very high order filter, but the reason for this was to achieve a very small passband compared to the sampling frequency of this digital filter.
- Calculate constants a_1 and a_2 .
- Calculate the roots z_1 and z_2 .
- Calculate the angles between the real and imaginary part of the complex roots.
- Convert the angles to frequency values f_1 and f_2 .

See appendix C.1 for the MATLAB source code.

Firstly an input signal consisting of two CW signals with frequencies at 200 MHz and 1000 MHz was simulated. Figure 30 shows the output of the simulation. It is clear that the simulation measured the frequencies of the input signal at 200 MHz and 1000 MHz correctly.

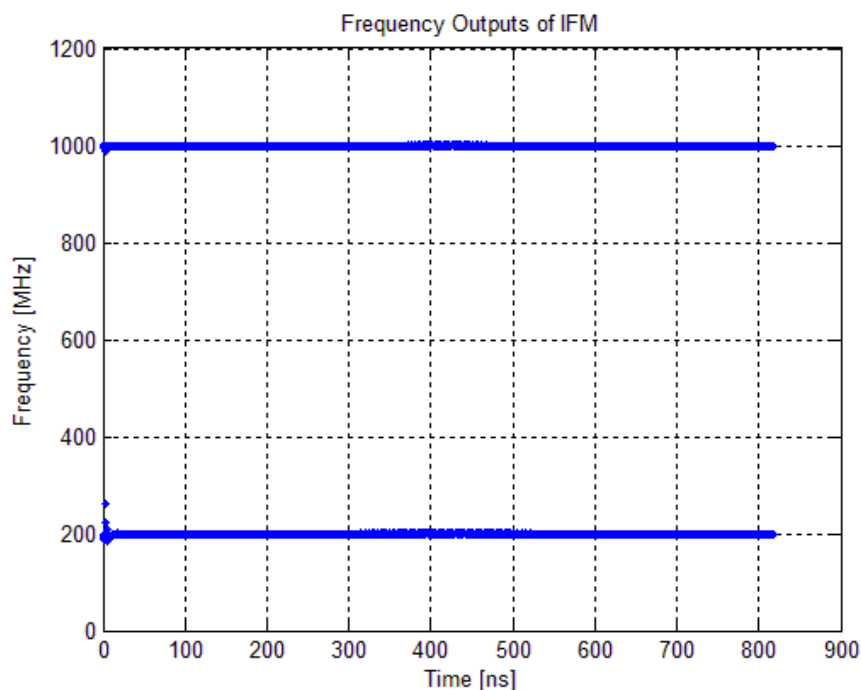


Figure 30: Frequency output = 200 MHz/1000 MHz

Then white noise with a signal-to-noise ratio of 40 dB was added to the input signal. Figure 31 shows the output of the simulation.

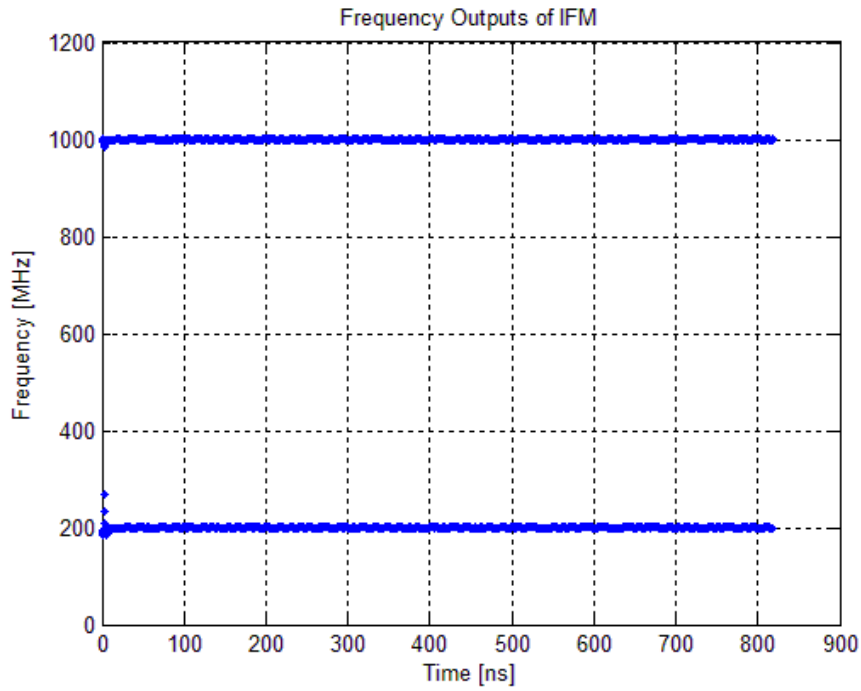


Figure 31: Frequency output = 200 MHz/1000 MHz, SNR = 40 dB

Then an input signal consisting of two CW signals, one at a fixed frequency of 500 MHz and the other sweeping its frequency from 200 MHz to 1000 MHz, was simulated. Figure 32 shows the output of the simulation. It can be seen that the frequencies of the two CW signals were successfully measured, while sweeping the one CW signal from 200 MHz to 1000 MHz.

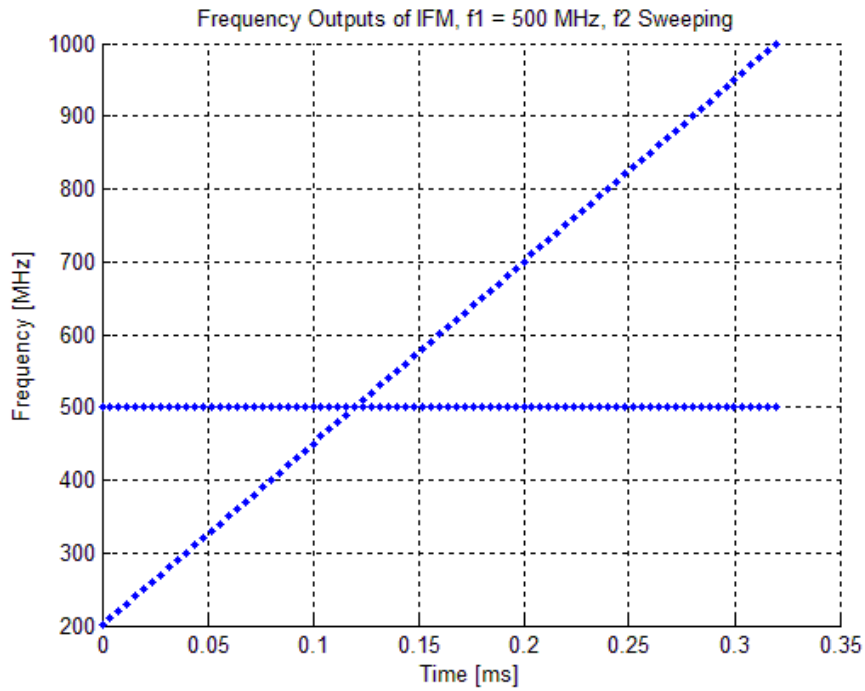


Figure 32: Frequency output = 500 MHz/sweeping

Then an input signal consisting of two CW signals with frequencies at 200 MHz (no phase jumps) and 1000 MHz (which contained two 180° phase jumps) was simulated. In Figure 33 it can be seen that the 180° phase jumps caused jumps in the frequency measurement, but is much better compare to the results achieve with the single line IFM as seen in Figure 28. A reason for this is that the delay line lengths used in prony's method is shorter compared to the delay line length used in the single line IFM simulation.

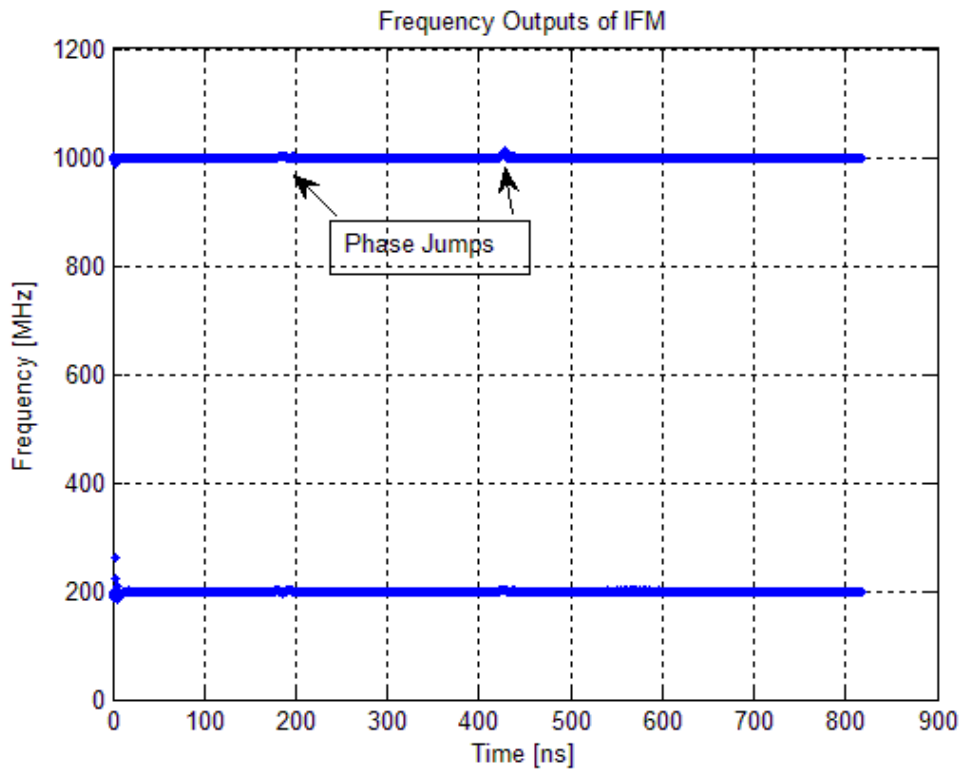


Figure 33: Frequency output = 200 MHz/1000 MHz

5.1.2 Six Delay Lines/Three Simultaneous Signals

Figure 34 shows a block diagram of an six delay line DIFM receiver using Prony's method. In the case of three simultaneous signals, one can choose six discriminators with different lags (or delay time) τ , 2τ , 3τ , 4τ , 5τ and 6τ .

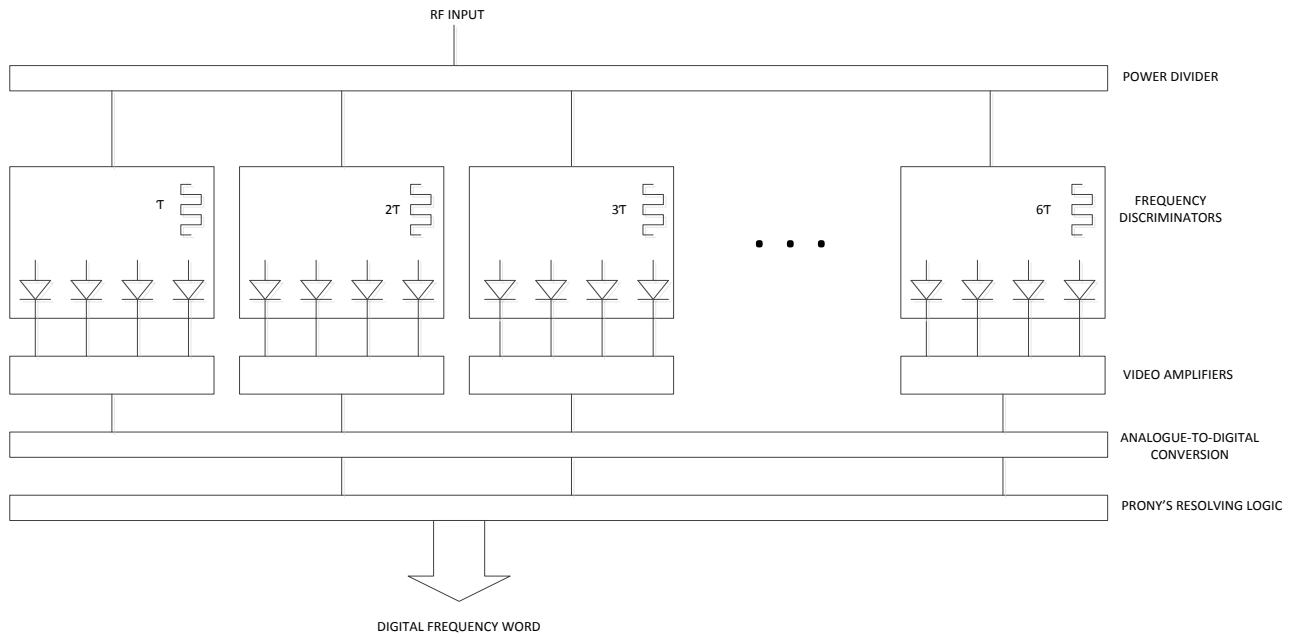


Figure 34: DIFM receiver using Prony's resolving method – six delay lines

A MATLAB simulation was implemented to simulate a six delay line DIFM receiver using Prony's method.

The simulation was implemented as follows:

- Generate time discrete amplitude samples for the input signal.
- Pass the time discrete signal through a Hilbert filter. This produces a complex time discrete signal.
- Generate six delayed complex time discrete signals with delays τ , 2τ , 3τ , 4τ , 5τ and 6τ .
- Multiply the non-delayed complex time discrete signal with the six conjugate delayed complex time discrete signals.
- Pass the results through an LPF (equiripple FIR filter, order = 100, passband frequency = 1 MHz, stopband frequency = 200 MHz) to get rid of the non-DC components.
- Calculate constants a_1 , a_2 and a_3 .
- Calculate the roots z_1 , z_2 and z_3 .
- Calculate the angles between the real and imaginary part of the complex roots.
- Convert the angles to frequency values f_1 , f_2 and f_3 .

See appendix C.2 for the MATLAB source code.

Firstly an input signal consisting of three CW signals with frequencies at 200 MHz, 600 MHz and 1000 MHz was simulated. Figure 35 shows the output of the simulation. It is clear that the simulation measured the frequencies of the input signal at 200 MHz, 600 MHz and 1000 MHz correctly.

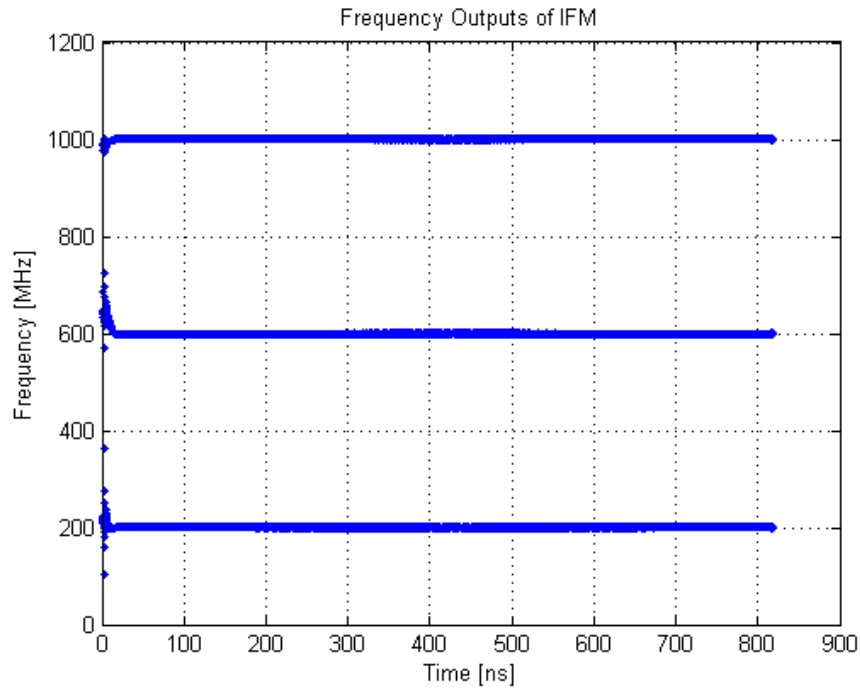


Figure 35: Frequency output = 200 MHz/600 MHz/1000 MHz

Then white noise with a signal-to-noise ratio of 40 dB was added to the input signal. Figure 36 shows the output of the simulation.

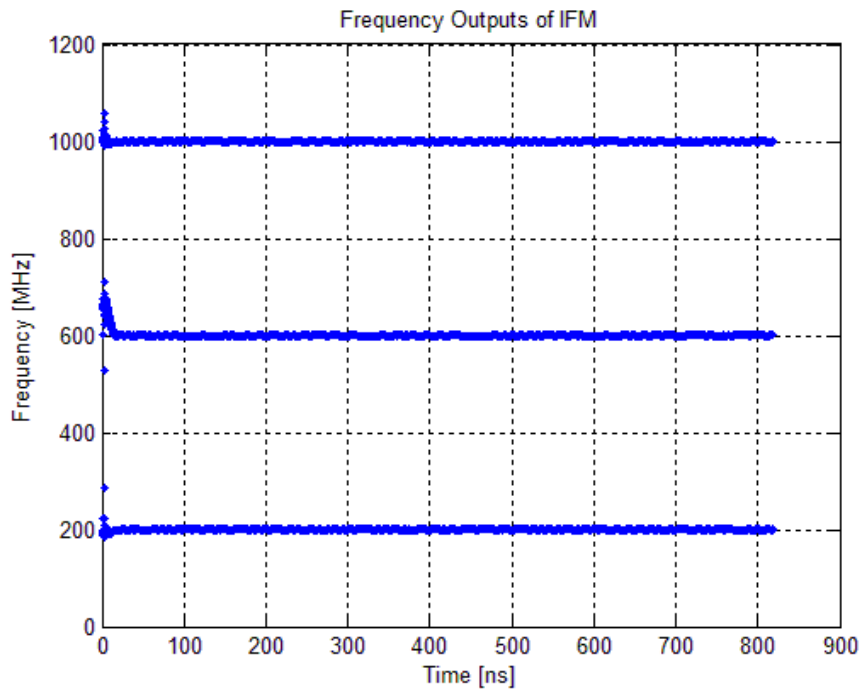


Figure 36: Frequency output = 200 MHz/600 MHz/1000 MHz, SNR = 40 dB

Then an input signal consisting of three CW signals, two at fixed frequencies of 400 MHz and 700 MHz and the other sweeping its frequency from 200 MHz to 1000 MHz, was simulated. Figure 37 shows the output of

the simulation. It can be seen that the frequencies of the three CW signals were successfully measured, while sweeping the one CW signal from 200 MHz to 1000 MHz.

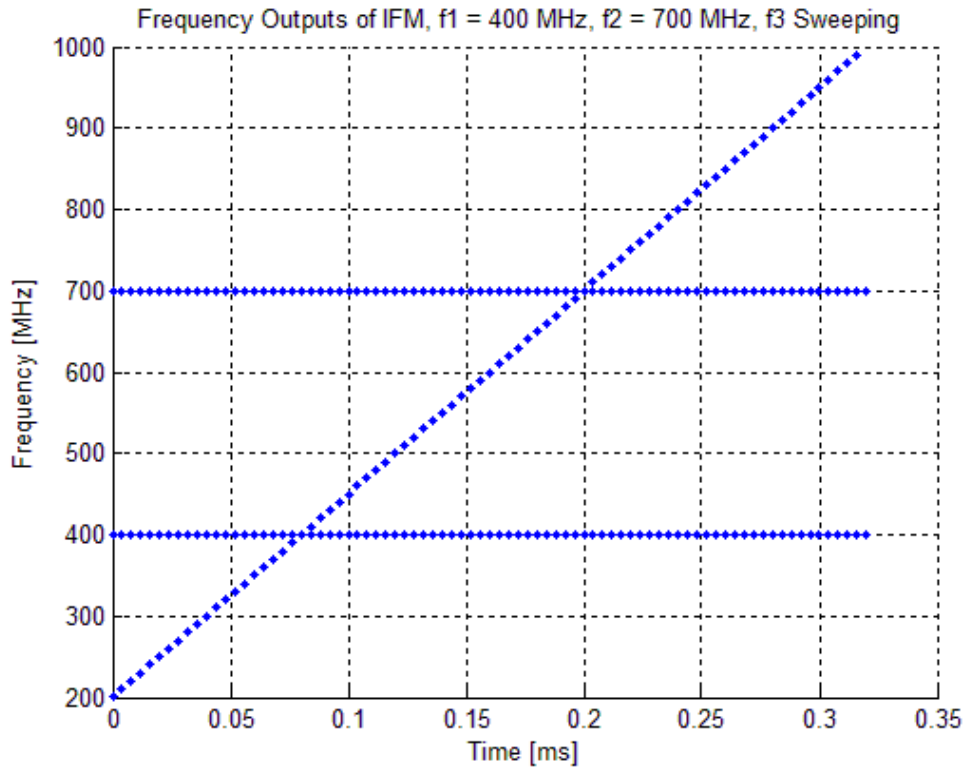


Figure 37: Frequency output = 400 MHz/700 MHz/Sweeping

5.1.3 Eight Delay Lines/Four Simultaneous Signals

Figure 38 shows a block diagram of an eight delay line DIFM receiver using Prony's method. In the case of four simultaneous signals one can choose eight discriminators with different lags (or delay time) τ , 2τ , 3τ , 4τ , 5τ , 6τ , 7τ and 8τ .

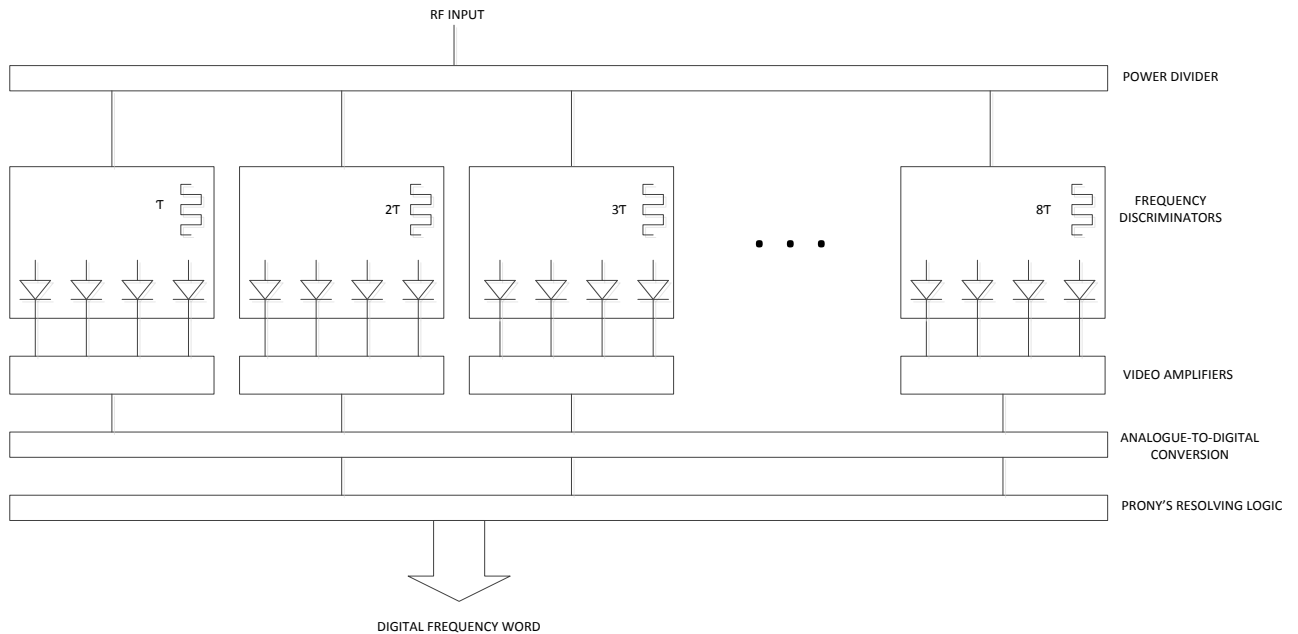


Figure 38: DIFM receiver using Prony's resolving method – eight delay lines

A MATLAB simulation was implemented to simulate an eight delay line DIFM receiver using Prony's method.

The simulation was implemented as follows:

- Generate time discrete amplitude samples for the input signal.
- Pass the time discrete signal through a Hilbert filter. This produces a complex time discrete signal.
- Generate eight delayed complex time discrete signals with delays τ , 2τ , 3τ , 4τ , 5τ , 6τ , 7τ and 8τ .
- Multiply the non-delayed complex time discrete signal with the eight conjugate delayed complex time discrete signals.
- Pass the results through an LPF (equiripple FIR filter, order = 100, passband frequency = 1 MHz, stopband frequency = 200 MHz) to get rid of the non-DC components.
- Calculate constants a_1 , a_2 , a_3 and a_4 .
- Calculate the roots z_1 , z_2 , z_3 and z_4 .
- Calculate the angles between the real and imaginary part of the complex roots.
- Convert the angles to frequency values f_1 , f_2 , f_3 and f_4 .

See appendix C.3 for the MATLAB source code.

Firstly an input signal consisting of four CW signals with frequencies at 200 MHz, 400 MHz, 700 MHz and 1000 MHz was simulated. Figure 39 shows the output of the simulation. It is clear that the simulation measured the frequencies of the input signal at 100 MHz, 400 MHz, 700 MHz and 1000 MHz correctly.

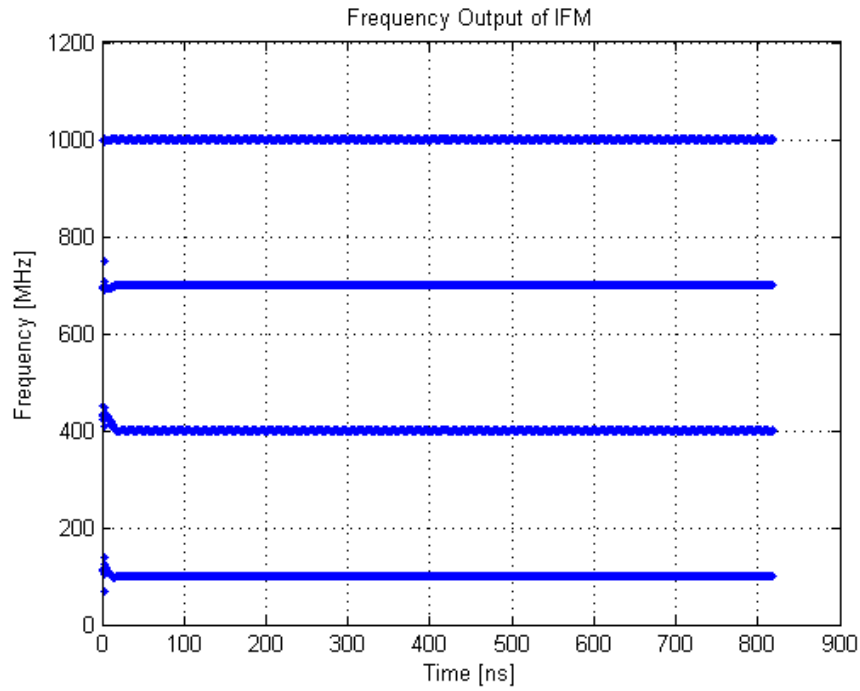


Figure 39: Frequency output = 100 MHz/400 MHz/700MHz/1000 MHz

Then white noise with a signal-to-noise ratio of 40 dB was added to the input signal. Figure 40 shows the output of the simulation.

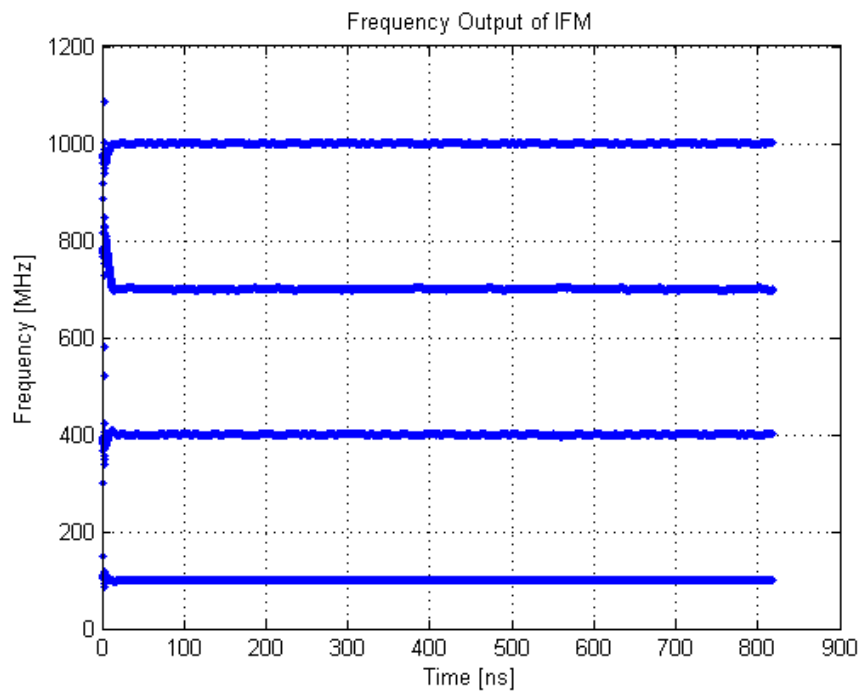


Figure 40: Frequency output = 100 MHz/400 MHz/700MHz/1000 MHz, SNR = 40 dB

Then an input signal consisting of four CW signals, three at fixed frequencies of 400 MHz, 600 MHz and 800 MHz and the other sweeping its frequency from 200 MHz to 1000 MHz, was simulated. Figure 41 shows the

output of the simulation. It can be seen that the frequencies of the four CW signals were successfully measured, while sweeping the one CW signal from 200 MHz to 1000 MHz.

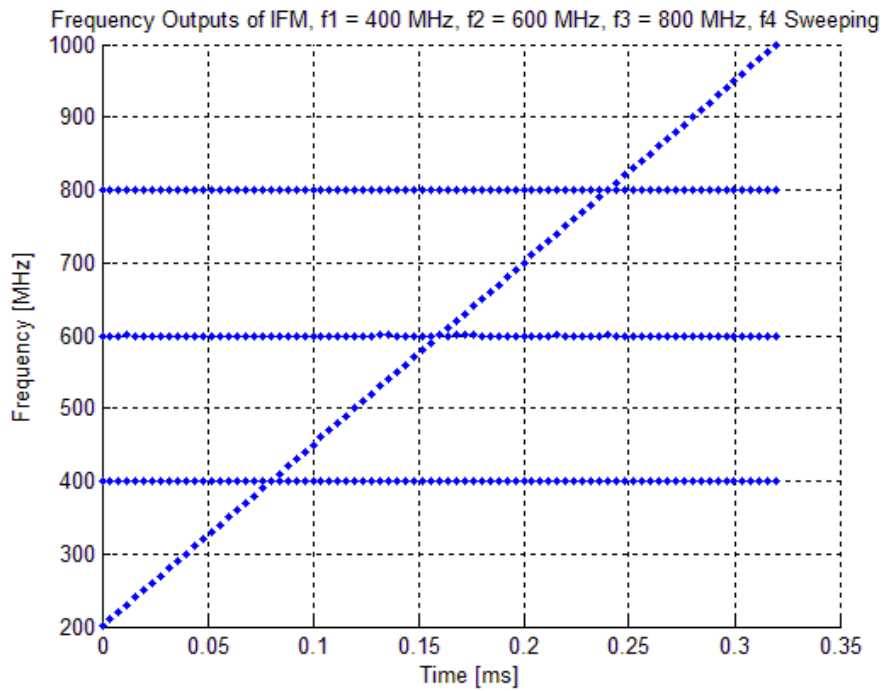


Figure 41: Frequency output = 400 MHz/600 MHz/800MHz/sweeping

5.1.4 Summary of Simulation Results

The results achieved in MATLAB with Prony's method to solve the problem with simultaneous signals seem very promising. This strengthens the argument for continuing to implement Prony's method in hardware and simulate it in hardware.

6 Hardware Implementation and Simulation

6.1 Background on the DRFM Platform

The DRFM was used as the hardware platform to implement a co-pulse IFM receiver.

The Council for Scientific and Industrial Research (CSIR) designed a wideband DRFM module that is discussed in this section. The wideband DRFM is designed as a 6U Versa Module European (VME) form-factor module, as seen in Figure 43. A block diagram of the hardware platform is shown in Figure 42 and the performance specifications in Table 2. Figure 43 shows the fourth and Figure 44 the fifth generation DRFM developed by the CSIR. The fourth generation DRFM has a Xilinx Virtex5 FPGA and 2 Gsps ADC and digital-to-analogue converter (DAC). The fifth generation DRFM has a Xilinx Virtex6 FPGA with a 5 Gsps ADC and DAC.

DRFMs are the main building block of hardware-in-the-loop, radar target and electronic counter-measure simulators.

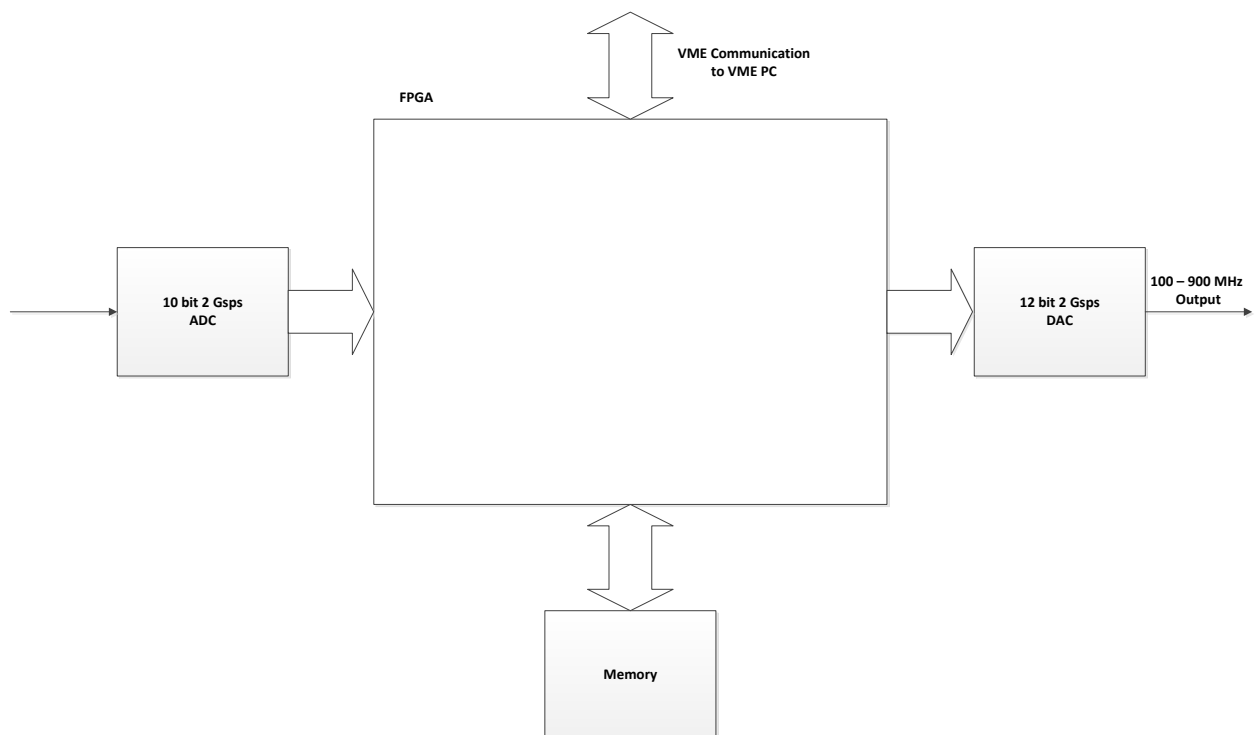


Figure 42: Block diagram of fourth generation DRFM hardware architecture

Table 2: Fourth generation DRFM performance specifications

| Performance Specification | Specification |
|---|----------------------------|
| Instantaneous Bandwidth (IBW) | 800 MHz |
| Input frequency range | 100 to 900 MHz |
| Maximum input power level (before ADC saturation) | -5 dBm |
| Maximum output power level over IBW | Between -7 dBm and -11 dBm |
| Output power flatness over IBW | 3 dB |
| Pulse width | 50 ns to CW |
| Spurious response | -45 dBc over 95% of IBW |
| Operating temperature range | 0 to 35 °C |



Figure 43: Fourth generation DRFM developed by CSIR



Figure 44: Fifth generation DRFM developed by the CSIR

6.2 Implementation Methodology

The fourth generation DRFM was used as the hardware platform. The ADC was used to sample the RF input signal and the main FPGA (Xilinx Virtex5) was used to implement [10],[13] the Hilbert transform, delays and FDs. Only the co-pulse (two simultaneous signals) IFM receiver was implemented.

A combination of System Generator and MATLAB was used to test/verify the design. For the first phase the Hilbert transform and delays only were implemented in System Generator and the FDs and Prony's resolving logic were implemented in MATLAB. This was implemented and tested.

For the second phase the FDs were also implemented in System Generator. Only Prony's resolving logic was implemented in MATLAB. This was implemented and tested.

For the third phase it was attempted to implement Prony's resolving logic in the FPGA. Because of insufficient DSP resources on the Xilinx Virtex5 FPGA and the fact that floating point numbers were not supported in System Generator for the Xilinx Virtex5 FPGA [15], this effort was unsuccessful. Future research can include Prony's resolving logic in the Xilinx Virtex6 main FPGA, which should have sufficient DSP resources (to be investigated) and also supports floating point numbers in System Generator [16].

6.3 High-level block diagram of the hardware implementation

Figure 45 shows the high-level block diagram of the implementation for the co-pulse IFM receiver on the DRFM hardware platform. Also see Appendix D for the top level of the implementation in System Generator.

The following sections will explain the different blocks in more detail.

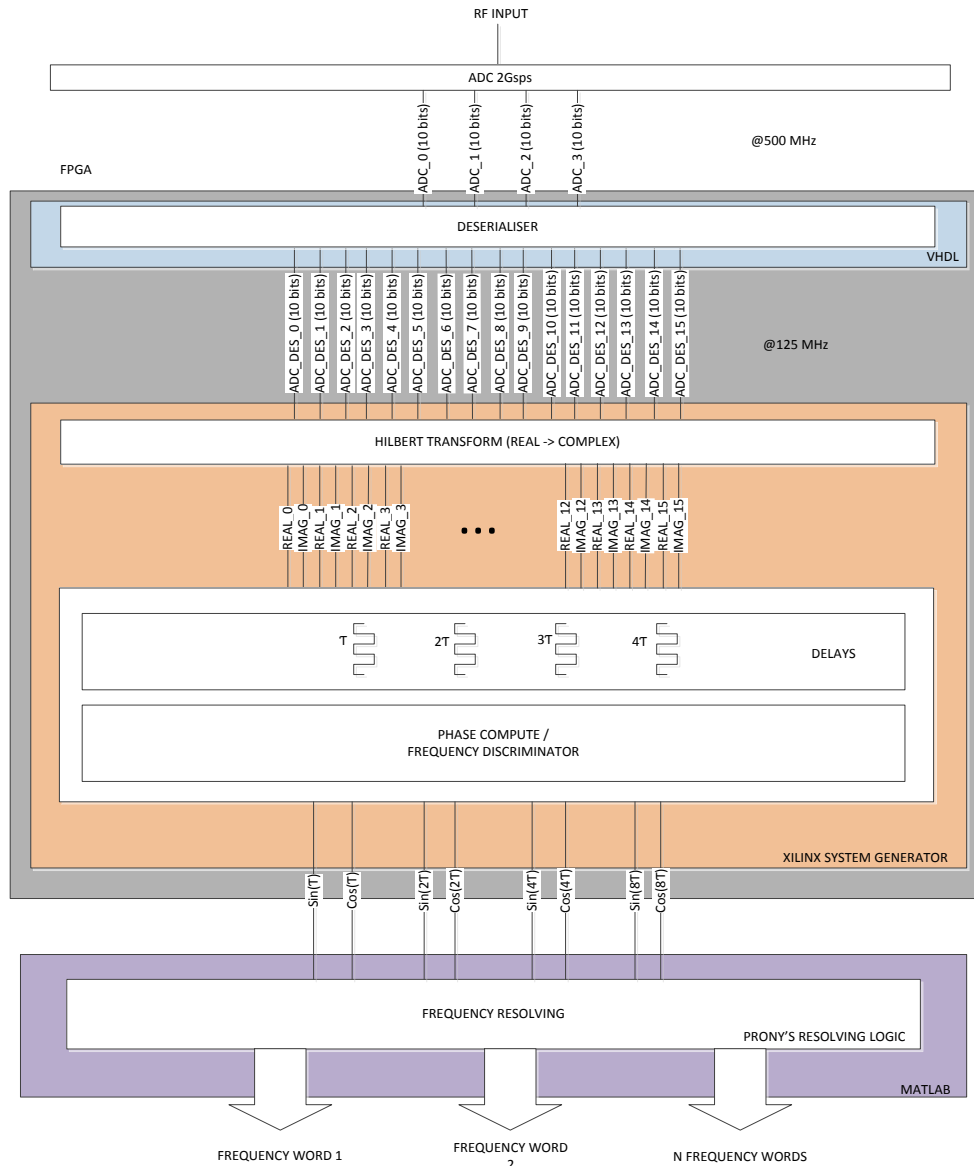


Figure 45: Co-pulse IFM receiver hardware implementation

6.4 Analogue-to-digital Converter

The ADC is a device on the DRFM hardware platform. It is a 2 GspS ADC, which is clocked at 2 GHz and samples the input signal at 2 GspS. The ADC gives 4 x 10 bit samples @ 500 MHz. These 4 x 10 bit samples are connected to the FPGA via high-speed low-voltage differential signalling (LVDS) lines.

6.5 Deserialiser

As mentioned in the previous section, the 4 x 10 bit samples @ 500 MHz are connected to the FPGA. For a FPGA this data rate is very high. Deserialisers must be used to slow down the data without throwing away bits. Deserialisers increase the data bus size, resulting in a reduced data rate. Deserialisers are used to transfer the 4 x 10 bit samples (40 bit bus in total) @ 500 MHz to 16 x 10 bit (160 bit bus in total) @ 125 MHz. A data rate of 125 MHz is much more acceptable for use inside an FPGA.

The Input Serialiser/Deserialiser (ISERDES) components were use in the FPGA to do the deserialisation. Also see Figure 46.

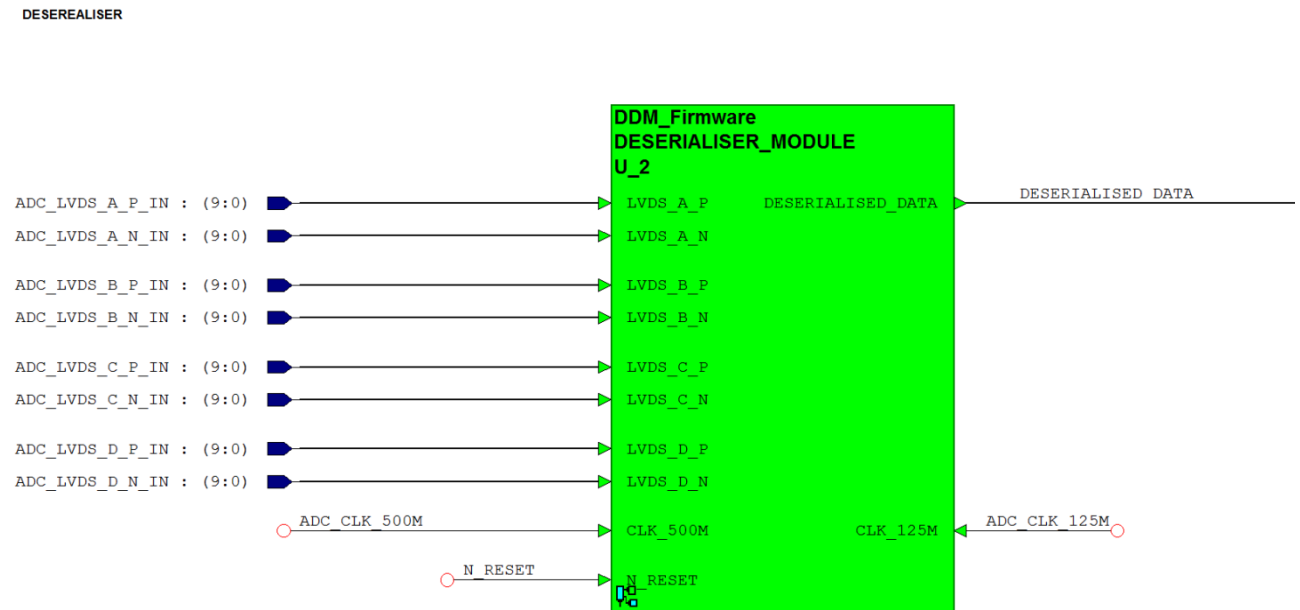


Figure 46: Deserialiser

6.6 Hilbert Transform

To be able to use Prony’s frequency resolving technique, the real amplitude samples must be transformed into complex samples (I and Q samples). This is done by generating a Hilbert transform in System Generator. MATLAB’s FDA (Filter Design and Analysis) tool was used to generate the parameters (coefficients) for the Hilbert transform/filter and implemented in System Generator.

For more details on Hilbert transforms, refer to reference [23].

6.7 Delays

Because the data is clocked, it is very easy to achieve delays in the digital domain. Figure 47 shows a one-clock delay. The data (sample) of one clock later are used for the delayed data.

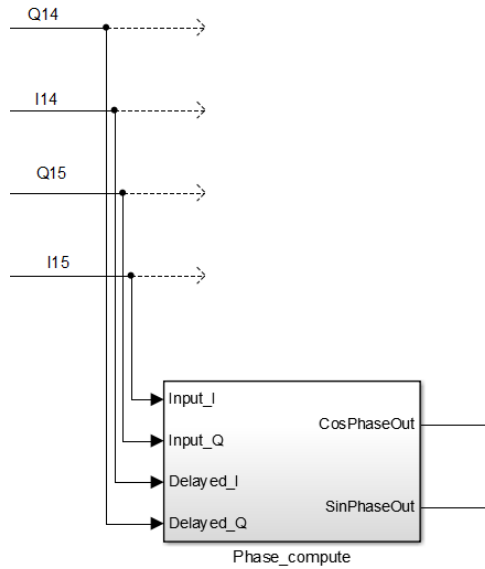


Figure 47: One sample (τ) delay

A two-clock delay and three clock delay can be achieved in a similar way.

6.8 Phase Compute/Frequency Discriminators

Figure 48 shows the Phase Compute (FD) used in System Generator. This block was implemented as shown in Figure 49.

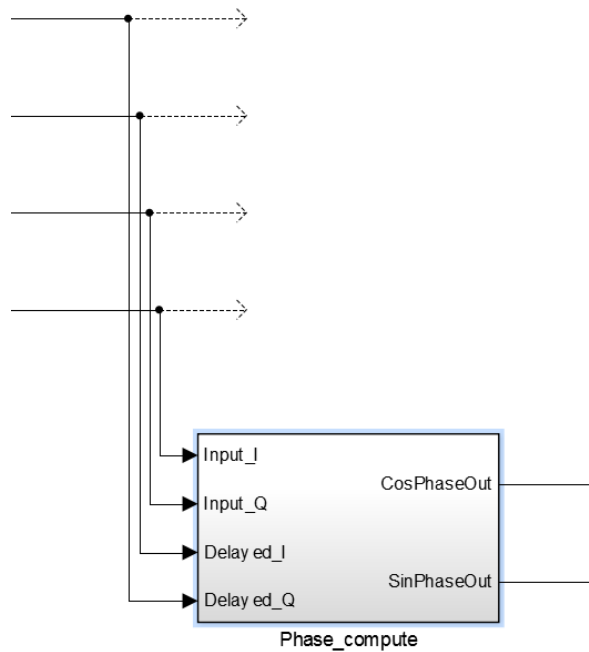


Figure 48: Phase compute/frequency discriminator

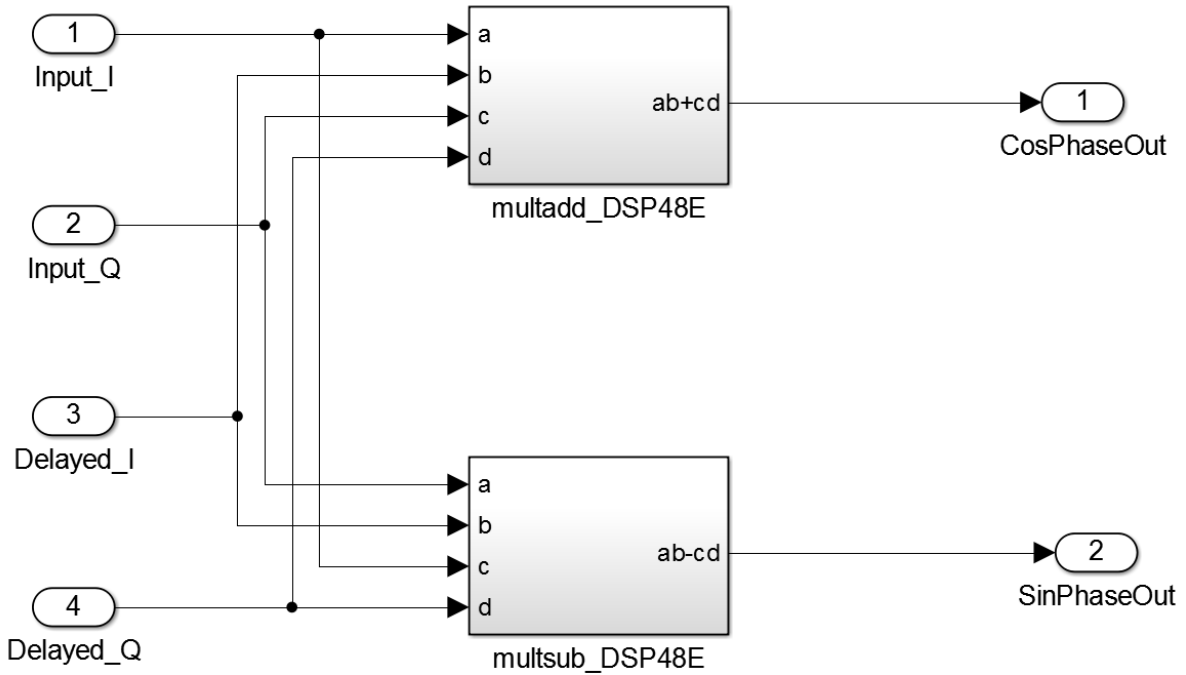


Figure 49: Frequency discriminator implementation

As explained in section 2.3.2, the function of the FD is to accept an RF input signal and produce two video signals I_{out} and Q_{out} . Take a complex input signal $Ae^{j2\pi fn}$, where A is the amplitude, f the frequency and n the sample number. If this input signal is multiplied with a conjugate delayed version.

$$\begin{aligned}
 & Ae^{j2\pi fn} \times \text{conj}(Ae^{j2\pi f(n-1)}) \\
 &= Ae^{j2\pi fn} \times \text{conj}(Ae^{j2\pi fn} \times e^{-j2\pi f}) \\
 &= Ae^{j2\pi fn} \times Ae^{-j2\pi fn} \times e^{j2\pi f} \\
 &= A^2 e^{j2\pi f} \\
 &= I_{out} + jQ_{out}
 \end{aligned} \tag{41}$$

If one takes equation (41) and rewrites the inputs as I and Q values, one sees the following:

$$\begin{aligned}
 & Ae^{j2\pi fn} \times \text{conj}(Ae^{j2\pi f(n-1)}) \\
 &= (I + jQ) \times \text{conj}(I_{delayed} + jQ_{delayed}) \\
 &= (I + jQ) \times (I_{delayed} - jQ_{delayed}) \\
 &= I \cdot I_{delayed} - jI \cdot Q_{delayed} + jI_{delayed} \cdot Q + Q \cdot Q_{delayed} \\
 &= (I \cdot I_{delayed} + Q \cdot Q_{delayed}) + j(I_{delayed} \cdot Q - I \cdot Q_{delayed}) \\
 &= I_{out} + jQ_{out} \\
 &= \text{CosPhaseOut} + j\text{SinPhaseOut}
 \end{aligned}$$

This explains the implementation as shown in Figure 49. This same method was also used in the MATLAB simulation code to calculate the outputs from the FD.

6.9 Frequency Resolving

6.9.1 Outputs from the Frequency Discriminators

6.9.1.1 Co-pulse IFM receiver (two simultaneous signals)

The outputs from the four FDs with delays τ , 2τ , 3τ and 4τ will be:

$$R(\tau) = P_1 \exp(jw_1\tau) + P_2 \exp(jw_2\tau)$$

$$R(2\tau) = P_1 \exp(j2w_1\tau) + P_2 \exp(j2w_2\tau)$$

$$R(3\tau) = P_1 \exp(j3w_1\tau) + P_2 \exp(j3w_2\tau)$$

$$R(4\tau) = P_1 \exp(j4w_1\tau) + P_2 \exp(j4w_2\tau)$$

Note that each of these outputs has a real and imaginary component (complex number). Note that these are not constants and change from sample to sample.

6.9.1.2 Three and more simultaneous signals

For three and more simultaneous signals the required outputs from the FDs (for the preselected delays) can be obtained as explained in section 6.9.1.1.

6.9.2 Constants $a_1 \dots a_N$

6.9.2.1 Co-pulse IFM receiver (two simultaneous signals)

As explained in section 4.1.2, a_1 and a_2 can be solved with the following equation:

$$a_1 = \frac{\begin{vmatrix} -R(4\tau) & R(2\tau) \\ -R(3\tau) & R(\tau) \end{vmatrix}}{\begin{vmatrix} R(3\tau) & R(2\tau) \\ R(2\tau) & R(\tau) \end{vmatrix}}$$

$$a_2 = \frac{\begin{vmatrix} R(3\tau) & -R(4\tau) \\ R(2\tau) & -R(3\tau) \end{vmatrix}}{\begin{vmatrix} R(3\tau) & R(2\tau) \\ R(2\tau) & R(\tau) \end{vmatrix}}$$

It is known that to calculate the determinant of a 2x2 matrix [11],[12].

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

However, it is known that a , b , c and d are complex numbers. Therefore

$$\begin{aligned} \begin{vmatrix} a_r + ja_i & b_r + jb_i \\ c_r + jc_i & d_r + jd_i \end{vmatrix} &= (a_r + ja_i)(d_r + jd_i) - (b_r + jb_i)(c_r + jc_i) \\ &= (a_r d_r - a_i d_i - b_r c_r + b_i c_i) + j(a_r d_i + a_i d_r - b_r c_i - b_i c_r) \end{aligned}$$

Furthermore, the division of two complex numbers is defined as follows:

$$\frac{a + bi}{c + di} = \left(\frac{ac + bd}{c^2 + d^2} \right) + \left(\frac{bc - ad}{c^2 + d^2} \right) i$$

This shows that if one takes the real and imaginary outputs from the four FDs, one can compute a_1 and a_2 with multipliers, adders and dividers (four dividers) in firmware. Also note that multipliers and adders add very little latency. What is also important to note is that one can parallelise many of the multiplications. In firmware a divide adds great latency, but a lookup table can be used for the divide. By using a lookup table for the divide, the latency is reduced significantly.

Latency of calculations is very important because the frequency measurement is required as quickly as possible.

6.9.2.2 Three and more simultaneous signals

For three and more simultaneous signals the constants $a_1 \dots a_N$ can be solved in a similar method as explained in section 6.9.1.2.

6.9.3 Determination of the Roots

Once the constants have been obtained as explained in section 6.9.2, the roots can be calculated. The calculation of the roots is a major concern for implementing Prony's resolving method in firmware.

6.9.3.1 Co-pulse IFM receiver (two simultaneous signals)

The calculation of the roots for two simultaneous signals is relatively easy.

$$ax^2 + bx + c = 0, \text{ where } a = 1, b = a_1 \text{ and } c = a_2$$

The roots can be calculated with the following formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The square root of a complex number $(a + jb)$ is $\pm(\gamma + \delta j)$, where

$$\gamma = \sqrt{\frac{a + \sqrt{a^2 + b^2}}{2}}$$

and

$$\delta = \text{sgn}(b) \sqrt{\frac{-a + \sqrt{a^2 + b^2}}{2}}$$

where

$$\text{sgn}(b) := \begin{cases} -1 & \text{if } b < 0, \\ 0 & \text{if } b = 0, \\ 1 & \text{if } b > 0 \end{cases}$$

In firmware a divide adds great latency, but if a lookup table is used for the divide, the latency can be reduced significantly.

6.9.3.2 Three simultaneous signals

The calculation for three simultaneous signals is much more intensive than for two simultaneous signals.

$$ax^3 + bx^2 + cx + d = 0, \text{ where } a = 1, b = a_1, c = a_2 \text{ and } d = a_3$$

The roots can be calculated with the formulas shown in Appendix F.

The latency for calculating the roots for three simultaneous signals will also be much more than for two simultaneous signals.

6.9.3.3 Four simultaneous signals

The calculation for four simultaneous signals is much more intensive than for three simultaneous signals.

$$ax^4 + bx^3 + cx^2 + dx + e = 0, \text{ where } a = 1, b = a_1, c = a_2, d = a_3 \text{ and } e = a_4$$

Refer to reference [11] and [12] for more detail on how to calculate the roots for a fourth-order polynomial.

The latency for calculating the roots for four simultaneous signals will be much more than for three simultaneous signals.

6.9.3.4 More than four simultaneous signals

The closed-form solutions can be found only up to the fourth order. Equations with a higher order can be solved, but through numerical methods [5]. This will require a great deal of computations/latency and for more than four simultaneous signals the FFT algorithm (as explained in 2.1.7) is recommended to resolve the frequencies.

6.9.3.5 Other ideas for calculating the roots

The calculation of the roots in firmware for two simultaneous signals is relatively easy. For more than two a soft-core processor, microprocessor or digital signal processor can be used to calculate the roots. Division is done much more easily in software than in firmware. Obviously this will cause the latency to increase drastically and it is not recommended.

A lookup table is also an option to determine the roots. Because of the large size of such lookup tables, it is also not recommended.

6.9.4 Determination of the Frequency Values

Once the roots have been obtained as explained in section 6.9.3, a lookup table is recommended to determine the frequencies. This lookup table can be pre-generated in MATLAB and then used in firmware.

Putting the steps required in sections 6.9.3 and 6.9.4 into perspective, the following method is recommended to determine the frequencies. The constants ($a_1..a_N$) are constant (not changing between samples) complex values for fixed input frequencies. It is rather recommended to determine the frequencies directly from the constants ($a_1..a_N$) by using a lookup table. The lookup table can be generated in MATLAB and used in the firmware to determine frequencies from the constants. If a lookup table is used, it can also accommodate more than four simultaneous signals, because the high latency calculations will be done in advance in MATLAB.

Another method can simply be to store the constants or roots as the frequency words. The frequency word can then be converted by user software or the jammer hardware to the actual frequencies in hertz. This is not ideal; it is rather recommended that the receiver outputs frequency values with no post-calculations required.

6.10 Hardware Simulation Results

Only the co-pulse IFM receiver was implemented and simulated. It was implemented as shown in Figure 45. An input signal consisting of two CW signals, one at a fixed frequency of 200 MHz and the other sweeping its frequency from 300 MHz to 900 MHz and with the power levels the same, was simulated. Figure 50 shows the output of the simulation.

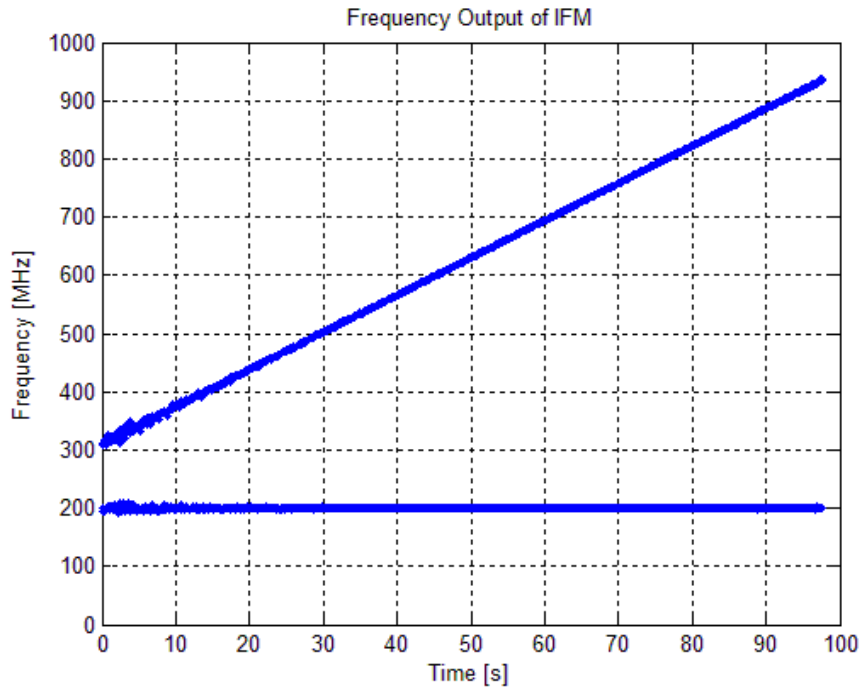


Figure 50: Hardware simulation of the co-pulse IFM receiver

Then an input signal consisting of two CW signals, one at a fixed frequency of 500 MHz and the other sweeping its frequency from 200 MHz to 900 MHz and with the power levels the same, was simulated. Figure 51 shows the output of the simulation

Note the measurement errors when the frequencies come close to each other. This is due to the LPF, which must filter out the difference frequency, which are not included in the simulation.

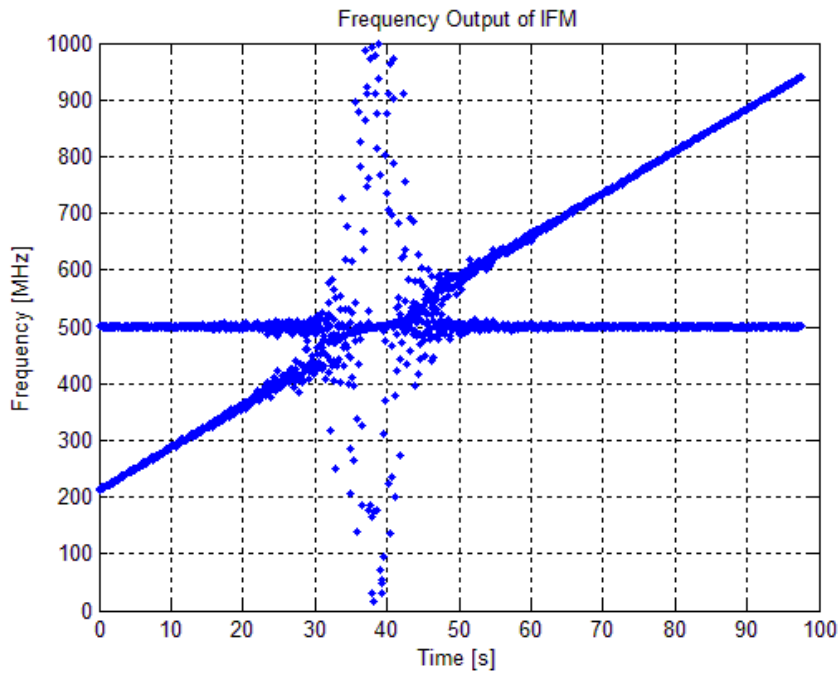


Figure 51: Hardware simulation of the co-pulse IFM receiver (frequency crossing)

Figure 52 shows the output of the simulation when an LPF (equiripple FIR filter, order = 50, Passband Frequency = 0.01 MHz, Stopband Frequency = 3 MHz) is used after the FDs.

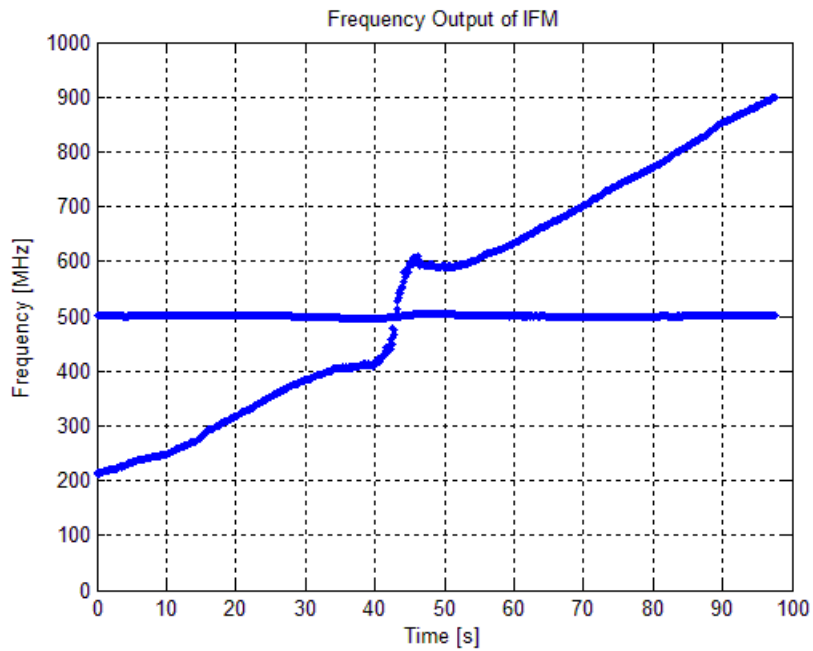


Figure 52: Hardware simulation of the co-pulse IFM receiver (frequency crossing with LPF)

Figure 53 shows the output of the simulation without an LPF after the FDs, but with the

$$\det(R) = \begin{vmatrix} R(3\tau) & R(2\tau) \\ R(2\tau) & R(\tau) \end{vmatrix} \text{ greater than a certain minimum value check.}$$

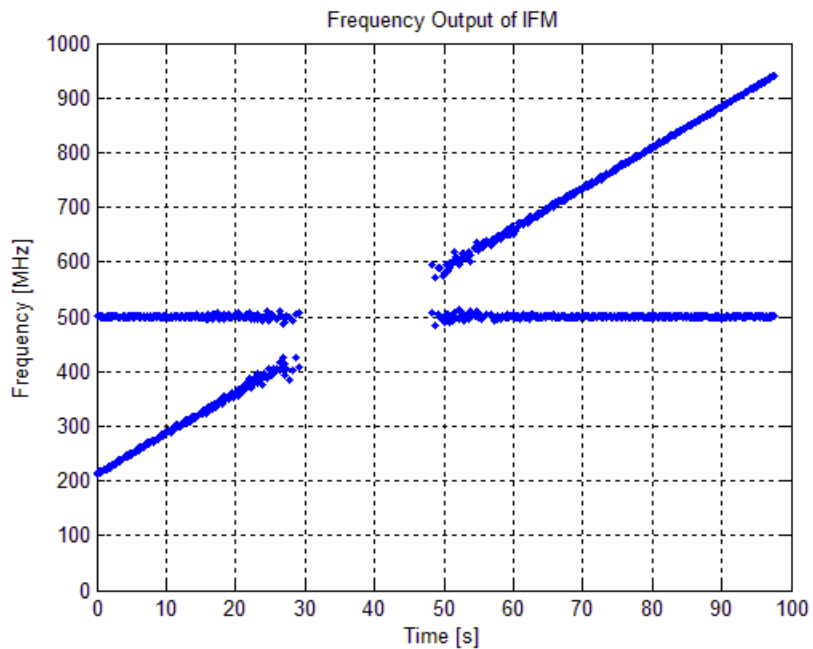


Figure 53: Hardware simulation of the co-pulse IFM receiver (frequency crossing with determinant check)

Figure 54 shows the output of the simulation with an LPF (equiripple FIR filter, order = 50, Passband Frequency = 0.01 MHz, Stopband Frequency = 3 MHz) after the FDs and with the

$\det(R) = \begin{vmatrix} R(3\tau) & R(2\tau) \\ R(2\tau) & R(\tau) \end{vmatrix}$ greater than a certain minimum value check. When the determinant check is smaller than the minimum value, the traditional IFM was used to measure the frequency.

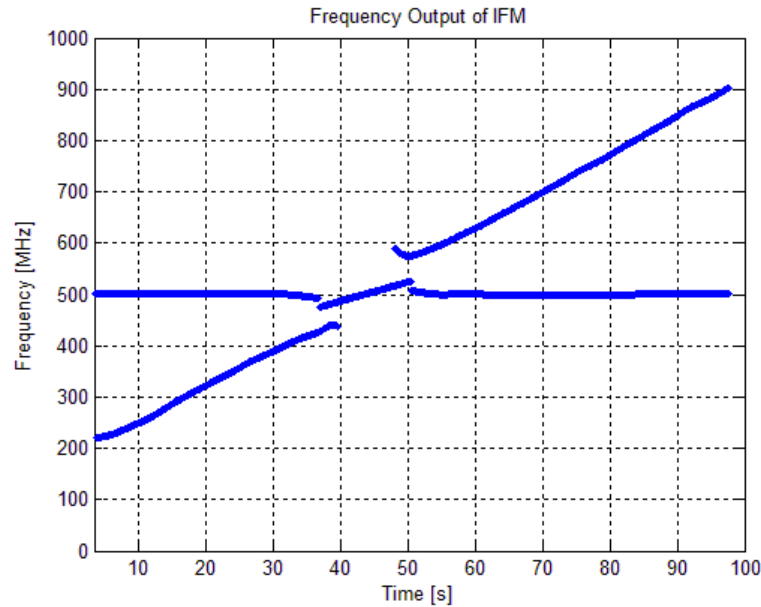


Figure 54: Hardware simulation of the co-pulse IFM receiver (LPF, determinant check with traditional IFM to resolve frequency crossing)

Then an input signal consisting of two CW signals, one at a fixed frequency of 200 MHz and the other sweeping its frequency from 300 MHz to 900 MHz and with a 30 dB power difference was simulated. Figure 55 shows the output of the simulation. No LPF or determinant check was included in this simulation.

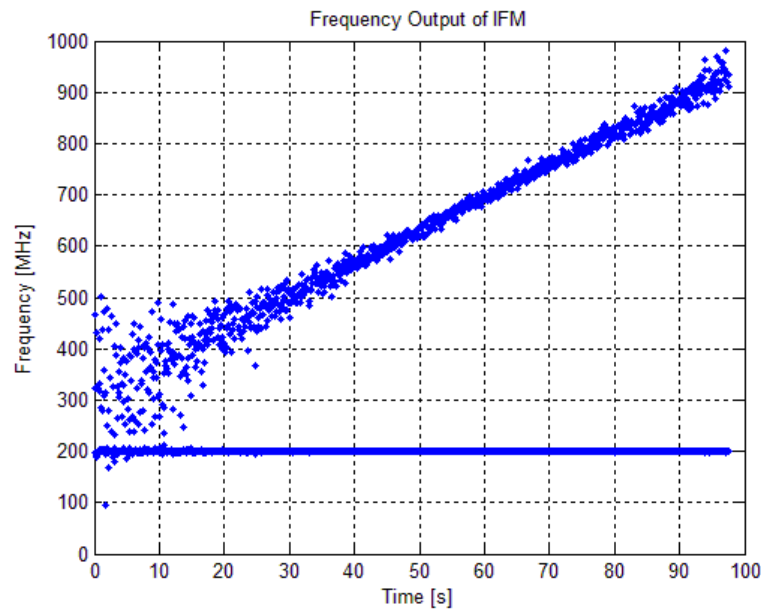


Figure 55: Hardware simulation of the co-pulse IFM receiver (30 dB power difference)

Figure 56 shows the output of the simulation with the $\det(R) = \begin{vmatrix} R(3\tau) & R(2\tau) \\ R(2\tau) & R(\tau) \end{vmatrix}$ greater than a certain minimum value check. When the determinant check is smaller than the minimum value, the traditional IFM was used to measure the frequency.

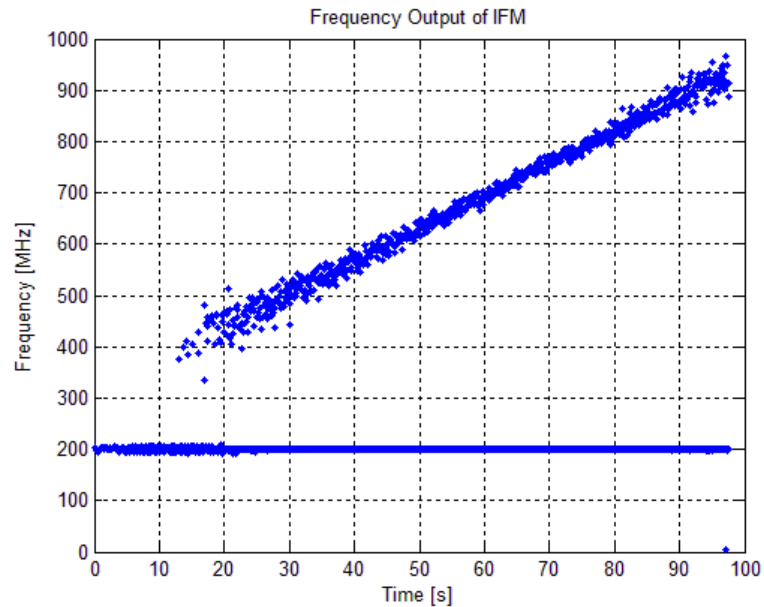


Figure 56: Hardware simulation of the co-pulse IFM receiver (30 dB power difference with determinant check)

To summarise, it is unfortunate that the whole design could not be implemented in the FPGA because of insufficient DSP resources on the Xilinx Virtex5 FPGA and because floating point numbers were not supported in System Generator for the Xilinx Virtex5 FPGA.

For a co-pulse IFM receiver it is recommend first to check for two simultaneous signals with the determinant check. If true, then resolve the two frequencies with Prony's method. If two signals are not present, use the traditional IFM resolving method to determine the frequency.

Nevertheless, the hardware simulation results achieved with the co-pulse IFM receiver implementation shown in Figure 45 are very promising.

7 Hardware Results achieved

7.1 Hardware Test Setup

The hardware implementation shown in Figure 45 was then added in an FPGA with the help of HDL Author software and Xilinx ISE Design Suite software. Appendix E shows the top level of the implementation in HDL Author and Appendix F shows the required MATLAB post processing. The implementation was synthesised, placed and routed, programmed and tested. A block diagram of the hardware test setup is shown in Figure 57.

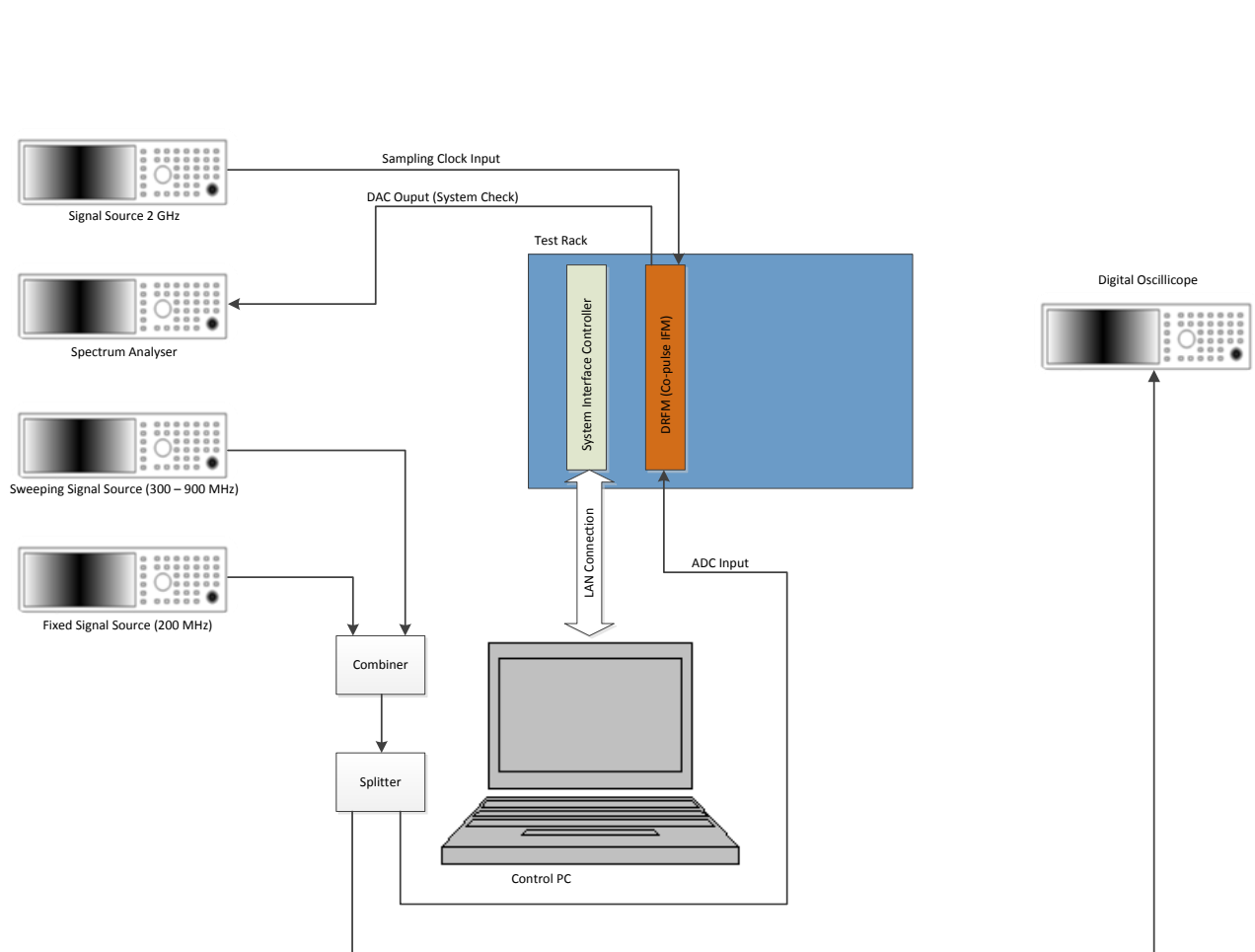


Figure 57: Block diagram of the test setup

Figure 58 shows the hardware test setup used in the laboratory.

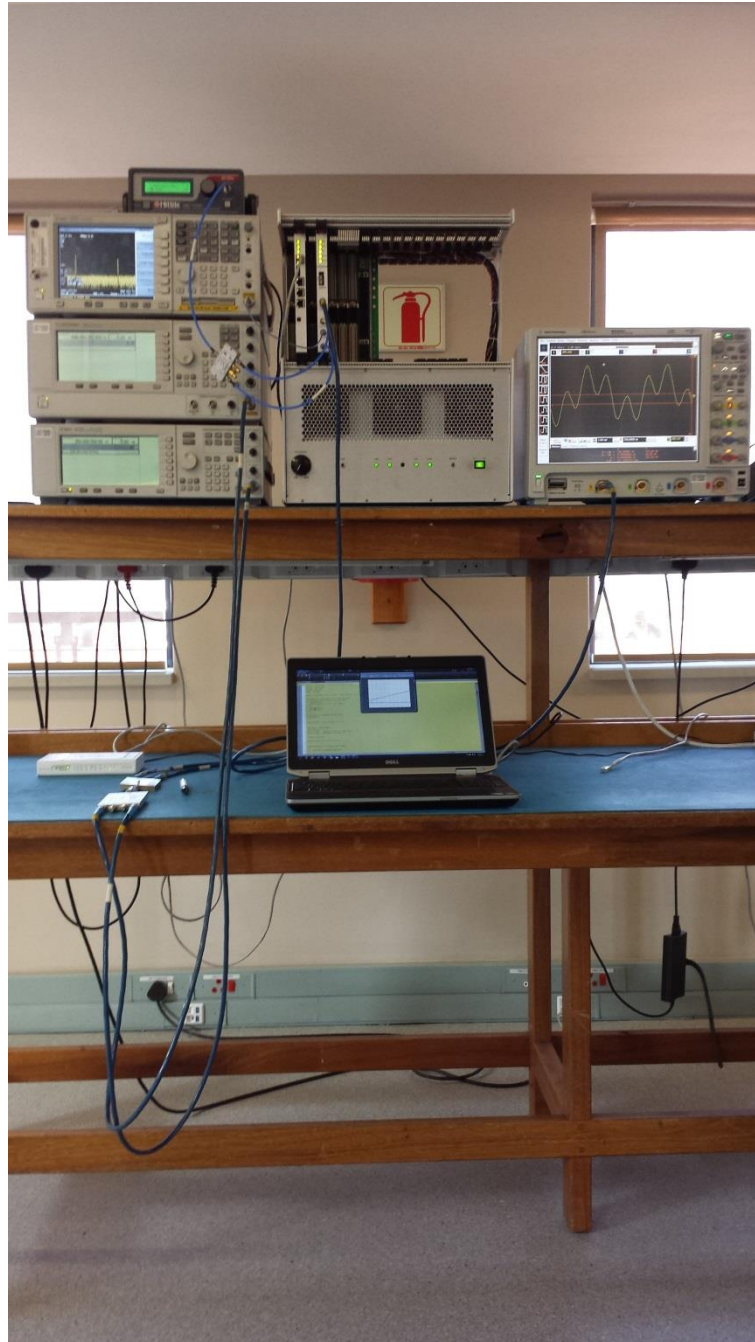


Figure 58: Hardware test setup

Figure 59 shows the DRFM module in the VME test rack. The firmware (with System Generator implementation) was implemented on the DRFM module.

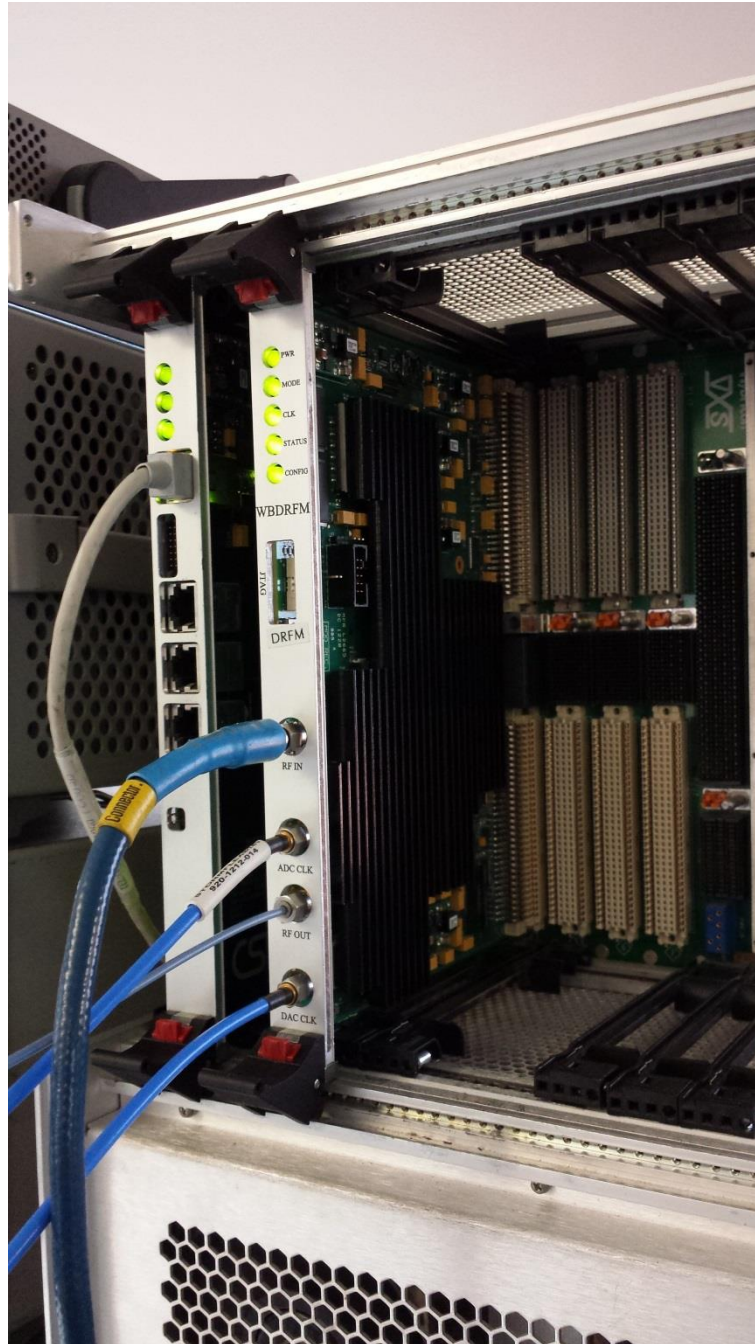


Figure 59: DRFM in the hardware test setup

Figure 60 shows the two signal generators that were used to generate the two signals. On the bench the combiner can be seen, which was used to combine the two signals.

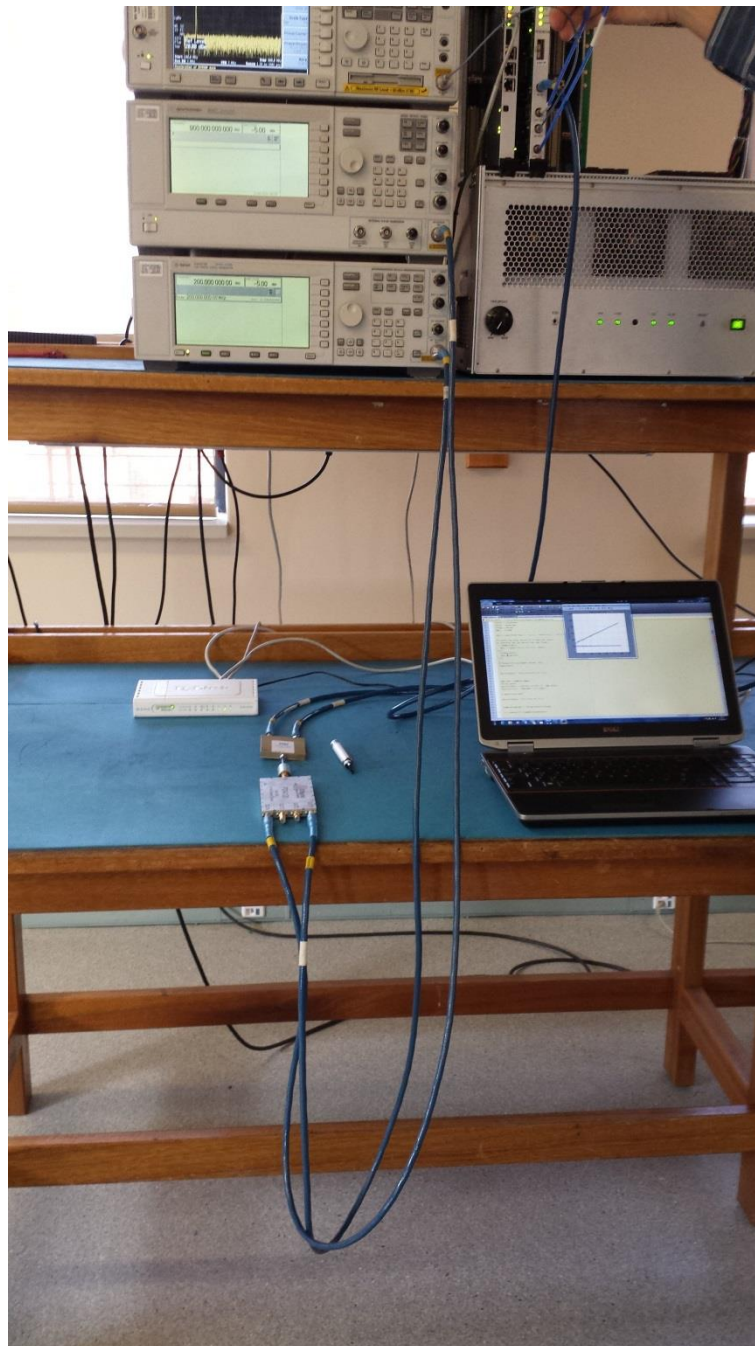


Figure 60: Combined signal generators

7.2 Hardware Implementation Results

An input signal consisting of two CW signals, one at a fixed frequency of 200 MHz and the other sweeping its frequency from 300 MHz to 900 MHz and with the same power levels, was tested. Figure 61 shows the output of the hardware implementation. This can be compared with the hardware simulated results in Figure 50.

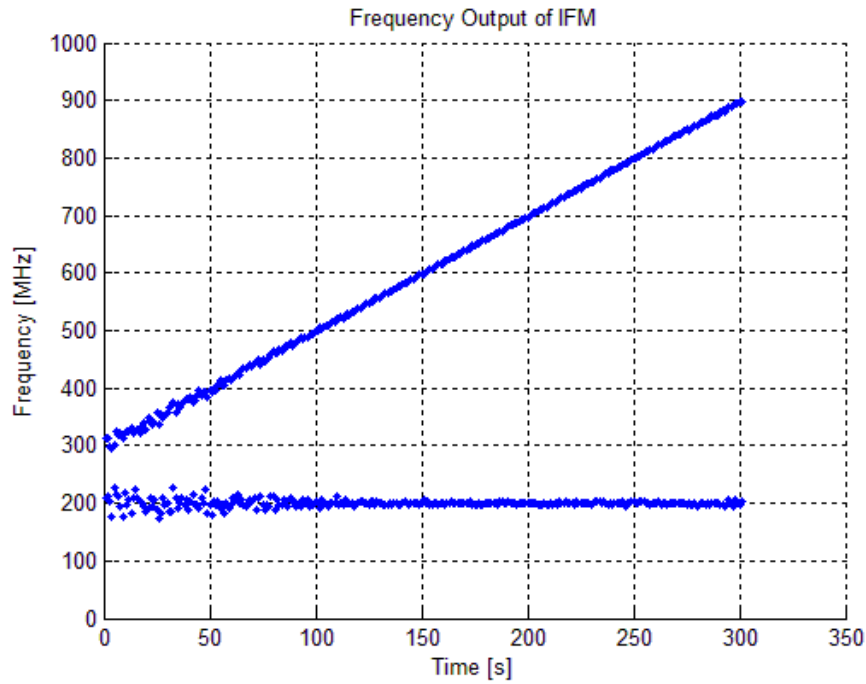


Figure 61: Hardware result of the co-pulse IFM receiver

Then an input signal consisting of two CW signals, one at a fixed frequency of 500 MHz and the other sweeping its frequency from 200 MHz to 900 MHz and with the power levels the same, was tested in hardware. Figure 62 shows the output of the simulation. This can be compared with the hardware simulated results in Figure 51.

Again note the result when the frequencies come close to each other. This is due to the LPF (after the FDs) that is not included in the hardware implementation.

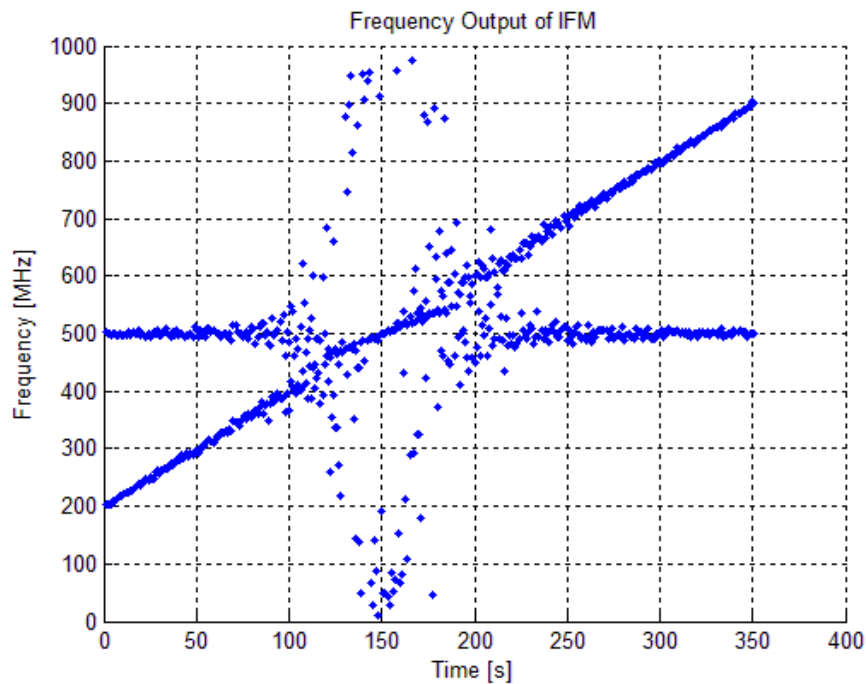


Figure 62: Hardware result of the co-pulse IFM receiver (frequency crossing)

Lastly, an input signal consisting of two CW signals, one at a fixed frequency of 200 MHz and the other sweeping its frequency from 300 MHz to 900 MHz but with a 30 dB power difference was tested. Figure 63 shows the output of the hardware implementation.

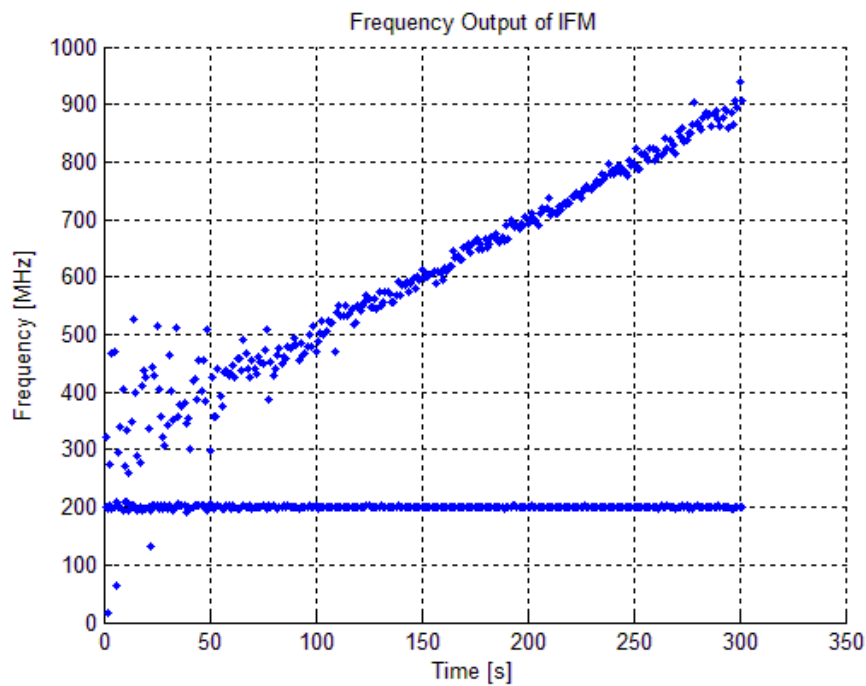


Figure 63: Hardware result of the co-pulse IFM receiver (30 dB power difference)

These tests were repeated a number of times with similar results. The results compared very well with the hardware simulation results.

8 Conclusion and suggestions for future work

During the very first exposure to IFM receivers, which involved taking the outputs from the discriminators to a frequency value, it was discovered that the phase compare is a very unique and challenging method of measuring frequency. The combination of IBW, frequency resolution/accuracy and throughput time (latency) of a well-developed IFM receiver is much better compared to other known acquisition receivers.

The simultaneous signal problem was discovered only later, and immediately after this discovery, the need was identified to investigate the problem to see if a solution could be found without giving away too much on throughput time.

Firstly, many different acquisition receivers were researched to make sure that the IFM receiver was still the best option, taking IBW, frequency resolution/accuracy and throughput time (latency) into consideration. The IFM receiver still proved to be best according to research on acquisition receivers. Then the traditional IFM receiver was researched.

During the literature study research, literature was discovered that proposed Prony's method as a solution to solve the issue of simultaneous signals. Literature submitted from as early as 1989 was found that proposed Prony's method. Prony's method also used the FDs used in traditional IFM receivers. However, no evidence of the use of this method could be found in current commercial IFM receivers, indicating that the implementation of Prony's method is the difficult part.

Then it was decided to simulate the traditional IFM receiver in MATLAB, first with only a single delay line and then with multiple delay lines.

After successful simulation on the traditional IFM receiver, Prony's method was simulated in MATLAB. This made use of the same line delay and FD "code" as in the traditional IFM receiver simulation. The differences were in the number and choice of delay lines and in the frequency word resolving part, which was replaced with Prony's resolving logic.

After successful MATLAB simulation of Prony's resolving method, it was then decided to start an investigation into implementing Prony's resolving logic in an FPGA. The CSIR already had a traditional IFM receiver implemented in FPGA. This was used as the baseline in an attempt to develop Prony's resolving method. Firstly, the Xilinx System Generator Tool was identified for implementing Prony's resolving logic and a lot of time was spent learning the tool and gaining better understanding of the tool and its capabilities.

The methodology used during this phase was to take Prony's resolving method and break it up into smaller parts. Each part was investigated in terms of how it could be implemented in System Generator. Then a part at a time was implemented and simulated in System Generator and the output of the simulation was checked with MATLAB to verify that the implementation was still correct. Thereafter the part was added in an FPGA (synthesised, placed and routed, programmed and tested) and checked with MATLAB to verify that the hardware implementation was still correct.

The discriminators and calculation of the constants (a_1 and a_2) were easily implemented and tested. After an attempt to calculate the roots, it was discovered that the Xilinx Virtex5 SX95t FPGA was running out of resources. Floating point calculations were then investigated and seemed to be a possible solution.

Unfortunately it was discovered that Xilinx System Generator only supports floating point calculations from Virtex6 onwards. The CSIR is in the process of developing a DRFM with a Xilinx Virtex6 FPGA, but it will only be available in a few months.

It was then decided rather to output the outputs of the discriminators from the FPGA and deal with Prony's resolving method in a MATLAB software application. This was implemented accordingly and tested in hardware and the results were captured.

The results from the MATLAB simulation were very good. The results of the hardware simulation and hardware implementation compared very well, but were not as good as the MATLAB simulation. This is because of limitations (sample width, resources etc.) in the hardware.

It is recommended for future research that the full implementation of Prony's method be implemented and tested when a DRFM with an FPGA (or DSP), which supports floating point calculation, becomes available, for instance, an implementation with no post-processing in MATLAB, i.e. an implementation that outputs the number of frequencies detected and the frequencies.

Pisarenko's method was not researched in this thesis, but it is recommended for future research. Prony's method is a mathematical method to solve a particular set of simultaneous nonlinear equations. Pisarenko's method uses the eigenvalue and eigenvector of the autocorrelation matrix. Prony's method is easier, but omits the noise problem completely. Pisarenko's method requires more calculations (more latency), but considers the noise problem. The results of Pisarenko's method can then be compared with Prony's method.

To conclude, the results achieved in the MATLAB simulations, hardware simulations and the hardware implementation compared very well and demonstrated a very feasible solution to the simultaneous signal problem of traditional IFM receivers.

References

- [1] James B. Tsui, *Digital Techniques for Wideband Receivers (2nd Edition)*, Scitech Publishing Inc, 2004
- [2] James B. Tsui, *Digital Microwave Receivers. Theory and concepts*, Artech House, 1989
- [3] James B. Tsui, *Microwave Receivers with Electronic Warfare Applications*, Artech House, 1986
- [4] David M. Rachman, *Polar Frequency Discriminators*, Master's dissertation, University of Cape Town, 1986
- [5] James B. Tsui, *IFM Receiver with Capability of Detecting Simultaneous Signals*, Microwave and Optical Technology Letters, 1989
- [6] H. Gruchalalla-Wesierski, *The Estimation of Simultaneous Signal Frequencies in the IFM Receiver using parametric methods*, Military University of Technology, Warsaw
- [7] P.L. Herselman, J.E. Cilliers, *A digital instantaneous frequency measurement technique using high-speed analogue-to-digital converters and field programmable gate arrays*, South African Journal of Science 102, July/August 2006
- [8] P.W. East, *Design Techniques and Performance of Digital IFM*, Proc. IEEE Vol 129, June 1982
- [9] J.P.Y. Lee, *Detection of complex and simultaneous signals using an instantaneous frequency measurement receiver*, Proc. IEE Vol. 132, 1985
- [10] Mark Zwolinski, *Digital System Design with VHDL (2nd Edition)*, Jones and Bartlett Publishers, 2000
- [11] Dennis G. Zill, Warren S. Wright, *Advanced Engineering Mathematics (2nd Edition)*, Jones and Bartlett Publishers, 2000
- [12] James Stewart, *Calculus (5th Edition)*, Brooks/Cole, 2003
- [13] *System Generator for DSP User Guide*, Xilinx, 2012
- [14] B. Klein, S. Hochgürtel, I. Krämer, A. Bell, K.Meyer, and R. Güsten, *High-resolution wide-band fast Fourier transform spectrometers*, Astronomy & Astrophysics, 2012
- [15] *Virtex-5 FPGA User Guide*, Xilinx, 2010
- [16] *Virtex-6 FPGA Configuration User Guide*, Xilinx, 2010
- [17] A.J. Weiss, B. Friedlander, *Simultaneous signals in IFM receivers*, IEE Proc. -Radar, Sensor Navig., Vol.144, No 4, June 1997
- [18] W.B. Sullivan, *Simultaneous signal errors in wideband IFM receivers*, Microwave Journal, September 1995
- [19] C.B. Hofmann, A.R. Baron, *Wideband ESM receiving systems (Part I)*, Microwave Journal, September 1980

- [20] C.B. Hofmann, A.R. Baron, *Wideband ESM receiving systems* (Part II), *Microwave Journal*, September 1980
- [21] M.I. Skolnik, *Introduction to Radar Systems*, McGraw-Hill Book Company, 1962
- [22] D. Heaton, *The system engineer's primer on IFM receivers*, *Microwave Journal*, February 1980
- [23] J.G. Proakis, D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*, 1995

Appendix A Single Delay Line Simulation MATLAB Code

```

% This one goes from almost 0 degrees to 360 degrees (200 MHz -
1000 MHz).

% Close all previous plots
close all;

% Frequency 1 [Hz]
fc1 = 200e6;
% Frequency 2 [Hz]
fc2 = 1000e6;

% Sampling rate [Hz]
fs = 10000e6;

% Time Steps
t = 0:1/fs:50000/fs;

disp('===== IFM Simulation =====\n')
disp('== 1. Input signal f = 200 MHz ==\n')
disp('== 2. Input signal f = 1000 MHz ==\n')

disp('== 3. Two input signals f1 = 200 MHz, f2 = 1000 MHz ==\n')
disp('=====\n')
a = input('Enter a option :');
if (a == 1) s = 50*cos(2*pi*fc1*t);
elseif (a == 2) s = 50*cos(2*pi*fc2*t);
elseif (a == 3) s = 50*cos(2*pi*fc1*t)+50*cos(2*pi*fc2*t); end;

% Plot the signal
figure(1);
plot(t*1e9,s);
set(gca, 'Xlim', [0 20]);
title('Input signal');
ylabel('Amplitude [mV]');
xlabel('Time [ns]');
grid on;
% Hilbert Transform - Transform Real Signal to Complex Signal
s_imag = hilbert(s);

% Delay Line Length = 9 sample delay
d1 = 9;

% Delay the Complex Signal
s_d1 = [s_imag(d1+1:end) zeros(1,d1)];

% Determine the Angle between the delayed signal and the original
signal

```

```

delay1 = angle(s_d1.*conj(s_imag));
% This angle is from -pi(-180 degrees)to +pi(180 degrees)
[radians]

% Convert the angle from 0 to 2*pi (360 degrees) [radians]
delay1(delay1<=0) = delay1(delay1<=0) + 2*pi;

% Display the angle in radians
delay1(100)
% Display the angle in degrees
delay1(100) * 180 / pi

% Changing angle to frequency (This is normally a lookup table)
w = delay1 / d1;
f = fs*w/(2*pi);

% Plot 2*pi ramp
x = linspace(0,pi,size(delay1,2));
xf=fs*x/(d1*pi*1000000); % Convert angle to frequency for plot
y1 = mod(2*x, 2*pi);

figure(2);
subplot(1,1,1);
plot(xf,delay1); % Horizontal Line = Angle
hold on;
plot(xf, y1,'r--'); % 2*pi Ramp
title('T = 9 Samples Delay Line');
ylabel('Phase [radians]');
xlabel('Frequency [MHz]');
grid on;
set(gca, 'Ylim', [0 2*pi]);

figure(3);
plot(f/1e6); ylabel('Frequency [MHz]');
title('Frequency Output of IFM');
xlabel('Time');
grid on;
set(gca, 'Ylim', [0 1200]);

```

Appendix B Multiple Delay Line Simulation MATLAB Code

```

% This one goes from almost 0 degrees to 360 degrees (200 MHz -
1000 MHz).

% Close all previous plots
close all;

% Frequency 1 [Hz]
fc1 = 200e6;
% Frequency 2 [Hz]
fc2 = 1000e6;

% Sampling rate [Hz]
fs = 10000e6;

% Time steps
t = 0:1/fs:50000/fs;

disp('===== IFM Simulation =====\n')
disp('== 1. Input signal f = 200 MHz ==\n')
disp('== 2. Input signal f = 1000 MHz ==\n')
disp('== 3. Two input signals f1 = 200 MHz, f2 = 1000 MHz ==\n')
disp('=====\n')
a = input('Enter a option :');
if (a == 1) s= 50*cos(2*pi*fc1*t);
elseif (a == 2) s= 50*cos(2*pi*fc2*t);
elseif (a == 3) s= 50*cos(2*pi*fc1*t)+50*cos(2*pi*fc2*t); end;

% Plot the signal
figure(1);
plot(t*1e9,s);
set(gca, 'Xlim', [0 20]);
title('Input signal');
ylabel('Amplitude [mV]');
xlabel('Time [ns]');
grid on;

% Hilbert Transform - Transform Real Signal to Complex Signal
s_imag = hilbert(s);

% Delay Line Length = 9 sample delay
d1 = 9;
% Delay Line Length = 18 (2 * 9) sample delay
d2 = 18;
% Delay Line Length = 36 (2 * 18) sample delay
d3 = 36;

```

```

% Delay the Complex Signal
s_d1 = [s_imag(d1+1:end) zeros(1,d1)];
s_d2 = [s_imag(d2+1:end) zeros(1,d2)];
s_d3 = [s_imag(d3+1:end) zeros(1,d3)];

% Determine the Angle between the delayed signal and the original
signal
delay1 = angle(s_d1.*conj(s_imag));
delay2 = angle(s_d2.*conj(s_imag));
delay3 = angle(s_d3.*conj(s_imag));
% This angle is from -pi(-180 degrees)to +pi(180 degrees)
[radians]

% Convert the angle from 0 to 2*pi (360 degrees) [radians]
delay1(delay1<=0) = delay1(delay1<=0) + 2*pi;
delay2(delay2<=0) = delay2(delay2<=0) + 2*pi;
delay3(delay3<=0) = delay3(delay3<=0) + 2*pi;

% Display the angle in radians
delay1(100)
delay2(100)
delay3(100)

% Display the angle in degrees
delay1(100) * 180 / pi
delay2(100) * 180 / pi
delay3(100) * 180 / pi

% Ambigiuty Resolving
k1 = zeros(size(delay1));
k2 = zeros(size(delay2));

k1(delay1 > pi) = 2;
k2(delay2 > pi) = 1;

k = k1+k2;

% Changing anlge to frequency (This is normally a lookup table)
w = (delay3 + 2*k*pi)/d3;
f = fs*w/(2*pi);

% 2*Pi ramp for Plot
x = linspace(0,pi,size(delay3,2));
xf1=fs*x/(d1*pi*1000000); % Convert angle to frequency for plot
y1 = mod(2*x, 2*pi);
xf2=2*fs*x/(d2*pi*1000000); % Convert angle to frequency for plot
y2 = mod(4*x, 2*pi);
xf3=4*fs*x/(d3*pi*1000000); % Convert angle to frequency for plot
y3 = mod(8*x, 2*pi);

figure(2);
subplot(3,1,1);

```



```
plot(xf1,delay1);
hold on;
plot(xf1, y1,'r--');
title('T = 9 Delay Line');
grid on;
set(gca, 'Ylim', [0 2*pi]);
ylabel('Phase [radians]');
xlabel('Frequency [MHz]');
grid on;
subplot(3,1,2);
plot(xf2,delay2);
hold on;
plot(xf2, y2,'r--');
title('T = 18 Delay Line');
grid on;
set(gca, 'Ylim', [0 2*pi]);
ylabel('Phase [radians]');
xlabel('Frequency [MHz]');
grid on;
subplot(3,1,3);
plot(xf3,delay3);
hold on;
plot(xf3, y3,'r--');
title('T = 27 Delay Line');
grid on;
set(gca, 'Ylim', [0 2*pi]);
ylabel('Phase [radians]');
xlabel('Frequency [MHz]');
grid on;

figure(3);
plot(f/1e6);
ylabel('Frequency [MHz]');
title('Frequency Output of IFM');
xlabel('Time');
grid on;
set(gca, 'Ylim', [0 1200]);
```

Appendix C Prony's Method Simulation MATLAB Code

C.1 Two Simultaneous Signals

```

% Clear all variables
clear all;

% Close all previous plots
close all;

% Sampling rate [Hz]
fs = 5000e6;

% Time steps
t = 0:1/fs:4095/fs;
%t = 0:1/fs:50000/fs;

disp('===== IFM Simulation =====\n')
disp('== 1. Input signal f = 200 MHz ==\n')
disp('== 2. Input signal f = 1000 MHz ==\n')
disp('== 3. Two input signals f1 = 200 MHz, f2 = 1000 MHz ==\n')
disp('== 4. Two input signals f1 = 300 MHz, f2 = 900 MHz ==\n')
disp('== 5. Two input signals f1 = 400 MHz, f2 = 800 MHz ==\n')
disp('== 6. Two input signals f1 = 500 MHz, f2 = 700 MHz ==\n')
disp('== 7. Two input signals f1 = 550 MHz, f2 = 650 MHz ==\n')
disp('== 8. Two input signals f1 = 600 MHz, f2 = 600 MHz ==\n')
disp('=====\n')
a = input('Enter a option :');
if (a == 1) s= 50*cos(2*pi*200e6*t);
elseif (a == 2) s= 50*cos(2*pi*1000e6*t);
elseif (a == 3) s= 50*sin(2*pi*200e6*t)+50*cos(2*pi*1000e6*t);
elseif (a == 4) s= 50*sin(2*pi*300e6*t)+50*cos(2*pi*900e6*t);
elseif (a == 5) s= 50*sin(2*pi*400e6*t)+50*cos(2*pi*800e6*t);
elseif (a == 6) s= 50*sin(2*pi*500e6*t)+50*cos(2*pi*700e6*t);
elseif (a == 7) s= 50*sin(2*pi*550e6*t)+50*cos(2*pi*650e6*t);
elseif (a == 8) s= 50*sin(2*pi*600e6*t)+50*cos(2*pi*600e6*t);
end;

% Plot the signal
figure(1);
plot(t*1e9,s);
set(gca, 'Xlim', [0 20]);
title('Input signal');
ylabel('Amplitude [mV]');
xlabel('Time [ns]');
grid on;

% Hilbert Transform - Transform Real Signal to Complex Signal

```

```

s_imag = hilbert(s);

% Delay Line Length = 2 sample delay
d1 = 2;
% Delay Line Length = 4 (2 * 2) sample delay
d2 = 4;
% Delay Line Length = 6 (3 * 2) sample delay
d3 = 6;
% Delay Line Length = 8 (4 * 2) sample delay
d4 = 8;

% Delay the Complex Signal
s_d1 = [s_imag(d1+1:end) zeros(1,d1)];
s_d2 = [s_imag(d2+1:end) zeros(1,d2)];
s_d3 = [s_imag(d3+1:end) zeros(1,d3)];
s_d4 = [s_imag(d4+1:end) zeros(1,d4)];

% Determine the SIN and COS components
% Multiply the delayed signal with the original signal, apply lpf
to filter
% out high frequency component
%lpf = [-2.6498e-019 7.977e-005 0.00036582 0.00095245 0.0019625
0.0035319 0.0057835 0.0087932 0.012558 0.01697 0.021811 0.026759
0.031423 0.035391 0.038282 0.039806 0.039806 0.038282 0.035391
0.031423 0.026759 0.021811 0.01697 0.012558 0.0087932 0.0057835
0.0035319 0.0019625 0.00095245 0.00036582 7.977e-005 -2.6498e-
019];

%OR

% Returns a discrete-time filter object.

%
% MATLAB Code
% Generated by MATLAB(R) 8.0 and the Signal Processing Toolbox
6.18.
%
% Generated on: 05-Nov-2013 09:29:34
%

% Equiripple Lowpass filter designed using the FIRPM function.

% All frequency values are in MHz.
Fs = fs/1000000;

N      = 100; % Order
Fpass  = 1;   % Passband Frequency
Fstop  = 200; % Stopband Frequency
Wpass  = 1;   % Passband Weight

```

```

Wstop = 1;      % Stopband Weight
dens   = 20;    % Density Factor

% Calculate the coefficients using the FIRPM function.
lpf = firpm(N, [0 Fpass Fstop Fs/2]/(Fs/2), [1 1 0 0], [Wpass
Wstop], ...
           {dens});

s_delay1 = filter(lpf,1,s_d1.*conj(s_imag));
s_delay2 = filter(lpf,1,s_d2.*conj(s_imag));
s_delay3 = filter(lpf,1,s_d3.*conj(s_imag));
s_delay4 = filter(lpf,1,s_d4.*conj(s_imag));

% Prony's Method
r_1 = s_delay1;
r_2 = s_delay2;
r_3 = s_delay3;
r_4 = s_delay4;

for counter = 1:1:size(r_1,2)
    a1(counter) = det([-1*r_4(counter) r_2(counter);-1*r_3(counter)
r_1(counter)])/det([r_3(counter) r_2(counter);r_2(counter)
r_1(counter)]);
    a2(counter) = det([r_3(counter) -1*r_4(counter);r_2(counter) -
1*r_3(counter)])/det([r_3(counter) r_2(counter);r_2(counter)
r_1(counter)]);

    r = roots([1 a1(counter) a2(counter)]);

    f1(counter) = angle(r(1))/(2*pi*4e-10);      % 2*pi*T
    f2(counter) = angle(r(2))/(2*pi*4e-10);

end;

figure(2);
plot(t*1e9,f1/1e6,'b. ');
hold;
plot(t*1e9,f2/1e6,'b. ');
xlabel('Time [ns]');
ylabel('Frequency [MHz]'); title('Frequency Outputs of IFM');
set(gca, 'Ylim', [0 1200]);
grid on;
hold;

```

C.2 Three Simultaneous Signals

```

% Clear all variables
clear all;

% Close all previous plots
close all;

% Sampling rate [Hz]
fs = 5000e6;

% Time steps
t = 0:1/fs:4095/fs;
%t = 0:1/fs:10000/fs;

disp('===== IFM Simulation
=====\n')
disp('== 1. Three input signals f1 = 200 MHz, f2 = 600 MHz, f3 =
1000 MHz ==\n')
disp('== 2. Three input signals f1 = 200 MHz, f2 = 400 MHz, f3 =
600 MHz ==\n')
disp('=====
=====\n')
a = input('Enter a option :');
if (a == 1) s=
50*cos(2*pi*200e6*t)+50*cos(2*pi*600e6*t)+50*cos(2*pi*1000e6*t);
elseif (a == 2) s=
50*cos(2*pi*200e6*t)+50*cos(2*pi*400e6*t)+50*cos(2*pi*600e6*t);
%elseif (a == 3) s= 50*sin(2*pi*200e6*t)+50*cos(2*pi*1000e6*t);
%elseif (a == 4) s= 50*sin(2*pi*300e6*t)+50*cos(2*pi*900e6*t);
%elseif (a == 5) s= 50*sin(2*pi*400e6*t)+50*cos(2*pi*800e6*t);
%elseif (a == 6) s= 50*sin(2*pi*500e6*t)+50*cos(2*pi*700e6*t);
%elseif (a == 7) s= 50*sin(2*pi*550e6*t)+50*cos(2*pi*650e6*t);
%elseif (a == 8) s= 50*sin(2*pi*600e6*t)+50*cos(2*pi*600e6*t);
end;

% Plot the signal
figure(1);
plot(t*1e9,s);
set(gca, 'Xlim', [0 20]);
title('Input signal');
ylabel('Amplitude [mV]');
xlabel('Time [ns]');
grid on

% Hilbert Transform - Transform Real Signal to Complex Signal
s_imag = hilbert(s);

```

```
% Delay Line Length = 2 sample delay
d1 = 2;
% Delay Line Length = 4 (2 * 2) sample delay
d2 = 4;
% Delay Line Length = 6 (3 * 2) sample delay
d3 = 6;
% Delay Line Length = 8 (4 * 2) sample delay
d4 = 8;
% Delay Line Length = 10 (5 * 2) sample delay
d5 = 10;
% Delay Line Length = 12 (6 * 2) sample delay
d6 = 12;

% Delay the Complex Signal
s_d1 = [s_imag(d1+1:end) zeros(1,d1)];
s_d2 = [s_imag(d2+1:end) zeros(1,d2)];
s_d3 = [s_imag(d3+1:end) zeros(1,d3)];
s_d4 = [s_imag(d4+1:end) zeros(1,d4)];
s_d5 = [s_imag(d5+1:end) zeros(1,d5)];
s_d6 = [s_imag(d6+1:end) zeros(1,d6)];

% Determine the SIN and COS components
% Multiply the delayed signal with the original signal, apply lpf
to filter
% out high frequency component
%lpf = [-2.6498e-019 7.977e-005 0.00036582 0.00095245 0.0019625
0.0035319 0.0057835 0.0087932 0.012558 0.01697 0.021811 0.026759
0.031423 0.035391 0.038282 0.039806 0.039806 0.038282 0.035391
0.031423 0.026759 0.021811 0.01697 0.012558 0.0087932 0.0057835
0.0035319 0.0019625 0.00095245 0.00036582 7.977e-005 -2.6498e-
019];

%OR

% Returns a discrete-time filter object.

%
% MATLAB Code
% Generated by MATLAB(R) 8.0 and the Signal Processing Toolbox
6.18.
%
% Generated on: 05-Nov-2013 09:29:34
%

% Equiripple Lowpass filter designed using the FIRPM function.

% All frequency values are in MHz.
Fs = fs/1000000;
```

```

N      = 100; % Order
Fpass  = 1;   % Passband Frequency
Fstop  = 200; % Stopband Frequency
Wpass  = 1;   % Passband Weight
Wstop  = 1;   % Stopband Weight
dens   = 20;  % Density Factor

% Calculate the coefficients using the FIRPM function.
lpf = firpm(N, [0 Fpass Fstop Fs/2]/(Fs/2), [1 1 0 0], [Wpass
Wstop], ...
           {dens});

s_delay1 = filter(lpf,1,s_d1.*conj(s_imag));
s_delay2 = filter(lpf,1,s_d2.*conj(s_imag));
s_delay3 = filter(lpf,1,s_d3.*conj(s_imag));
s_delay4 = filter(lpf,1,s_d4.*conj(s_imag));
s_delay5 = filter(lpf,1,s_d5.*conj(s_imag));
s_delay6 = filter(lpf,1,s_d6.*conj(s_imag));

% Prony's Method
r_1 = s_delay1;
r_2 = s_delay2;
r_3 = s_delay3;
r_4 = s_delay4;
r_5 = s_delay5;
r_6 = s_delay6;

for counter = 1:1:size(r_1,2)
    %counter = 129;

    deter1(counter) = det([r_5(counter) r_4(counter)
r_3(counter);r_4(counter) r_3(counter) r_2(counter);r_3(counter)
r_2(counter) r_1(counter)]);
    a1(counter) = det([-1*r_6(counter) r_4(counter) r_3(counter);-
1*r_5(counter) r_3(counter) r_2(counter);-1*r_4(counter)
r_2(counter) r_1(counter)])/det([r_5(counter) r_4(counter)
r_3(counter);r_4(counter) r_3(counter) r_2(counter);r_3(counter)
r_2(counter) r_1(counter)]);
    a2(counter) = det([r_5(counter) -1*r_6(counter)
r_3(counter);r_4(counter) -1*r_5(counter)
r_2(counter);r_3(counter) -1*r_4(counter)
r_1(counter)])/det([r_5(counter) r_4(counter)
r_3(counter);r_4(counter) r_3(counter) r_2(counter);r_3(counter)
r_2(counter) r_1(counter)]);
    a3(counter) = det([r_5(counter) r_4(counter) -
1*r_6(counter);r_4(counter) r_3(counter) -

```

```

1*r_5(counter);r_3(counter) r_2(counter) -
1*r_4(counter)]/det([r_5(counter) r_4(counter)
r_3(counter);r_4(counter) r_3(counter) r_2(counter);r_3(counter)
r_2(counter) r_1(counter)]);

r = roots([1 a1(counter) a2(counter) a3(counter)]);

f1(counter) = angle(r(1))/(2*pi*4e-10); % 2*pi*T
f2(counter) = angle(r(2))/(2*pi*4e-10);
f3(counter) = angle(r(3))/(2*pi*4e-10);

end;

figure(2);
plot(t*1e9,f1/1e6,'b. ');
hold;
plot(t*1e9,f2/1e6,'b. ')
plot(t*1e9,f3/1e6,'b. ')
xlabel('Time [ns]');
ylabel('Frequency [MHz]'); title('Frequency Outputs of IFM');
set(gca, 'Ylim', [0 1200]);
grid on;
hold;

```

C.3 Four Simultaneous Signals

```

% Clear all variables
clear all;

% Close all previous plots
close all;

% Sampling rate [Hz]
fs = 5000e6;

% Time steps
t = 0:1/fs:4095/fs;
%t = 0:1/fs:10000/fs;

disp('==== IFM Simulation
====\n')
disp('== 1. Four input signals f1 = 100 MHz, f2 = 400 MHz, f3 =
700 MHz f4 = 1000 MHz ==\n')

```



```

disp('=====
=====\\n')
a = input('Enter a option :');
if (a == 1) s=
50*cos(2*pi*100e6*t)+50*cos(2*pi*400e6*t)+50*cos(2*pi*700e6*t)+50*
cos(2*pi*1000e6*t);
%elseif (a == 3) s= 50*sin(2*pi*200e6*t)+50*cos(2*pi*1000e6*t);
%elseif (a == 4) s= 50*sin(2*pi*300e6*t)+50*cos(2*pi*900e6*t);
%elseif (a == 5) s= 50*sin(2*pi*400e6*t)+50*cos(2*pi*800e6*t);
%elseif (a == 6) s= 50*sin(2*pi*500e6*t)+50*cos(2*pi*700e6*t);
%elseif (a == 7) s= 50*sin(2*pi*550e6*t)+50*cos(2*pi*650e6*t);
%elseif (a == 8) s= 50*sin(2*pi*600e6*t)+50*cos(2*pi*600e6*t);
end;

% Plot the signal
figure(1);
plot(t*1e9,s);
set(gca, 'Xlim', [0 20]);
title('Input signal');
ylabel('Amplitude [mV]');
xlabel('Time [ns]');
grid on

% Hilbert Transform - Transform Real Signal to Complex Signal
s_imag = hilbert(s);

% Delay Line Length = 2 sample delay
d1 = 2;
% Delay Line Length = 4 (2 * 2) sample delay
d2 = 4;
% Delay Line Length = 6 (3 * 2) sample delay
d3 = 6;
% Delay Line Length = 8 (4 * 2) sample delay
d4 = 8;
% Delay Line Length = 10 (5 * 2) sample delay
d5 = 10;
% Delay Line Length = 12 (6 * 2) sample delay
d6 = 12;
% Delay Line Length = 14 (7 * 2) sample delay
d7 = 14;
% Delay Line Length = 16 (8 * 2) sample delay
d8 = 16;

% Delay the Complex Signal
s_d1 = [s_imag(d1+1:end) zeros(1,d1)];
s_d2 = [s_imag(d2+1:end) zeros(1,d2)];
s_d3 = [s_imag(d3+1:end) zeros(1,d3)];
s_d4 = [s_imag(d4+1:end) zeros(1,d4)];
s_d5 = [s_imag(d5+1:end) zeros(1,d5)];
s_d6 = [s_imag(d6+1:end) zeros(1,d6)];
s_d7 = [s_imag(d7+1:end) zeros(1,d7)];

```

```

s_d8 = [s_imag(d8+1:end) zeros(1,d8)];

% Determine the SIN and COS components
% Multiply the delayed signal with the original signal, apply lpf
to filter
% out high frequency component
%lpf = [-2.6498e-019 7.977e-005 0.00036582 0.00095245 0.0019625
0.0035319 0.0057835 0.0087932 0.012558 0.01697 0.021811 0.026759
0.031423 0.035391 0.038282 0.039806 0.039806 0.038282 0.035391
0.031423 0.026759 0.021811 0.01697 0.012558 0.0087932 0.0057835
0.0035319 0.0019625 0.00095245 0.00036582 7.977e-005 -2.6498e-
019];

%OR
% Returns a discrete-time filter object.

%
% MATLAB Code
% Generated by MATLAB(R) 8.0 and the Signal Processing Toolbox
6.18.
%
% Generated on: 05-Nov-2013 09:29:34
%

% Equiripple Lowpass filter designed using the FIRPM function.

% All frequency values are in MHz.
Fs = fs/1000000;

N      = 100; % Order
Fpass  = 1;   % Passband Frequency
Fstop  = 200; % Stopband Frequency
Wpass  = 1;   % Passband Weight
Wstop  = 1;   % Stopband Weight
dens   = 20;  % Density Factor

% Calculate the coefficients using the FIRPM function.
lpf = firpm(N, [0 Fpass Fstop Fs/2]/(Fs/2), [1 1 0 0], [Wpass
Wstop], ...
           {dens});

s_delay1 = filter(lpf,1,s_d1.*conj(s_imag));
s_delay2 = filter(lpf,1,s_d2.*conj(s_imag));
s_delay3 = filter(lpf,1,s_d3.*conj(s_imag));
s_delay4 = filter(lpf,1,s_d4.*conj(s_imag));
s_delay5 = filter(lpf,1,s_d5.*conj(s_imag));
s_delay6 = filter(lpf,1,s_d6.*conj(s_imag));
s_delay7 = filter(lpf,1,s_d7.*conj(s_imag));
s_delay8 = filter(lpf,1,s_d8.*conj(s_imag));

```

```

% Prony's Method
r_1 = s_delay1;
r_2 = s_delay2;
r_3 = s_delay3;
r_4 = s_delay4;
r_5 = s_delay5;
r_6 = s_delay6;
r_7 = s_delay7;
r_8 = s_delay8;

for counter = 1:1:size(r_1,2)
    %counter = 129;

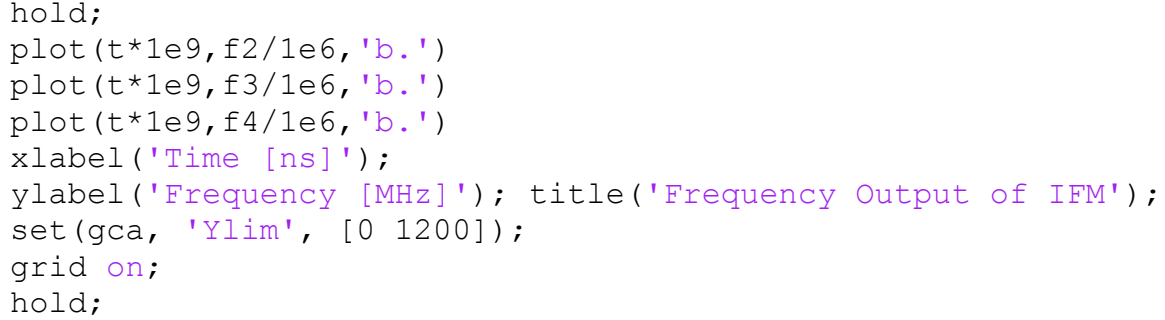
    deter1(counter) = det([r_7(counter) r_6(counter) r_5(counter)
r_4(counter);r_6(counter) r_5(counter) r_4(counter)
r_3(counter);r_5(counter) r_4(counter) r_3(counter)
r_2(counter);r_4(counter) r_3(counter) r_2(counter)
r_1(counter)]);
    a1(counter) = det([-1*r_8(counter) r_6(counter) r_5(counter)
r_4(counter);-1*r_7(counter) r_5(counter) r_4(counter)
r_3(counter);-1*r_6(counter) r_4(counter) r_3(counter)
r_2(counter);-1*r_5(counter) r_3(counter) r_2(counter)
r_1(counter)])/deter1(counter);
    a2(counter) = det([r_7(counter) -1*r_8(counter) r_5(counter)
r_4(counter);r_6(counter) -1*r_7(counter) r_4(counter)
r_3(counter);r_5(counter) -1*r_6(counter) r_3(counter)
r_2(counter);r_4(counter) -1*r_5(counter) r_2(counter)
r_1(counter)])/deter1(counter);
    a3(counter) = det([r_7(counter) r_6(counter) -1*r_8(counter)
r_4(counter);r_6(counter) r_5(counter) -1*r_7(counter)
r_3(counter);r_5(counter) r_4(counter) -1*r_6(counter)
r_2(counter);r_4(counter) r_3(counter) -1*r_5(counter)
r_1(counter)])/deter1(counter);
    a4(counter) = det([r_7(counter) r_6(counter) r_5(counter) -
1*r_8(counter);r_6(counter) r_5(counter) r_4(counter) -
1*r_7(counter);r_5(counter) r_4(counter) r_3(counter) -
1*r_6(counter);r_4(counter) r_3(counter) r_2(counter) -
1*r_5(counter)])/deter1(counter);

    r = roots([1 a1(counter) a2(counter) a3(counter) a4(counter)]);

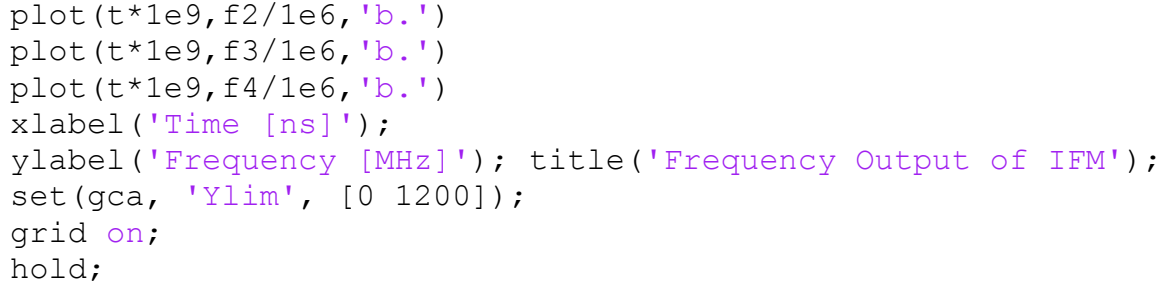
    f1(counter) = angle(r(1))/(2*pi*4e-10);    % 2*pi*T
    f2(counter) = angle(r(2))/(2*pi*4e-10);
    f3(counter) = angle(r(3))/(2*pi*4e-10);
    f4(counter) = angle(r(4))/(2*pi*4e-10);

```

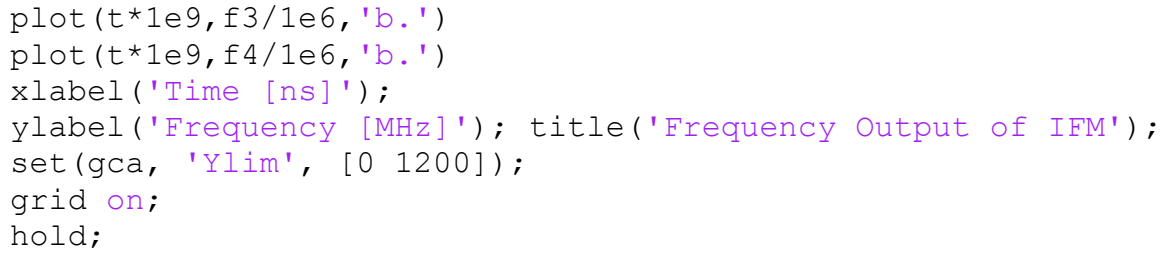
```
end;  
  
figure(2);  
plot(t*1e9,f1/1e6,'b.');
```



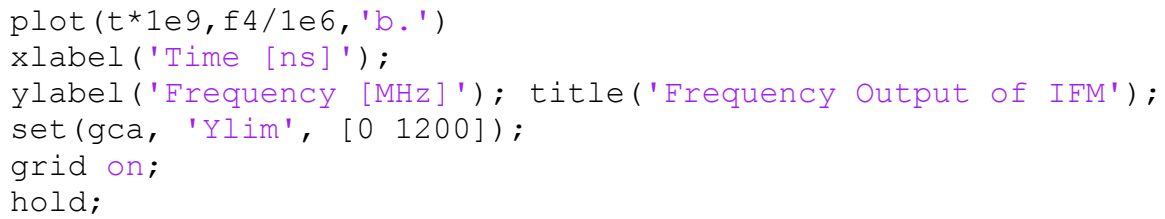
```
hold;  
plot(t*1e9,f2/1e6,'b.')
```



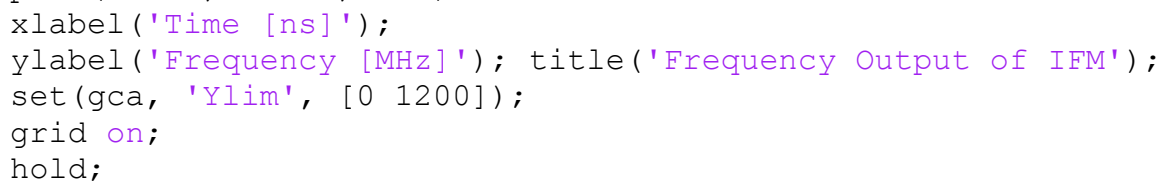
```
plot(t*1e9,f3/1e6,'b.')
```



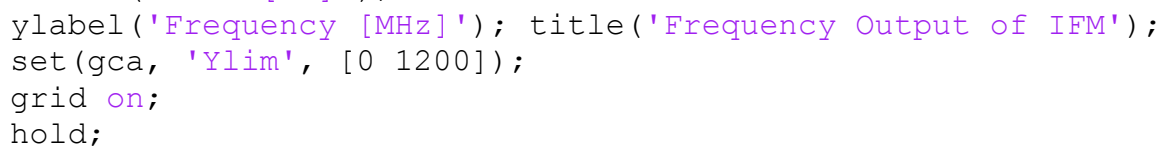
```
plot(t*1e9,f4/1e6,'b.')
```




```
xlabel('Time [ns]');
```




```
ylabel('Frequency [MHz]'); title('Frequency Output of IFM');
```




```
set(gca, 'Ylim', [0 1200]);
```



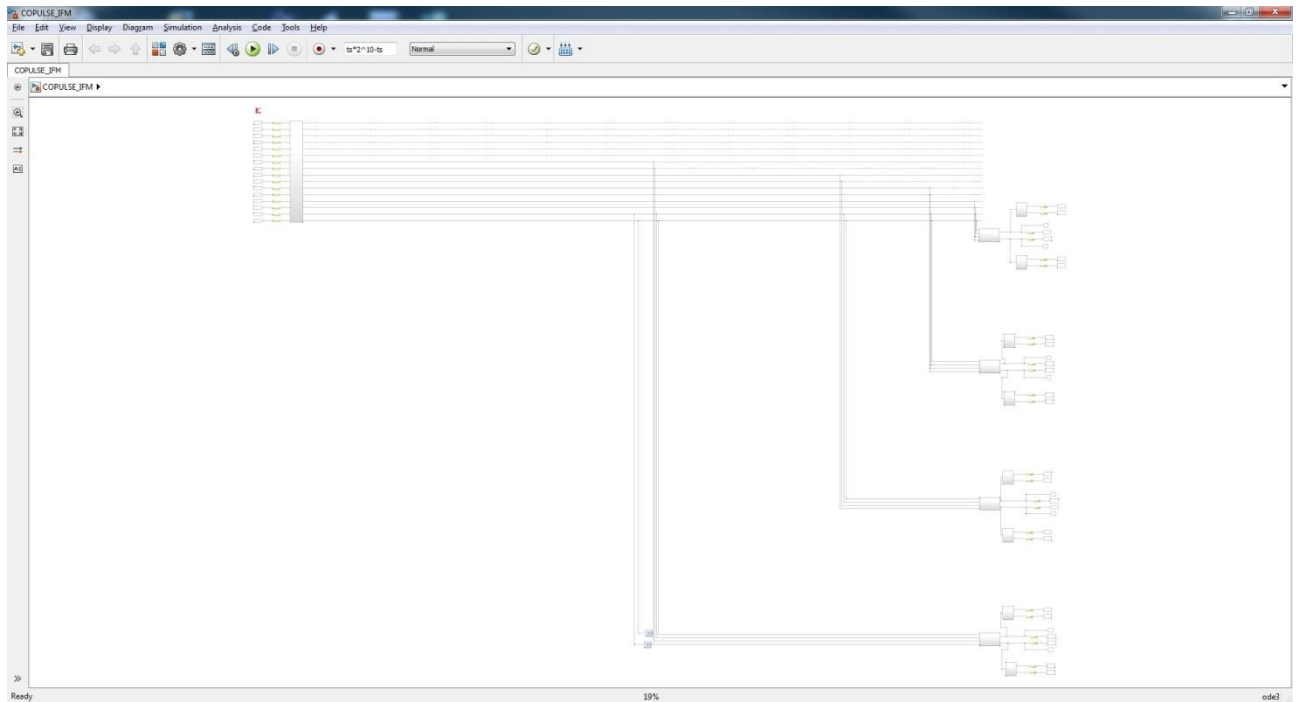
```
grid on;
```



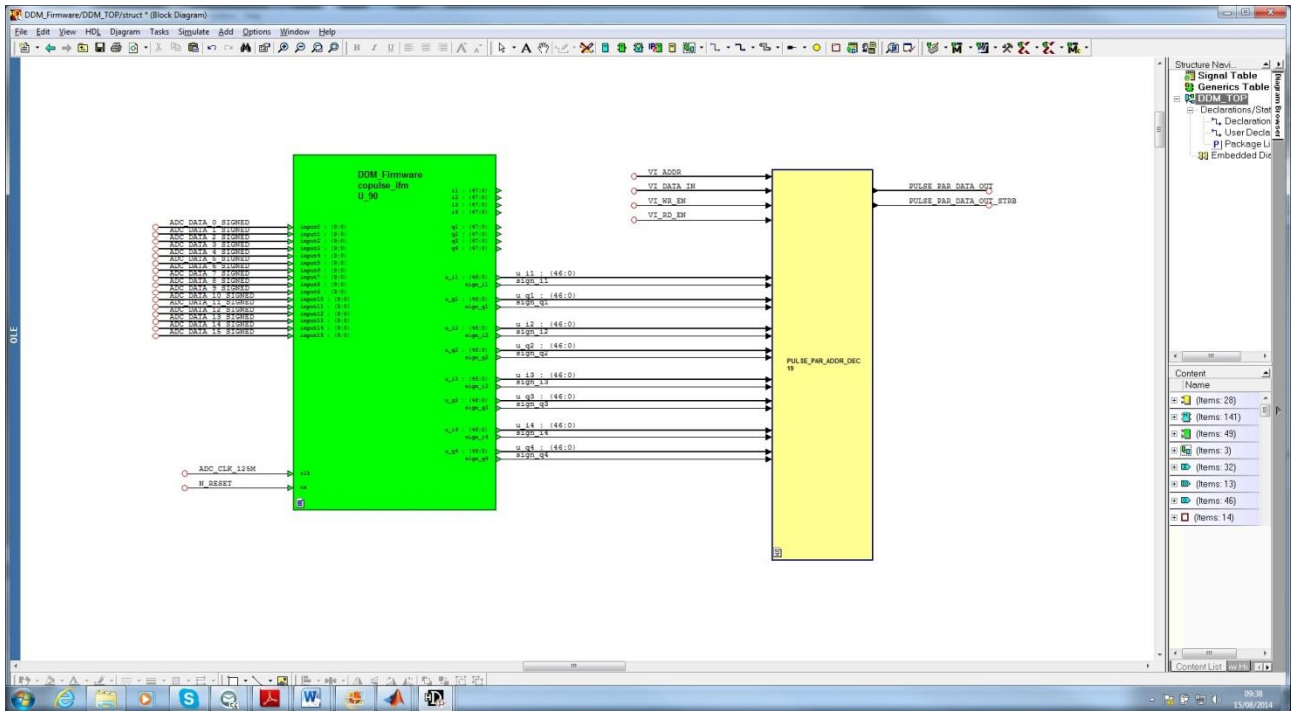
```
hold;
```



Appendix D Co-pulse DIFM System Generator Top Level



Appendix E Co-pulse IFM in Top Level of Firmware Implementation



Appendix F Roots of a Cubic Equation

$$ax^3 + bx^2 + cx + d = 0$$

$$x_1 = \frac{\sqrt[3]{\sqrt{(-27a^2d + 9abc - 2b^3)^2 + 4(3ac - b^2)^3} - 27a^2d + 9abc - 2b^3}}{3\sqrt[3]{2a}} - \frac{b}{3a}$$

$$x_2 = -\frac{1}{6\sqrt[3]{2a}}(1 - i\sqrt{3})$$

$$\frac{\sqrt[3]{\sqrt{(-27a^2d + 9abc - 2b^3)^2 + 4(3ac - b^2)^3} - 27a^2d + 9abc - 2b^3} + (1 + i\sqrt{3})(3ac - b^2)}}{3 \times 2^{2/3} a^3 \sqrt[3]{\sqrt{(-27a^2d + 9abc - 2b^3)^2 + 4(3ac - b^2)^3} - 27a^2d + 9abc - 2b^3}} - \frac{b}{3a}$$

$$x_3 = -\frac{1}{6\sqrt[3]{2a}}(1 + i\sqrt{3})$$

$$\frac{\sqrt[3]{\sqrt{(-27a^2d + 9abc - 2b^3)^2 + 4(3ac - b^2)^3} - 27a^2d + 9abc - 2b^3} + (1 - i\sqrt{3})(3ac - b^2)}}{3 \times 2^{2/3} a^3 \sqrt[3]{\sqrt{(-27a^2d + 9abc - 2b^3)^2 + 4(3ac - b^2)^3} - 27a^2d + 9abc - 2b^3}} - \frac{b}{3a}$$

Appendix G MATLAB Post-processing

```

%% Phase1: Calculate the frequencies from the I and Q Values
for counter = 25:1:1000

    r_1(counter) = combined_IQ(counter,1) +
i*combined_IQ(counter,2);
    r_2(counter) = combined_IQ(counter,3)+ i*combined_IQ(counter,4);
    r_3(counter) = combined_IQ(counter,5) +
i*combined_IQ(counter,6);
    r_4(counter) = combined_IQ(counter,7) +
i*combined_IQ(counter,8);

    divide_by(counter) = abs(det([r_3(counter)
r_2(counter);r_2(counter) r_1(counter)]));

    if (divide_by(counter) > 10e9)
        a1_matlab(counter) = det([-1*r_4(counter) r_2(counter);-
1*r_3(counter) r_1(counter)])/det([r_3(counter)
r_2(counter);r_2(counter) r_1(counter)]);
        a2_matlab(counter) = det([r_3(counter) -
1*r_4(counter);r_2(counter) -1*r_3(counter)])/det([r_3(counter)
r_2(counter);r_2(counter) r_1(counter)]);

        r_matlab = roots([1 a1_matlab(counter) a2_matlab(counter)]);

        %% if bigger 500 than values is negative. Offset negative
value by 1000
        if ((angle(r_matlab(1))/(2*pi*0.001)) > 0)
            f1_matlab(counter) = angle(r_matlab(1))/(2*pi*0.001);    %
2*pi*T
        else
            f1_matlab(counter) = (angle(r_matlab(1))/(2*pi*0.001))+1000;
        end;

        if ((angle(r_matlab(2))/(2*pi*0.001)) > 0)
            f2_matlab(counter) = angle(r_matlab(2))/(2*pi*0.001);    %
2*pi*T
        else
            f2_matlab(counter) = (angle(r_matlab(2))/(2*pi*0.001))+1000;
        end;
    else
        f1_matlab(counter) = -100;
        f2_matlab(counter) = -100;

    end;
end;

```