MODULATION CLASSIFICATION OF LPI RADAR SIGNALS USING WIGNER VILLE DISTRIBUTION AND ITS IMPLEMENTATION IN FPGA

MASTER OF TECHNOLOGY
IN
INTEGRATED CIRCUIT TECHNOLOGY

MARUTHI M
06PHMI03

School of Physics
UNIVERSITY OF HYDERABAD
HYDERABAD - 5000046

June 2008

MODULATION CLASSIFICATION OF LPI RADAR SIGNALS USING WIGNER VILLE DISTRIBUTION AND ITS IMPLEMENTATION IN FPGA

Submitted
In partial fulfillment of the academic requirements for
the award of

MASTER OF TECHNOLOGY
IN
INTEGRATED CIRCUIT TECHNOLOGY

By
MARUTHI M
06PHMI03

School of Physics
UNIVERSITY OF HYDERABAD
HYDERABAD - 5000046

June 2008

UNIVERSITY OF HYDERABAD

HYDERABAD-5000046

INDIA

DECLARATION

I, Maruthi M here by declare that the work embodied in this dissertation entitled **"MODULATION CLASSIFICATION OF LPI RADAR SIGNALS USING WIGNER VILLE DISTRIBUTION AND ITS IMPLEMENTATION IN FPGA"** submitted to the University Of Hyderabad for partial fulfillment of the degree of **M.Tech in Integrated Circuit Technology** has been carried out by me under the supervision of **Dr.Samrat L. Sabat**, School of Physics, University Of Hyderabad. To the best of my knowledge, this work has not been submitted for any other degree in any University.

Maruthi M

M.Tech(IC Technology)

Reg.NO: 06PHMI03

UNIVERSITY OF HYDERABAD

HYDERABAD-5000046

INDIA

CERTIFICATE

This is to certify that the project work "**MODULATION CLASSIFICATION OF LPI RADAR SIGNALS USING WIGNER VILLE DISTRIBUTION AND ITS IMPLEMENTATION IN FPGA**" carried out by **Mr. Maruthi M** bearing the Reg.No.06PHMI03 under my guidance in partial fulfillment of the requirements for the award of **Master of Technology in Integrated Circuit Technology** in University of Hyderabad. The matter embodied in this thesis has not been submitted in any other university for the award of any other degree.

Dr. Samrat L. Sabat,
Project Supervisor
School of Physics

Dean
School of Physics.

ACKNOWLEDGEMENTS

This work would not have been complete without the help and support of many people. I am grateful to my supervisor **Dr. Samrat L.Sabat,** University of Hyderabad for his invaluable guidance and support throughout the project work, and more importantly during times of extreme crisis.

I take this opportunity to thank the members of my family and all my friends for their moral support during times of depression. Their encouragement and support was invaluable in completing this work. I specifically thanks to Xilinx for providing tools under XUP program to carryout this work.

ABSTRACT

The Low Probability of Intercept (LPI) radar signals are hard to identify by using traditional signal processing techniques (such as FFT) due to its non-stationary behavior. So Time Frequency Analysis (TFA) is a good choice to analyze such type of signals. In this report Wigner Ville Distribution (WVD) algorithm is being used for analyzing and extracting features of these signals

The modulation classification and extraction of different parameters of modulated signals are carried out by performing WVD operation of input signal followed by a series of image processing technique such as thresholding, binarization, cropping . The proposed technique is verified in a MATLAB simulation environment. Different types of modulation signals are simulated and proposed scheme is applied to the simulated signals for modulation classification. Simulation result infers that the proposed techniques works satisfactory and classifies the simulated signal with 95% accuracy.

Since the WVD algorithm is computationally intensive operation as it performs FFT operation, it takes considerably amount of time. A novel approach for partitioning this algorithm is carried out and part of this operation is implemented in FPGA to accelerate the speed of design. The simulation for verifying its operation is done by using Hardware In Loop (HIL) simulation.

HIL simulation result also infers the acceleration of implementing FFT of the WVD algorithm in FPGA.

This dissertation presents the classification of LPI signals using WVD and their parameter extraction. It includes the classification and parameter extraction of BPSK, FMCW and Polyphase codes. WVD is designed with FFT implemented into FPGA, Kernel using sysgen black box and verified their functionality using HIL simulation. The dissertation also compares and presents the resources utilized and time performance of the WVD with Vertex 2Pro and Spartan 3E FPGAs.

Contents **Page No**

About the CD

The CD-ROM accompanying this thesis includes all the models of the Sysgen used in this dissertation. It also includes the MATLAB code used in this work. For details see the related documents in the CD.

LIST OF FIGURES:

LIST OF TABLES

ABBREVATIONS

ASIC	Application-Specific Integrated Circuit
CLB	Configurable Logic Block
DSP	Digital Signal Processing/ Processor
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
HIL	Hardware in the Loop
ISE	Integrated Software Environment
LPI	Low Probability of Intercept
STFT	Short Time Fourier Transform
WVD	Wigner Ville Distribution
TFA	Time Frequency Analysis

CHAPTER 1

INTRODUCTION

1.1Statement of the Problem

In this work, a new approach for classifying LPI radar signals and extracting different parameters of modulated signal is proposed. The classification and parameter extraction of LPI radar signals using Wigner Ville Distribution (WVD) is computational intensive and time consuming process. Hence it is necessary to reduce the computational time of WVD.

1.2 Motivation and objective of this study

LPI Radar signals are difficult to identify because of their low power, wide bandwidth and frequency variability, hence it became a challenging field for the academia. Now a days, identification of LPI radar signals has become an important issue because of its properties. It is hard to analyze the LPI radar signal using traditional frequency analysis like FFT, hence TFA analysis is being used in this report to classify and extract the LPI radar signal parameters.

The different types of LPI radar signals using WVD are examined and their parameters are extracted manually [2]. It is a time consuming process to manually classify and extract the different parameters of the LPI radar signals, hence it is required to develop an algorithm to classify and extract different parameters of LPI radar signals without any manual intervention.

The Autonomous classification of LPI radar signals and feature extraction algorithm using neural networks is studied by Zilberman [3], it gives poorer classification results for all LPI signals except BPSK signal [3].

The objective of this work is i) to develop an efficient classification algorithm for LPI radar signals In this report signal analysis is carried out using WVD algorithm. Since WVD takes more time for its computation so the next objective of the work is ii) to find a technique, which can accelerate the computational time of WVD operation

1.3 Technical Approach

The objective of modulation classification of LPI radar signal is to identify the types of modulation carried out in emitter radar. The processing is carried out by using well known Time Frequency Analysis algorithm known as Wigner Ville Distribution algorithm. [8]. It has advantages of independently controlling the time and frequency resolution [2]. Modulation classification and extraction of different parameters such as carrier frequency, bandwidth etc., are carried out using WVD and image processing techniques. WVD is a computationally intensive operation [2] as it calculates Fast Fourier Transform (FFT) of a signal N^2 times, where N is number of input samples. So to accelerate it, FFT is being implemented in FPGA and acceleration in computation is studied in this work. The results obtained in HIL simulation is compared with results obtained from MATLAB.

1.4 Thesis Scope

This thesis is restricted to define algorithms and strategies for classification of LPI radar signals using WVD and implementation of a part of this algorithm in FPGA to accelerate the design. Algorithmic verification is carried out by HIL simulation of WVD in FPGA.

1.5 Hardware in the Loop (HIL) Simulation

Traditionally functional testing of a design is done by giving a set of known inputs and measuring the response of the system to these inputs. Now a days there is more pressure to get products to market faster and reduce design cycle times. This has led to a need for *dynamic* testing, where designs are tested while in use with the entire system. Because of the cost and safety concerns, simulating the rest of the system with real-time hardware is preferred for testing individual components of in the actual system. Dynamic testing also encompasses a larger range of test conditions compared to static testing. *Hardware-in-the-Loop* (HIL) *simulation* [7] is one of the dynamic testing strategy used in this work.

1.5.1 Advantages of HIL simulation

A hardware subsystem can be tested using HIL simulation without having the entire system ready, thus making testing an effective part of the development process, from design to deployment. Real-time HIL simulations not only enable shorter time to complete the design by reducing the development period, but also reduce cost of testing the design.

1.6 Field Programmable Gate Array (FPGA)

The FPGA is two-dimensional array of configurable (programmable) logic blocks along with configurable interconnects between these blocks. Design engineers can configure (program) such devices to perform a tremendous variety of tasks [1]. The internal architecture of FPGA is shown in figure 1.1.

FPGAs are composed of an array of Configurable Logic Blocks (CLBs) surrounded by a ring of Input/Outputs Blocks (IOBs). The CLBs are the primary building blocks that contain elements for implementing customizable gates, flip flops, and wiring for connectivity. The IOBs provide circuitry for communicating signals with external devices.

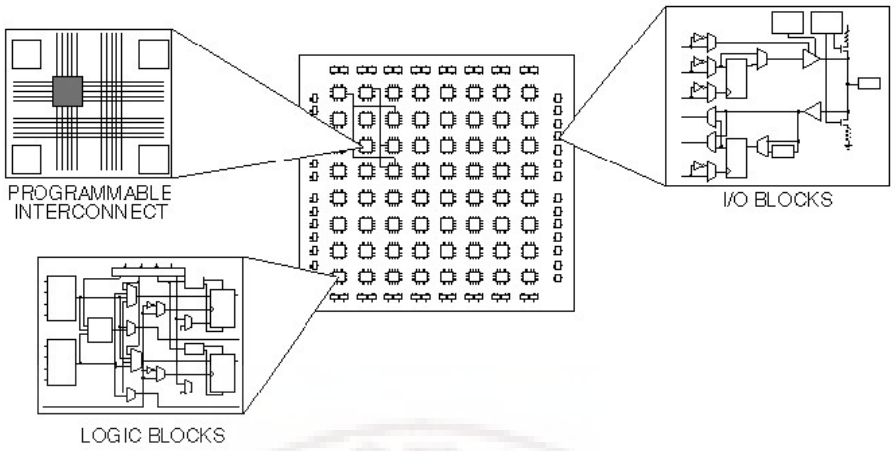


Figure 1.1 Internal Architecture of FPGA

FPGAs are often used to prototype ASIC design or to provide a hardware platform on which to verify the physical implementation of new algorithms. However, their low development cost and short time-to-market are increasingly finding their way into final products (some of the major FPGA vendors actually have devices that they specifically market as competing directly against ASICs).

Now a days FPGAs are becoming popular because of its wide application in different fields quite a few are communications, digital signal processing (DSP), instrumentation and SOC design

1.6.1 Advantages of FPGA over Digital Signal Processors (DSP)

1. Recent FPGAs have included Multipliers especially for performing DSP tasks more efficiently. – For example, a 1M-gate Vertex-II™ device has 40 multipliers that can operate at more than 100MHz. In comparison with the DSP this gives 4000M multiplies per second [4].
2. DSP is well suited to extremely complex math's-intensive tasks, with conditional processing. It is limited in performance by the clock rate, and the number of useful operations it can do per clock. As an example, a TMS320C6201 has two multipliers and a 200MHz clock – so can achieve 400M multiplies per second [4].

3. When sample rates grow above a few MHz, a DSP has to work very hard to transfer the data without any loss. This is because the processor must use shared resources like memory busses, or even the processor core which can be prevented from taking interrupts for some time. An FPGA on the other hand dedicates logic for receiving the data, so can maintain high rates of I/O [4].

1.6.2 Limitations of FPGA

1). Low speed compared to ASIC

2). Limited resources

3). High power consumption than ASIC

1.7 Tools Used

1.7.1 Xilinx ISE

ISE (Integrated Software Environment) design suite is the unification of various Xilinx's design tools (software) for logic embedded and DSP applications. It is a powerful yet flexible integrated design environment that allows us to design Xilinx FPGA and CPLD devices from design entry to implementation it includes a user interface called as project navigator that helps us to manage the entire design process including design entry, simulation, synthesis, implementation, and finally download the configuration into FPGA device.

1.7.2 Xilinx System Generator (Sysgen)

The Xilinx System Generator for DSP bridges the gap between the high-level abstract version of a design and its actual implementation in a Xilinx FPGA. The System Generator for DSP developed in partnership with Math Works Inc. enables designers to develop high-performance DSP systems for Xilinx FPGAs using the popular MATLAB/Simulink. New capabilities include estimation of FPGA resources from within Simulink, HDL Co-Simulation, Hardware in the loop simulation and real time debugging using chip scope pro as shown in figure 1.2.

Figure 1.2 Performing FPGA Hardware in the Loop Simulations with Simulink

1.8 Thesis Structure

This section briefly outlines the contents of the chapter. The dissertation is organized into four chapters. A list of all reference sources consulted is given chapter wise.

- **Chapter 1** is an introductory chapter describing technical approach for modulation classification and extraction of parameter of LPI radar signals and the motivation behind solving such a problem. Thesis scope is also included in this chapter. It gives the brief introduction to HIL simulation, FPGAs and the tools used for the work.
- **Chapter 2** gives an introduction to different LPI radar signals, presents complete procedure for classification of LPI radar signals using WVD. The results are also included along with extracted signal parameters.
- **Chapter 3** presents the complete procedure for implementation of FFT into FPGA and its HIL simulation using sysgen. The resources utilized/ consumed by this implementation are also included along with the timing reports.
- **Chapter 4** Conclusions and future efforts needed for this work are drawn in this Chapter.

Appendix A gives the procedure for creation of Kernel black box and implementation of FFT into FPGA

References

[1] Clive Max Maxfield *The Design Warrior's Guide to FPGA*, Elsevier

[2] Jen-Yu Gau "Analysis of Low Probability of Intercept Radar Signals using Wigner Distribution," Sep 2002.

[3] E. R. Zilberman, P.E. Pace "Autonomous Time Frequency Morphological Feature Extraction Algorithm for LPI Radar Modulation Classification"

[4] www.hunteng.co.uk/info/fpga-or-dsp.htm

[5] XILINX ISE®, version 9.1

[6] XILINX SYSTEM GENERATOR, version 9.1

[7] zone.ni.com/devzone/cda/tut/p/id/3567

[8] Time Frequency Analysis for Single Channel Applications, John Saunders ,High Performance Embedded Computing (HPEC) Conference September 30, 2004

CHAPTER 2
MODULATION CLASSIFICATION USING WVD

2.1 LPI RADAR SIGNAL

On the modern battlefield, radar systems face serious threats from EA (Electronic Attack) and ARMs (Anti Radiation Missiles) due to the lack of efficient technology for detecting the signal sent by the emitter. To perform the necessary target detection of enemy , tracking and simultaneously hide themselves from an enemy attack, user radar systems should be Low Probability of Intercept (LPI) [1]. Low probability of intercept radars has the property of low peak power, wide bandwidth, frequency variability, Continuous Wave (CW) radiation with a large time bandwidth product, and other design features [3]. This features of LPI radar signals makes it difficult to be detected or identified by passive intercept receiver devices such as Electronic Support (ES) or Electronic Intelligence (ELINT) receivers.

LPI radars attempt to provide detection of targets at a longer range than intercept receivers. The success of LPI radars is measured by the degree of difficulty to be detected by other user.

The transmitter makes use of sophisticated frequency and phase modulation to spread the signal bandwidth making the signal hard to intercept. The receiver makes use of the appropriate matched filter so that the radar performance is similar to that of traditional pulsed radar radiating the same amount of average power.

The different modulation signals commonly used in LPI radars are Binary Phase Shift Keying (BPSK), Frequency Modulation Continuously Wave (FMCW), Polyphase code modulation (Frank, P1, P2, P3, and P4), COSTAS frequency hopping, and PSK/FSK signals etc., [3].

2.2 Different Types of LPI Radar signals

2.2.1 Binary Phase Shift Keying (BPSK)

Phase shift keying (PSK) is an angle-modulated, constant-amplitude digital modulation technique. PSK is similar to conventional phase modulation except that with PSK the input signal is a binary digital signal and a limited number of output phases are possible.

Binary phase shift keying (BPSK) has two phases for a single carrier frequency ("binary" meaning "2"). One output phase represents logic 1 and the other is logic 0. As the input digital signal changes state, the phase of the output carrier shifts between two angles that are 180° and 0° with respect to input carrier. BPSK is a form of suppressed-carrier, square-wave modulation of a continuous wave (CW) signal.

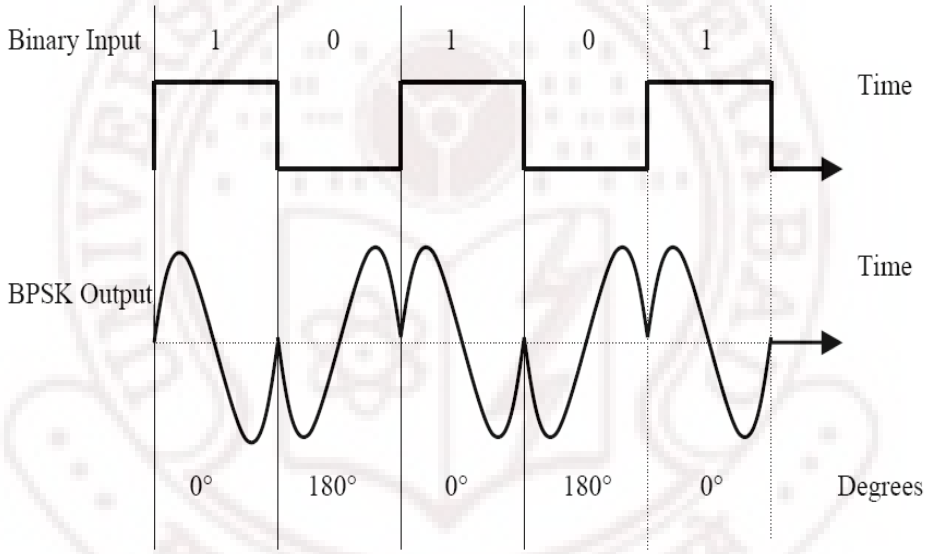


Figure 2.1 BPSK Modulation output phase vs. Time relation ship.

Barker Codes

The Barker codes are a sequence of bits used to generate the BPSK signal, table 2.1 lists the different types of barker sequences available for the generation of BPSK signal, Figure 2.2 illustrates this concept for a Barker code of length seven.

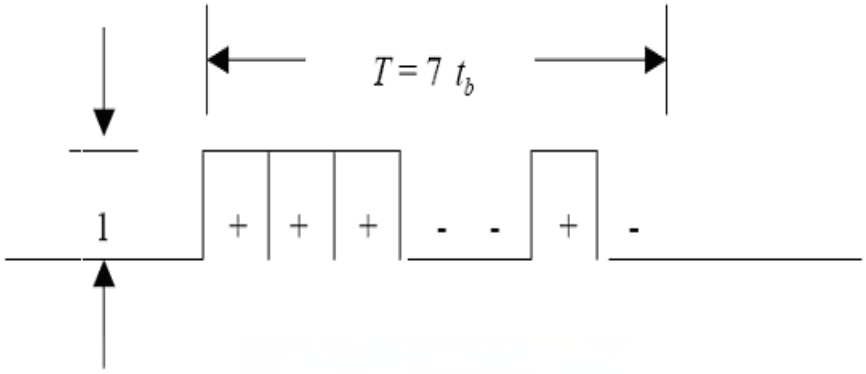


Figure 2.2 Barker code of length 7 with 7 equal pulse widths.

Table 2.1 Barker codes

Code Length	Code Elements	Sidelobe Level, dB
2	+ -, + +	-6.0
3	+ + -	-9.5
4	+ + - +, + + + -	-12.0
5	+ + + - +	-14.0
7	+ + + - - + -	-16.9
11	+ + + - - - + - - + -	-20.8
13	+ + + + + - - + + - + - +	-22.3

+ sign indicates the high level at the input and – indicates the low level of the input.

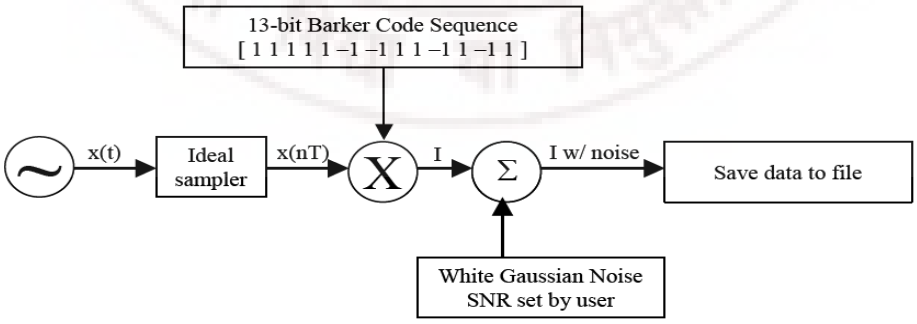


Figure 2.3 BPSK Transmitter Block Diagram.

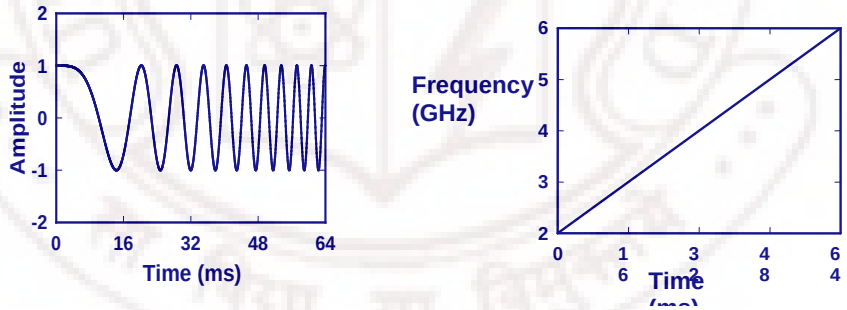
Figure 2.3 shows a functional block diagram of a BPSK transmitter. A sinusoidal carrier is phase shift modulated with a message signal that takes the 13 bit Barker sequence as

shown in the block diagram (figure 2.3). The modulated signal is then corrupted with channel distortion, which is assumed to be white Gaussian noise.

2.2.2 Frequency Modulated Continuous Wave (FMCW)

FMCW modulation technique is used for obtaining refined range information from radar by frequency modulating a continuous signal. The FMCW modulation is readily compatible with a wide variety of solid-state transmitters. The frequency measurement of FMCW, which is performed to obtain range measurement, can be performed digitally using the FFT. Due to their low power and wide bandwidth the FMCW signals are difficult to detect with conventional intercept receivers.

The Linear FMCW technique varies frequency in a sawtooth fashion as shown in figure 2.4 (a), whereby frequency values ramp up over the course of a modulation period as shown in figure 2.4 (b) and then dropped to the starting frequency to begin the ramp up for the next modulation period. Several repetitions of this sequence produce a sawtooth shape in the frequency vs. time plot. By processing the reflected waveform, one can extract range information about the target.



(a) (b)

Figure 2.4 (a) linear FMCW modulation signal (b) Frequency varies linearly with time

Figure 2.5 shows a functional block diagram of a linear FMCW transmitter. A sinusoidal carrier is frequency-modulated with a message signal that takes the triangular shape as shown in the block diagram (figure 2.5). The modulated signal is then corrupted with channel distortion, which is assumed to be additive white Gaussian noise (AWGN).

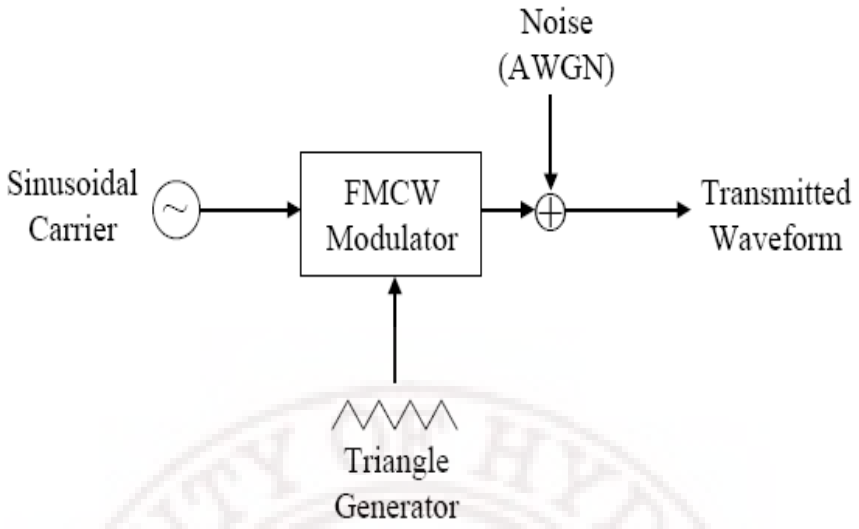


Figure 2.5 Block diagram of FMCW

2.2.3 Polyphase code modulation

Codes that use harmonically related phases based on fundamental phase increment are called poly-phase codes [7]. In this case, a single pulse of width τ is divided into N (code length) equal groups, the phase within each sub-pulse is held constant with respect to carrier wave. The general polyphase equation is given as

$$x(t) = A\cos(\frac{\phi(t)}{N}) \tag{2.1}$$

Where $x(t)$ is polyphase modulated signal and $\phi(t)$ is phase of the input carrier.

The polyphase codes including Frank code, P1 code, P2 code, P3 code, and P4 code

2.2.3.1 Frank code

Frank codes are a family of polyphase codes that are closely related to the chirp and have been used successfully in LPI radars. The Frank code has a code length of N^2, i.e., each pulse is divided into N^2 equal group, the phase of each group is held constant, N^2 is corresponding pulse compression ratio. If i is the number of the sample in a given frequency and j is the number of the frequency, the phase of the ith sample of jth frequency is [7]

$$\phi_{i,j} = 2\Pi(i-1)(j-1). \quad (2.2)$$

Where i=1:*N* and j=1: *N*.

2.2.3.2 P1 code

In case of a double sideband detection (local oscillator is at band center) of a step approximation of a linear frequency modulation, which is the P1 code. The P1 code also consists of N^2 elements. If *i* is the number of the sample in a given frequency and *j* is the number of the frequency, the phase of the *i*th sample of *j*th frequency is [7]

$$\phi_{i,j} = -\frac{\Pi}{N}[N-(2j-1)][(j-1)N+(i-1)]. \quad (2.3)$$

Where i=1: *N* and j=1: *N*.

2.2.3.3 P2 code

The P2 code also consists of N^2 elements. If *i* is the number of the sample in a given frequency and *j* is the number of the frequency, the phase of the *i*th sample of *j*th frequency is [7]

$$\phi_{i,j} = -\frac{\Pi}{2N}(2i-1-N)(2j-1-N). \quad (2.4)$$

Where i=1: *N* and j=1: *N*.

2.2.3.4 P3 code

In the case of linear frequency modulation waveforms, the conceptual coherent detection using single sideband detection and sampling at the Nyquist rate yield a polyphase code call the P3. The phase of the *i*th sample of the P3 code is given by [7]

$$\phi_i = \frac{\Pi}{N}(i-1)(i-N). \quad (2.5)$$

Where i=1: *N*.

2.2.3.5 P4 code

The P4 code consists of the discrete phase of the linear chirp waveform taken at specific time intervals and exhibits the same range Doppler coupling associated with the chirp waveform. The peak sidelobe levels are lower than those of the unweighted chirp waveform. The code element of the P4 code are given by [7]

$$\phi_i = \frac{\Pi}{N}(i-1)^2 - \Pi(i-1). \qquad (2.6)$$

Where i=1:*N*.

2.3 Modulation Classification

Several types of Frequency Analysis methods have been developed to classify and analyze the LPI radar signals, such as Fast Fourier Transformation (FFT) to analyze and graphically display the result for user interpretation. Improvements are being considered to provide representations beyond the conventional FFT. The use of Time Frequency Analysis (TFA) provides an accurate analysis of the signal along with parameter extraction and modulation classification.

Among the different types of Time Frequency Analysis (TFA) the Wigner Ville Distribution (WVD) gives best time frequency resolution [12] and also it has advantages of independently controlling the time and frequency resolution. This unique feature is not possible with Short-Time Fourier Transform (STFT). The disadvantage with STFT is that it is less accurate than WVD for a non stationary signal.

The WVD is a two-dimensional analysis describing the frequency content of a signal as a function of time. It calculates the instantaneous correlation of the input signal and finds the fourier transformation of the obtained instantaneous correlation function.

2.4 Time Frequency Analysis

Time-frequency analysis (TFA) is one of the techniques used for characterizing signals whose frequency components vary in time, such as transient signals. Whereas the technique of the frequency analysis (for e.g. Fourier transform) is used to obtain the frequency spectrum of stationary signals, such a technique is not appropriate for analyzing a signal that has time-varying frequency components [1]-[2].

In order to analyze a signal whose component frequencies vary in time, one first obtains a time-frequency distribution of the signal, which represents the signal in both the time and frequency domains simultaneously. There are five TFA techniques.

1) Short-Time Fourier Transform (STFT) for a discrete data is '*x(l)*' is defined as:

$$STFT(l,k) = \sum_{n=-\infty}^{\infty} w_n(l-n).x(l).e^{-2\Pi jkn/N} \qquad (2.7)$$

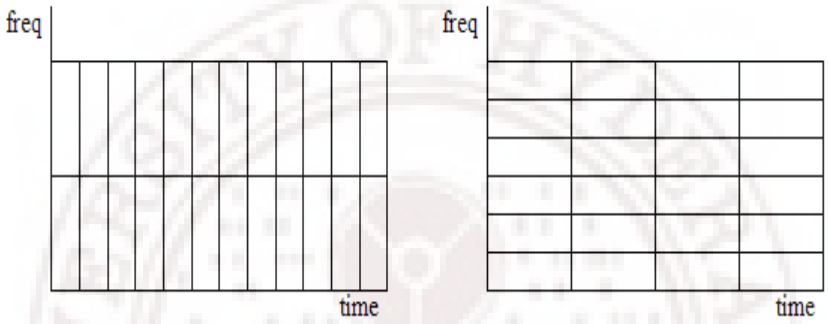


Figure 2.6 Comparison of STFT resolutions.

Estimating of the TFA using STFT involves computing a Fourier Transform of a discrete time series '*x (l)*' using a sliding window 'w_n'. At each time index '*l*', '*N*' point fourier transform (FT) is computed, giving a spectral estimate at the frequency index '*k*'. STFT is a sliding windowed FT technique. Reducing '*tw*' (width of sliding window) produces a reduction in frequency resolution and an increase in time resolution. Increasing '*tw*' has the opposite effect. It should be noted that the frequency and time resolution depends on the window length '*tw*' and not on the frequency. For simplicity a rectangular window is used here but other "gentler" windows also can be used to produce a spectrum with smaller sidelobes.

WVD is on of the TFA technique where the time and frequency resolution can be independently controlled detailed study of WVD is presented in Sec 2.5

2) Wigner Distribution of a continuous signal '*x(t)*' is defined as:

$$WD(t,\omega) = \int_{\tau=-\infty}^{\infty} x(t+\tau/2).x^*(t-\tau/2).e^{-j\omega\tau}\,d\tau \qquad (2.8)$$

Where 'ω' is angular frequency $2\pi f$, the integral is from $-\infty$ to ∞, and the * indicates the complex conjugate of the signal '*x(t)*', ' τ ' is correlation time interval.

3) Wigner Ville Distribution of a discrete signal '*x(l)*' is defined as:

$$W(l,\omega) = 2\sum_{n=-N}^{N-1} x(l+n)x^*(l-n)e^{-j2\omega n} \qquad (2.9)$$

Where '*n*' is the correlation interval, '*l*' discrete sampling time interval, ' ω ' is angular frequency $2\pi f$, *'N'* is window length. The WVD offers very good time and frequency resolution compared with the Spectrogram of STFT.

4) Smoothed Wigner Distribution (SWD) for a discrete data '*x*(*l*)' is defined as:

$$SWD(l,k) = 2\sum_{n=-N}^{N-1} w_N(n).\{\sum_{m=-M}^{M-1} w_M(m)x(l+m+n).x^*(l+m-n)\}.e^{-2\Pi jkn/N} \qquad (2.10)$$

Where ' w_N ' and ' w_M ' are two different windows of length *2N* and *2M*, '*m*' and '*n*' are correlation time intervals, '*k*' represents the discrete frequency and '*l*' is the sampling time interval of '*x(l)*'.

The discrete time WVD is periodic in π, but for SWD the discrete time is periodic in 2π. So unless the signal '*x*(*l*)' has been sampled at least twice the Nyquist rate, there will be aliasing in the WD. This problem can be solved by sampling the signal at twice the Nyquist rate of '*x* (*l*)'.

5) Choi-Williams Distribution (CWD) for a discrete data *x*(*l*) is defined as:

$$CWD(l,k) = 2\sum_{n=-N}^{N-1} w_N(n).\{\sum_{m=-M}^{M-1} w_M(m)\sqrt{\sigma/4\Pi n^2}.e^{-\frac{\sigma m^2}{4\Pi^2}} * x(l+m+n).x^*(l+m-n)\}.e^{-2\Pi jk\frac{n}{N}} \qquad (2.11)$$

Where ' w_N ' and ' w_M ' are two different windows of length *2N* and *2M*, '*m*' and '*n*' are correlation time intervals, '*l*' is the sampling time interval of '*x(l)*' and '*k*' represents the

discrete frequency, 'σ' represents the attenuation factor. The parameter 'σ' can be adjusted to attenuate cross-terms. The CWD is a further refinement of the SWD, in which extra smoothing is achieved with an exponential weighting function. Lower values of 'σ' attenuate cross-terms but degrade time and frequency resolution [4]. For large values of 'σ' , cross-terms are not suppressed at all, and the CWD becomes equivalent to the SWD. A good choice of 'σ' is in the range 0.1 to 10.

2.4.1 Comparison of TFA techniques

The time and frequency resolution of STFT cannot be controlled independently; using WVD the time and frequency resolutions can be independently controlled. The limitations of discrete WVD are the presence of cross term. The cross terms in WVD can be reduced by using the SWD and CWD techniques [1]-[2] and are computational intensive.

2.5 Wigner Ville Distributions (WVD)

The discrete Wigner Ville Distribution is a two dimensional Time Frequency analysis of input (Modulated) signal as shown in figure 2.7. The WVD calculates the instantaneous correlation of the input signal and then computes Fourier Transformation on the resulting signal after correlation [1]-[2].

Let '*x(n)*' be a complex discrete time analytic signal with Fourier Transform '*X(f)*' then the discrete WVD distribution is defined as

$$W(l,\omega) = 2\sum_{n=-N}^{N-1} x(l+n)x^*(l-n)e^{-j2\omega n} \qquad (2.12)$$

Where '$x^*(n)$' is complex conjugate representation of '$x(n)$', 'l' is discrete time index and 'ω' gives the frequency information

The function $x(l+n)x^*(l-n)$ is used to calculate the instantaneous correlation of $x(n)$ and is represented by '$f(l,n)$' known as Kernel function, i.e.,

$$f(l,n) = x(l+n)x^*(l-n) \qquad (2.13)$$

Different types of Frequency Analysis methods have been developed to analyse the LPI radar signals which use conventional Fast Fourier Transformation (FFT) to analyze

and graphically display the result for user interpretation. Improvements are being considered to provide representation beyond the conventional FFT. The use of Time Frequency Analysis (TFA) along with classification and extraction of different parameters of modulated signal provide an accurate analysis and intercept of unknown modulated signals in real time.

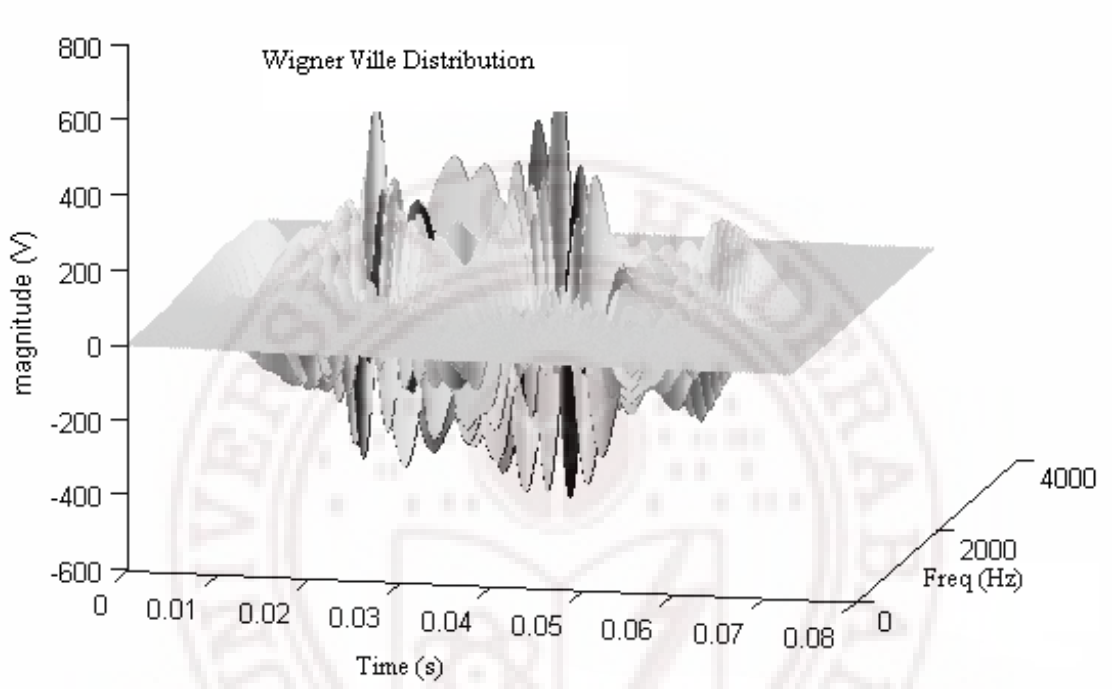


Figure2.7 WVD of Frequency Modulated Continuous Wave signal.

2.5.1 Disadvantages of WVD

The disadvantage of WVD is due to the presence of cross terms, it can be avoided by using CWD, the implementation of CWD is computational intensive [3] and consumes a lot of resources to implement into FPGA.

2.6 Classification of LPI Radar signals

The classifier algorithm examines the unique features between different types of LPI Radar signals and it determines the type of modulation technique used. The classification algorithm includes thresholding, Binarization or Normalization, Cropping and Classification as shown in figure 2.8.

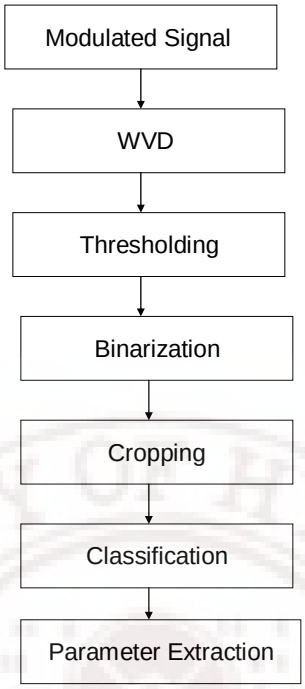
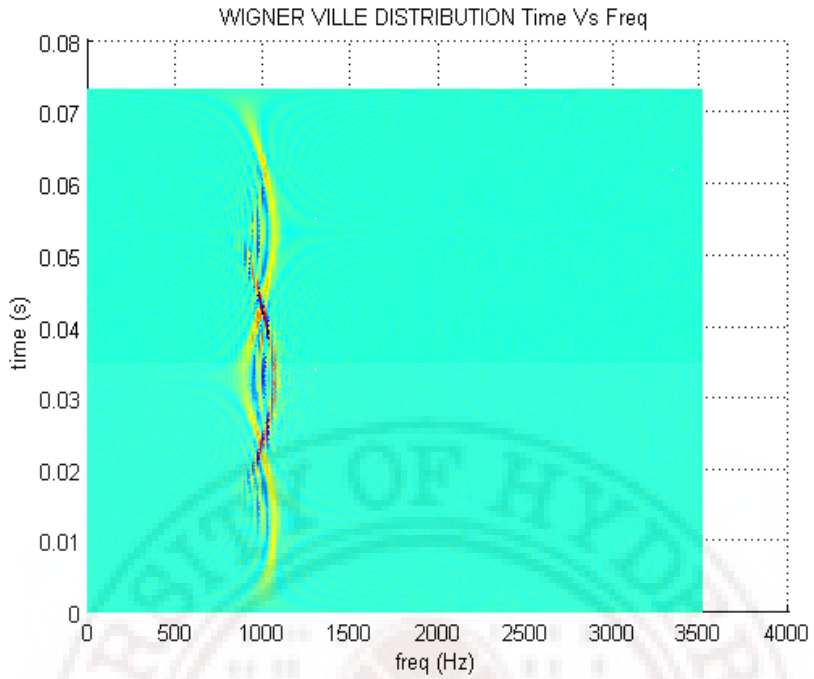


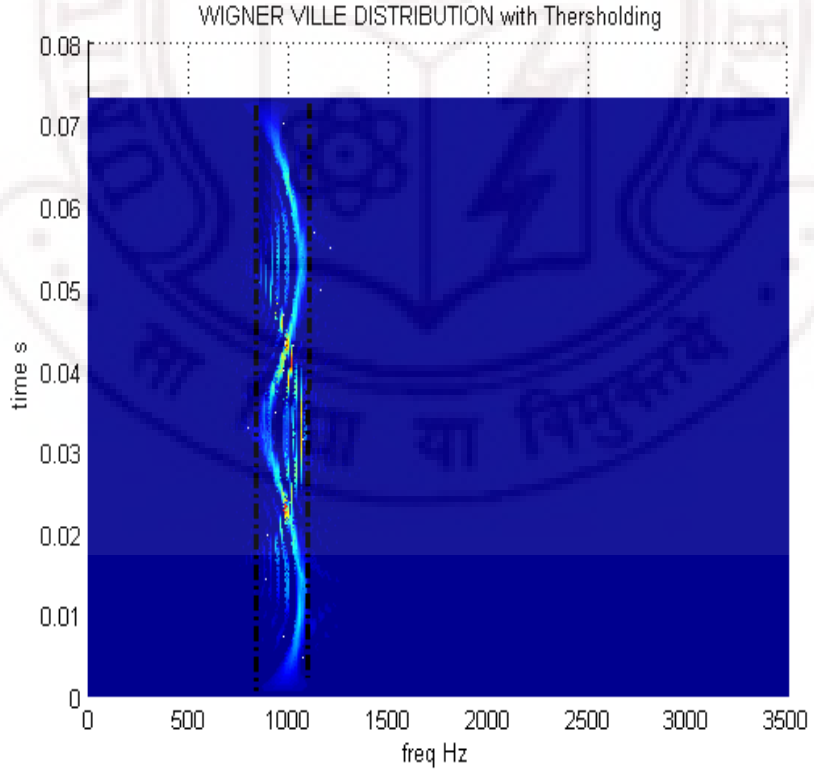
Figure 2.8 Flow chart of Classification Algorithm

2.6.1 Thresholding

Thresholding is a technique used to limit the signal values to a defined range of amplitude, specified by the threshold value. The key parameter in the thresholding process is the choice of the threshold value and its value depends on the peak value of WVD amplitude. The figure 2.9 (a) shows the Wigner Ville distribution of FMCW before thresholding and figure 2.9 (b) is obtained by thresholding the WVD of FMCW. Thresholding is one of the key procedures for classification of LPI radar signals.



(a)



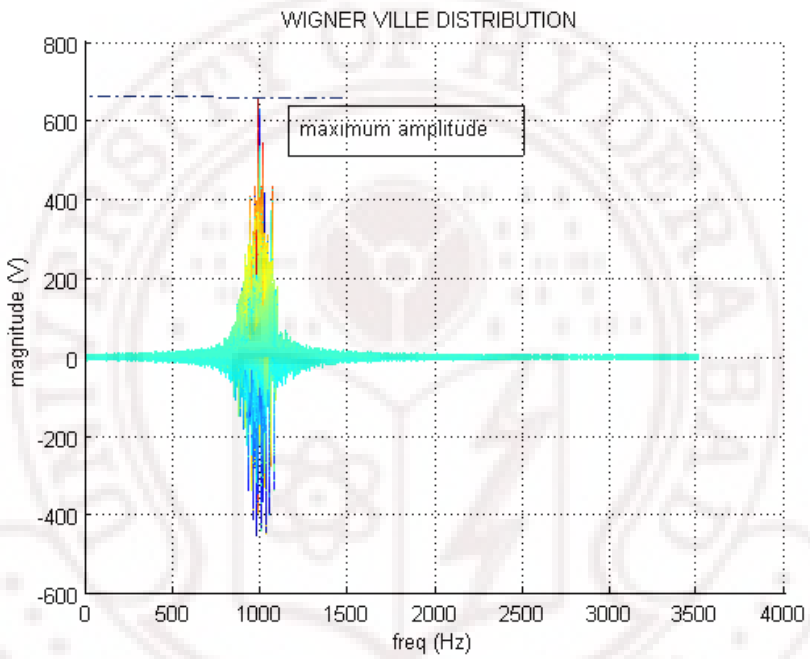
(b)

Figure 2.9WVD of FMCW (a)before and (b) after thresholding

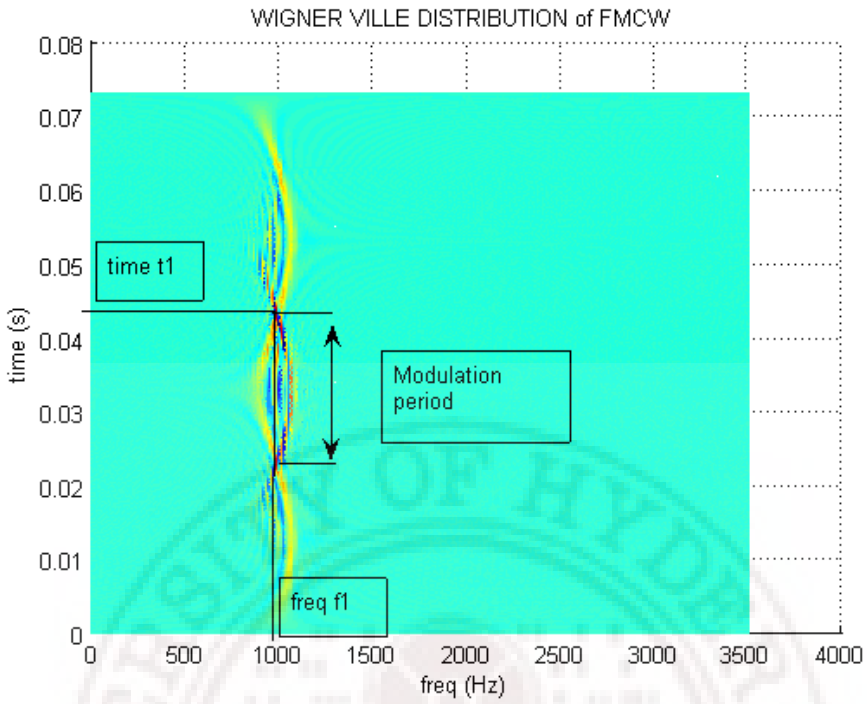
2.6.1.1 Procedure to find Threshold value

The First step in thresholding process is to find the threshold value. The threshold value is taken to be half the maximum value (3 dB) of WVD plot. Figure 2.10 (a) shows the maximum value of WVD plot and figure 2.10 (b) shows the time and frequency coordinates (t1,f1) of the peak value.

The next step in thresholding is to discard the signal whose value is below the threshold level and make them to zero as shown in figure 2.11.



(a)



(b)

Figure 2.11 WVD of FMCW (a) magnitude vs frequency (b) time vs frequency

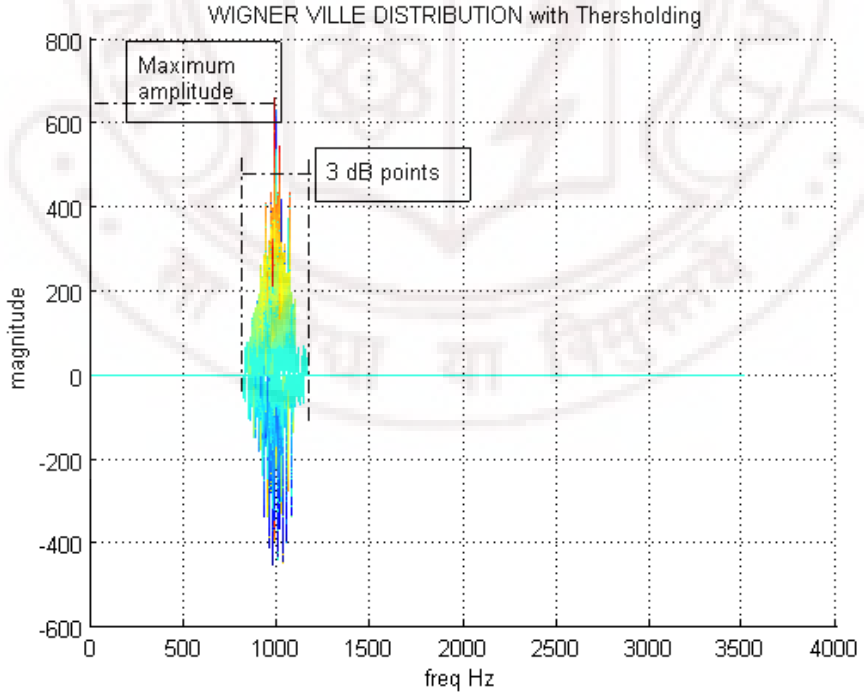


Figure 2.12 WVD plot of FMCW after thresholding time vs frequency

2.6.2 Binarization

Binarization is a technique used to represent the WVD plot in terms of zeros and ones, the signal value greater than threshold value is represented by '1' and value less than threshold value by '0'. The figure 2.12 (a) below shows time vs frequency plot of the Binarization of FMCW obtained after thresholding.

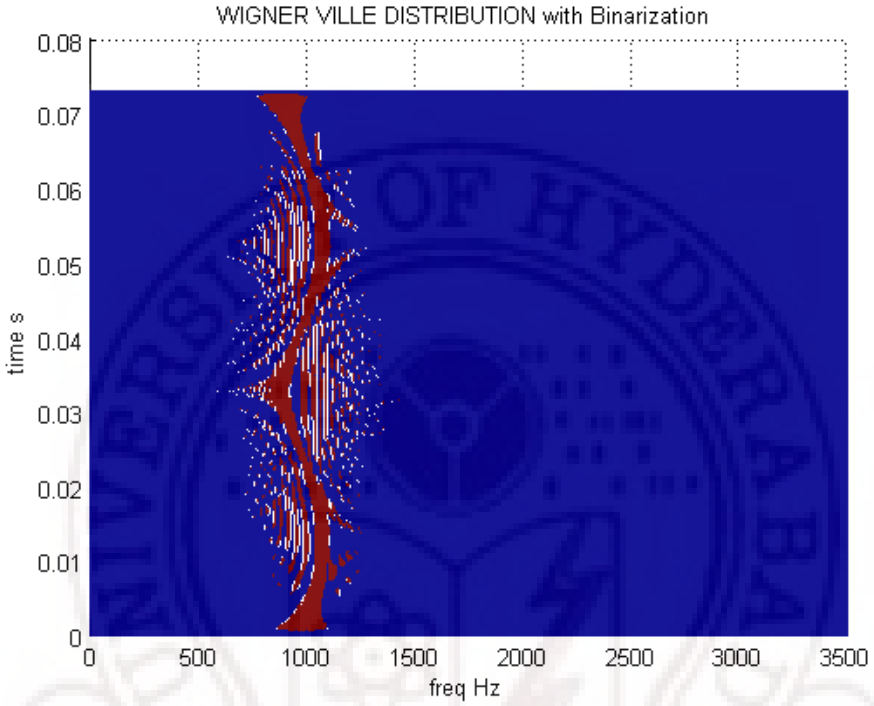


Figure 2.12 (a)

Binarization process reduces the computations involved in the cropping stage by representing the signal with 2 levels ('0' and '1'). The figure 2.12 (b) below shows magnitude vs frequency plot of the Binarization of FMCW obtained after thresholding

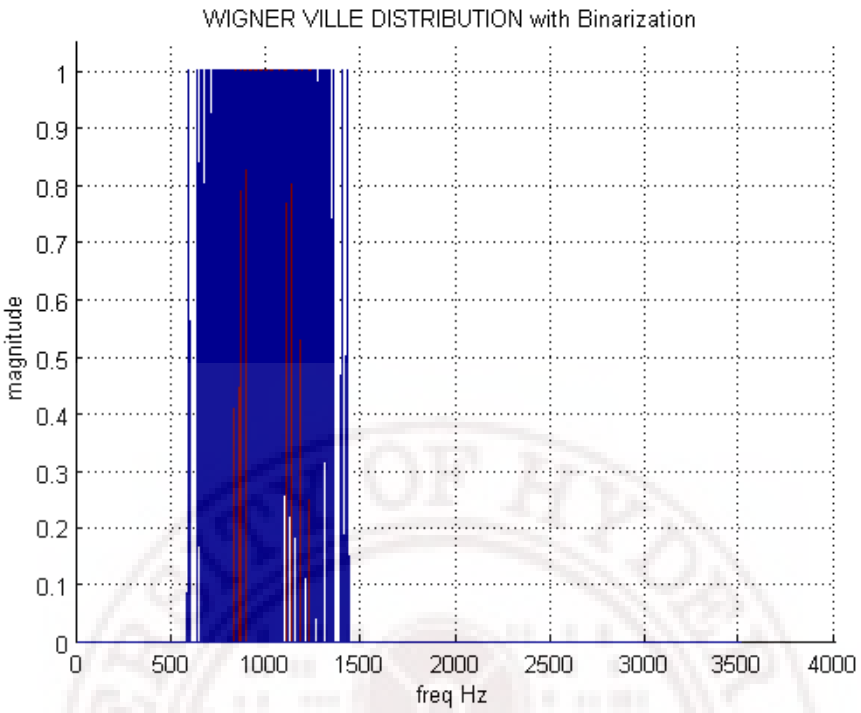


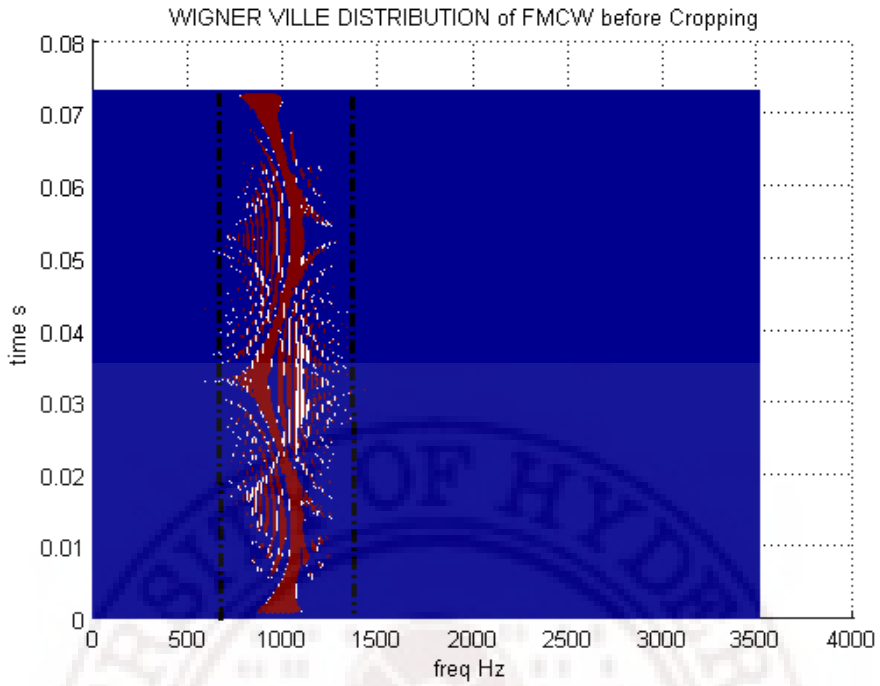
Figure 2.12 (b)

Figure 2.12 WVD of FMCW with Binarization (a) time vs frequency (b) magnitude vs frequency

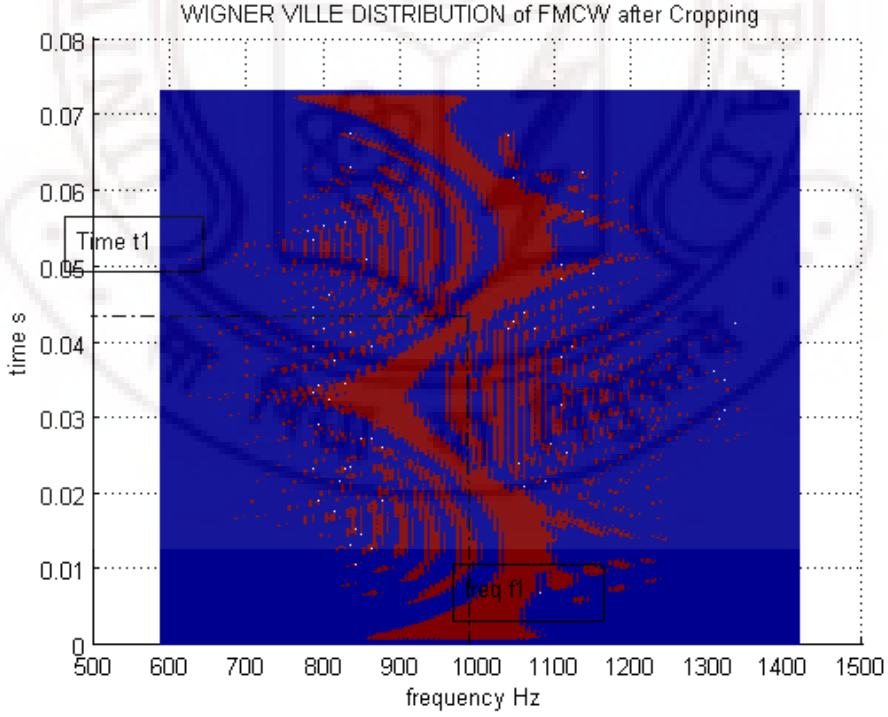
2.6.3 Cropping

Cropping refers to the removal of an unwanted and irrelevant signal details from the graph (image) to improve aspect ratio. It represents the part of the graph where the most of the signal contents are present

The figure 2.13 (b) shows the WVD plot of FMCW obtained after cropping. The plot contains only the part of graph where most of the signal content is present. Cropping reduces the computation involved during classification by reducing the graph area.



(a)



(b)

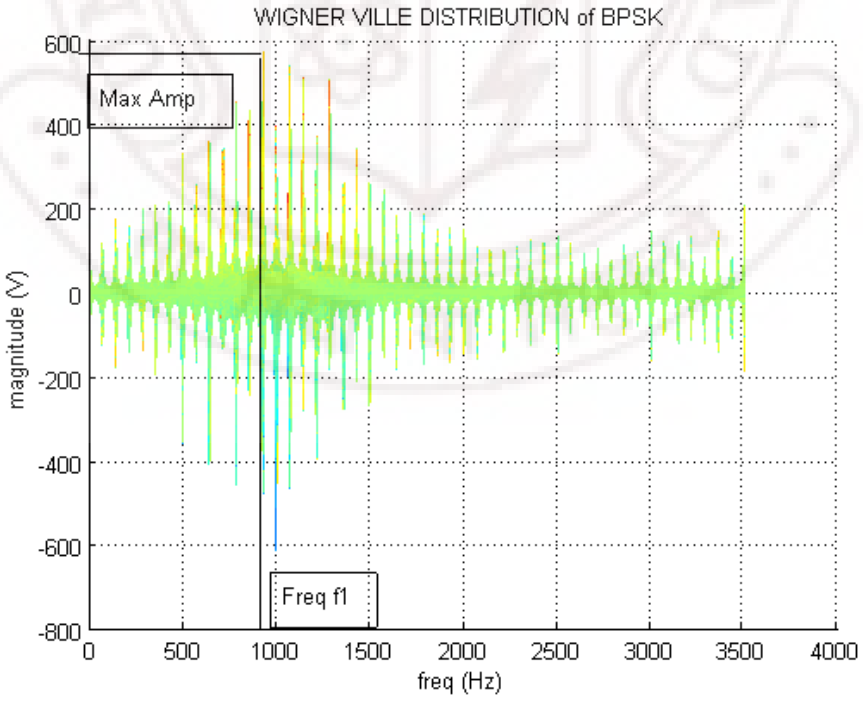
Figure 2.13 WVD of FMCW (a)before and (b) after Cropping

2.6.4 Classification

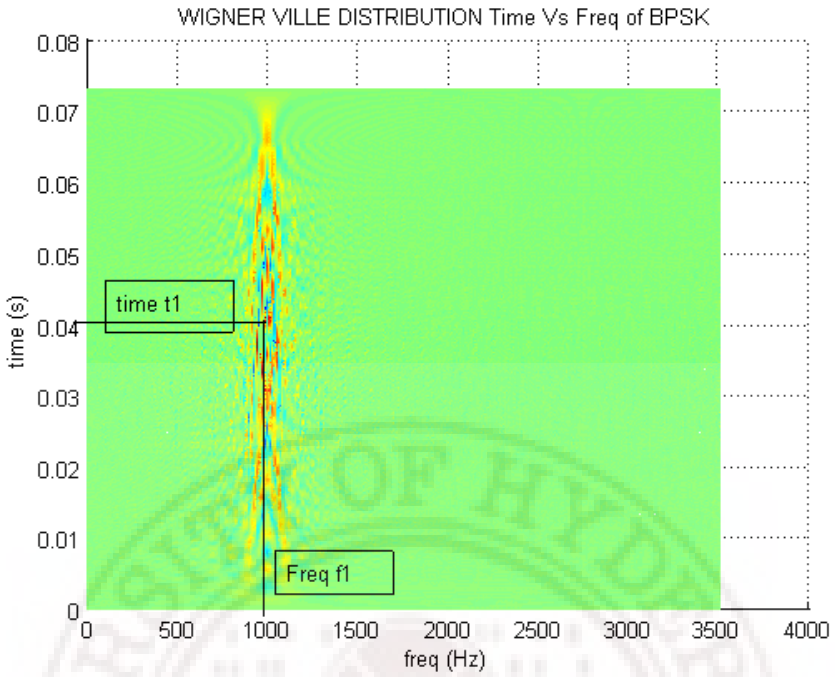
The classification process is to identify the unique features of the different modulation signal and classify them accordingly. The classification process involves series of image processing and it is computational immense than Thresholding, Binarization and Cropping. The classification of various BPSK and FMCW modulated signal using this algorithm are listed in table 2.2 and 2.3.

2.6.4.1 Classification Procedure

The unique features that distinguish FMCW and BPSK signals are i) WVD of BPSK signal have peaks at regular frequency intervals as shown in figure 2.14 (a). ii) The number of frequency components present at time interval t1 in WVD of FMCW as shown in figure 2.10 (b) are less than WVD of BPSK at time interval t1 as shown in figure 2.14 (b). and iii) the sum of frequency components increases if we move above or below the time interval t1 for WVD of FMCW as shown in figure 2.10 (b). BPSK and FMCW are classified using above three distinguish features between WVD output of FMCW and BPSK signals.



(a)



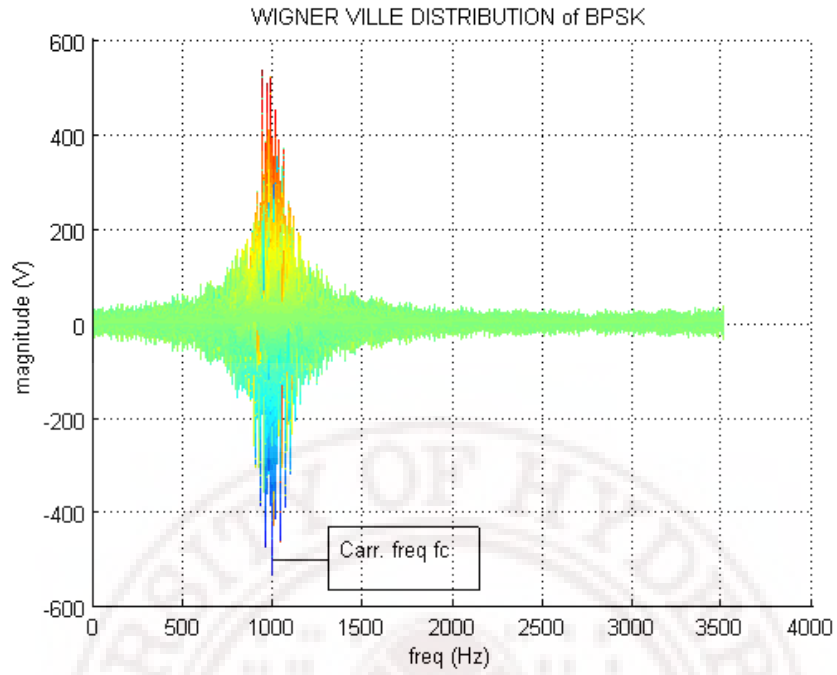
(b)

Figure 2.14 WVD plot of BPSK (a) magnitude vs frequency (b) time vs frequency

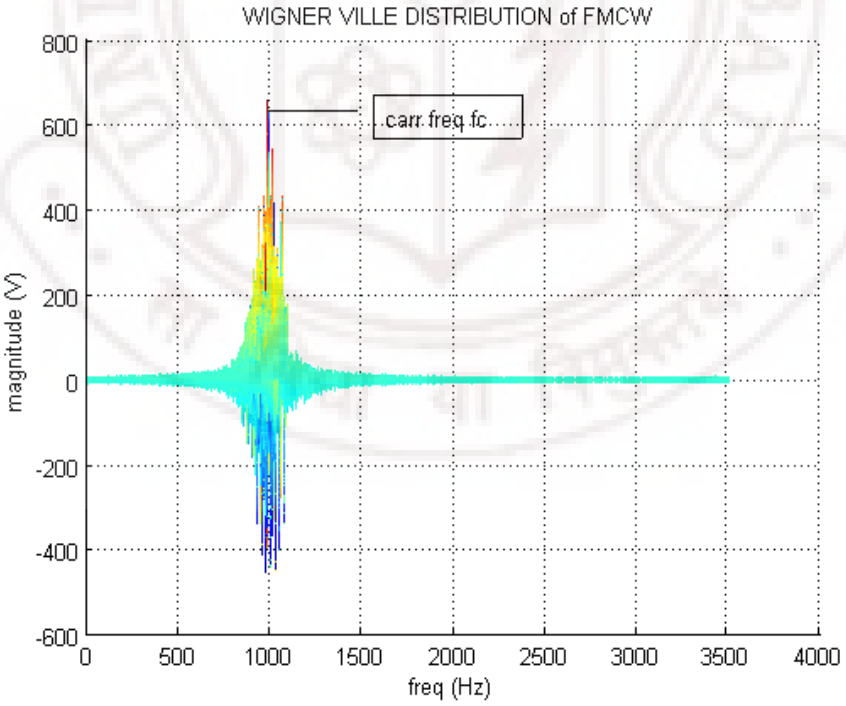
2.7 Parameter Extraction

After the classification of LPI signals, the different parameters of the modulated signals are extracted in this process. Varying the different parameters of BPSK and FMCW modulated signals are studied and tabulated in table 2.2 and 2.3. The process of extracting different parameters of BPSK and FMCW is explained below.

The carrier frequency of BPSK signal is the minimum amplitude frequency component in the WVD as shown in figure2.15 (a) and for FMCW it is maximum value frequency component in the WVD as shown in figure 2.15 (b). The Bandwidth for both BPSK and FMCW are the 3dB points in WVD as shown in figure 2.16 (b) and in figure 2.16 (a). The Barker code length for BPSK signal is obtained by counting the number of peaks with in the 3dB bandwidth in the WVD as shown in figure 2.14(a). The code period of BPSK modulated signal is equal to (Barker length/carrier frequency). The extraction of modulation period of FMCW is shown in figure 2.10 (b).

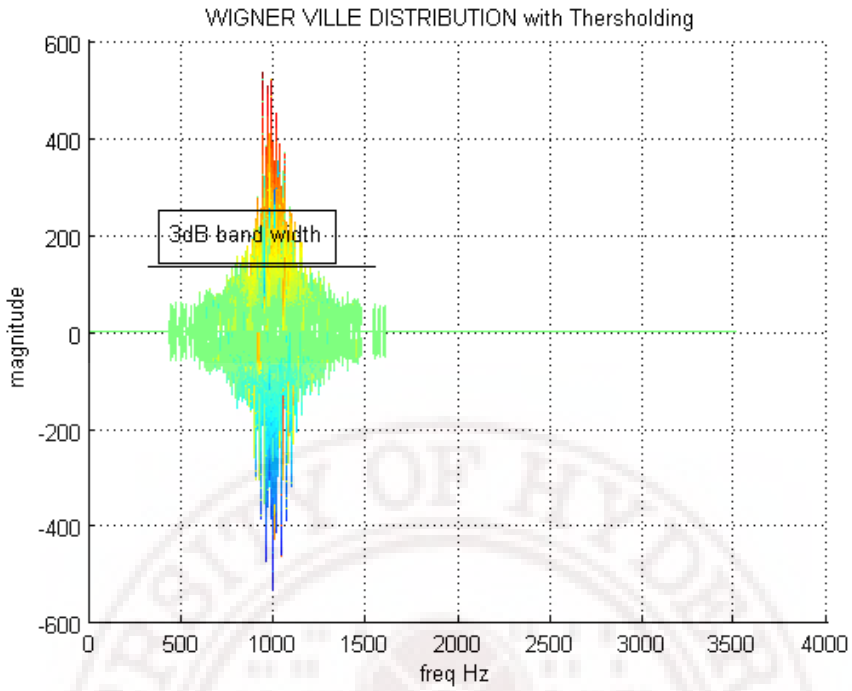


(a)

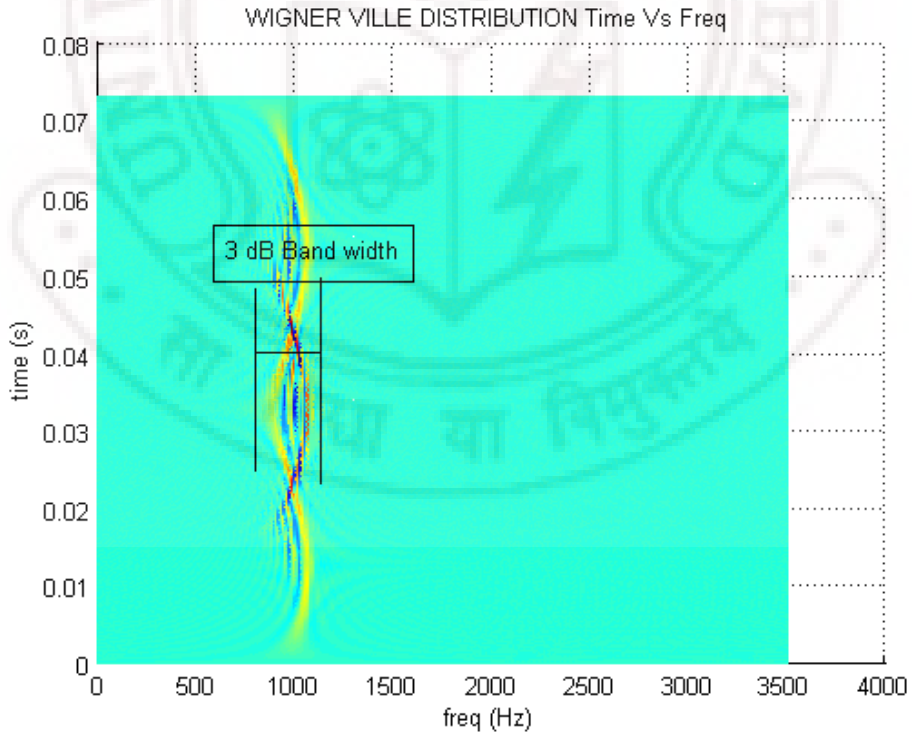


(b)

Figure 2.15 carrier frequency calculation of (a) BPSK (b) FMCW



(a)



(b)

Figure 2.16 Band width calculation of (a) BPSK (b)FMCW

2.8 Classification and extraction of different modulation parameters of Frank, P1, P2, P3, P4 codes

The classification and extraction of modulation parameters of polyphase codes are worked with out WVD algorithm. The unique features to distinguish between different polyphase codes are i) the P2 code has initial non zero phase and other polyphase codes has initial zero phase. ii) the phase length of Frank, P1 code is 'N^2' and for P3 and P4 codes it is 'N' iii) the initial phase for Frank code is zero for 'N' cycles and for P1 code it is less than 'N' cycles iv) the phase after first cycle for P3 code is positive and for P4 code it is negative. The figure 2.17 shows the flow chart for classification and extraction of modulation parameters of polyphase codes. Table 2.4 to 2.8 lists the classification of Polyphase codes along with the parameters.

The different parameters of polyphase codes are carrier frequency, bandwidth, code length and cycles per phases (CPP). The carrier frequency of the polyphase codes is obtained by applying inverse cosine function on the input modulated signal. The CPP is extracted by generating a cosine signal using carrier frequency obtained and subtract it from the modulated signal and the number of non zero terms gives CPP. The band width is given as carrier frequency/ CPP. The code length is extracted by using the inverse cosine transformation on the phase.

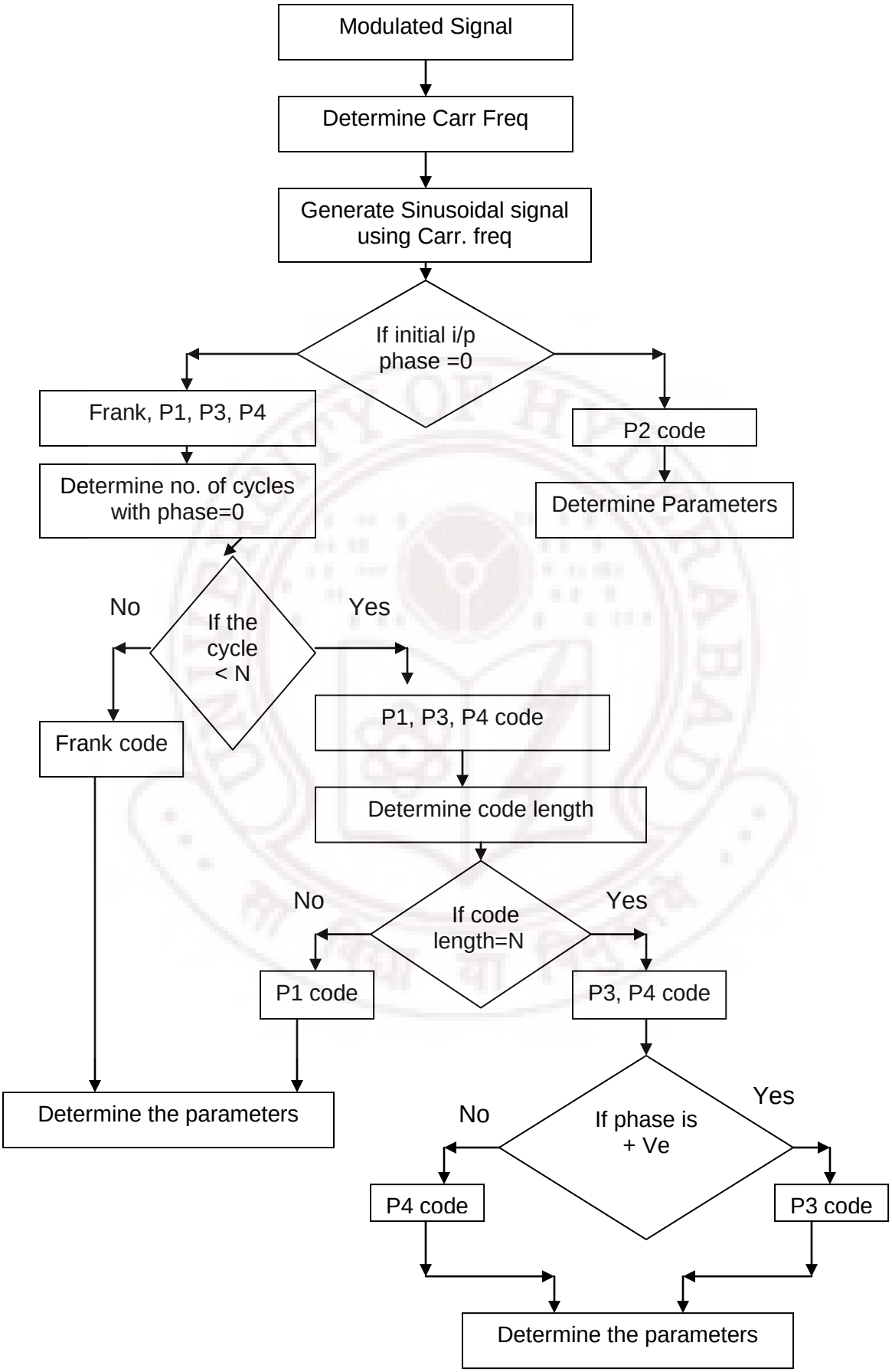


Figure 2.17 Flow chart of Classification and Parameter extraction of Polyphase codes

2.9 Simulation reports and Results

The different types of LPI radar signals are generated using MATLAB are shown in figure 2.18 to figure 2.20 and are applied to the classification and extraction algorithm, the tables [2.2 to 2.8] below lists the modulated input applied to the classification algorithm along with the different parameters of the input signal generated using MATLAB such as carrier frequency, bandwidth etc., and also it lists the output of the classification algorithm along with their extracted parameters. The histograms figure [2.21 to 2.27] shows the percentage estimation of different parameter of LPI radar signals.

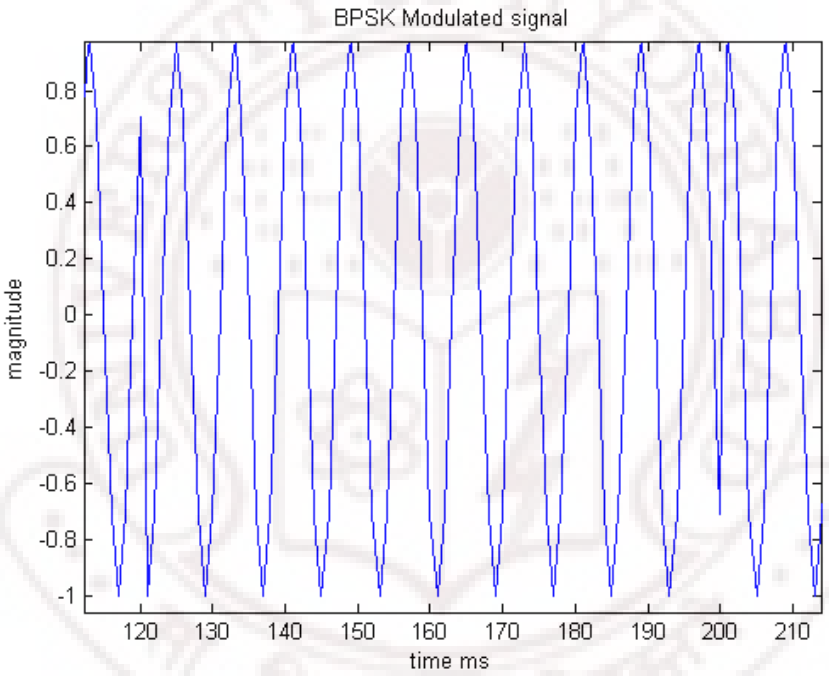


Figure 2.18 BPSK signal with carrier frequency=1000, sampling frequency=8000, barker code length=7, NPBB=5.

The below table 2.2 shows the classification results of BPSK signal using WVD and the parameter extraction for different values of carrier frequency, sampling frequency, bandwidth and Barker code length of the BPSK signal.

Table 2.2 Parameter Extraction of BPSK

Input to classifier					Output of Classifier	Parameter Extraction		
Mod. Type	Sam. Freq (Hz)	Carr. Freq (Hz)	Band Width (Hz)	Barker code/NPBB		Band Width (Hz)	Carr. Freq (Hz)	Barker code/NPBB
BPSK	7,000	1,000	200	7/5	BPSK	184	998.04	7/5
BPSK	10,000	2,000	2,000	7/1	BPSK	2001	2001	7/1
BPSK	10,000	2,000	1,000	7/2	BPSK	996	2001	11/2
BPSK	10,000	1000	1000	7/1	BPSK	1005	927	11/1
BPSK	7,000	1,000	200	11/5	BPSK	170	998	11/5
BPSK	10,000	2,000	2,000	11/1	BPSK	1816	2001	11/1
BPSK	10,000	2,000	1,000	11/2	BPSK	996	2001	11/2
BPSK	10,000	1000	1000	11/1	BPSK	1005	996	11/1
BPSK	16,000	5000	1000	11/5	BPSK	1000	5000	11/5

The below table 2.3 shows the classification results of FMCW signal using WVD and the parameter extraction for different values of carrier frequency, sampling frequency, bandwidth and Modulation period of the FMCW signal.

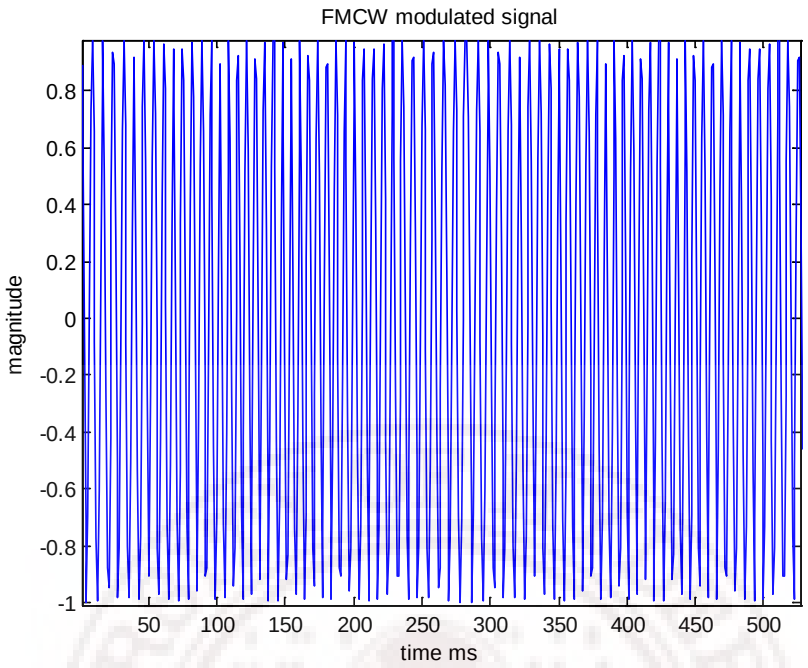


Figure 2.19 FMCW signal with carrier frequency=1000, sampling frequency=7000, BW=250, modulation period=0.02s

Table 2.3 Parameter Extraction of FMCW

Input to classifier					Output of Classifier	Parameter Extraction		
Mod. Type	Sam. Freq (Hz)	Carr. Freq (Hz)	Band Width (Hz)	Mod. Period (sec)		Carr. Freq (Hz)	Band Width (Hz)	Mod. Period (sec)
FMCW	7,000	1,000	250	0.02	FMCW	998	374	0.019
FMCW	7,000	1,000	500	0.01	FMCW	991	540	0.0097
FMCW	7,000	1,000	500	0.02	FMCW	998	499	0.02
FMCW	7,000	2,000	1,000	0.02	FMCW	2016	1093	0.019
FMCW	10,000	3,000	1,000	0.01	FMCW	3017	1103	0.002
FMCW	10,000	2,000	1,000	0.02	FMCW	2001	996	0.02
FMCW	8,000	2,000	500	0.02	FMCW	1968	695	0.016
FMCW	8,000	2,000	1,000	0.01	FMCW	1984	1304	0.03

The below table 2.4 shows the classification results of Frank code and the parameter extraction for different values of carrier frequency, sampling frequency, bandwidth and Code length/ cycles per phase (CPP) of the Frank code.

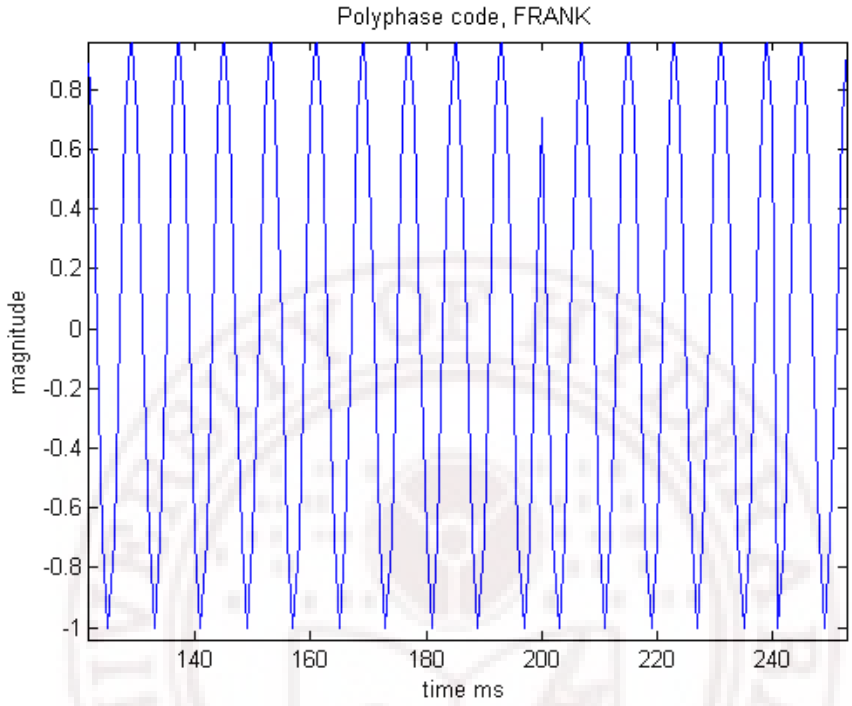


Figure 2.20 Polyphase FRANK signal with carrier frequency=1000, sampling frequency=8000, code length = 16, CPP=5

Table 2.4 Parameter Extraction of Frank

Input to Classifier					Output of Classifier	Parameter Extraction		
Mod. Type	Sam. Freq (Hz)	Carr. Freq (Hz)	Band Width (Hz)	Code length/cpp		Band Width(Hz)	Carr. Freq (Hz)	Code length/cpp
FRANK	7,000	1,000	200	16/5	FRANK	199	998	16/5
FRANK	10,000	2,000	400	25/5	FRANK	388	1940	25/5
FRANK	10,000	2,000	400	49/5	FRANK	393	1968	49/5
FRANK	10,000	1000	200	36/5	FRANK	185	927	36/5
FRANK	7,000	1,000	200	64/5	FRANK	199	998	64/5

FRANK	10,000	2,000	250	36/8	FRANK	248	1988	36/8
FRANK	10,000	2,000	400	36/5	FRANK	400	2001	36/5
FRANK	10,000	1000	200	25/5	FRANK	199	996	25/5
FRANK	16,000	5000	1000	25/5	FRANK	1000	5000	25/5

The below table 2.5 shows the classification results of P1 code and the parameter extraction for different values of carrier frequency, sampling frequency, bandwidth and Code length/CPP of the P1 code.

Table 2.5 Parameter Extraction of P1

Input to Classifier					Output of Classifier	Parameter Extraction		
Mod. Type	Sam. Freq (Hz)	Carr. Freq (Hz)	Band Width (Hz)	Code length /cpp		Band Width (Hz)	Carr. Freq (Hz)	Code length/cpp
P1	7,000	1,000	200	16/5	P1	193	968	16/5
P1	10,000	2,000	400	25/5	P1	396	1980	25/5
P1	10,000	2,000	400	49/5	P1	396	1980	49/5
P1	10,000	1000	200	36/5	P1	189	946	36/5
P1	7,000	1,000	200	64/5	P1	200	998	64/5
P1	10,000	2,000	250	36/8	P1	248	1988	36/8
P1	10,000	2,000	400	36/5	P1	405	2025	36/5
P1	10,000	1000	200	25/5	P1	199	996	25/5
P1	16,000	5000	1000	25/5	P1	998	4990	25/5

The below table 2.6 shows the classification results of P2 code and the parameter extraction for different values of carrier frequency, sampling frequency, bandwidth and Code length/CPP of the P2 code.

Table 2.6 Parameter Extraction of P2

Input to Classifier					Output of Classifier	Parameter Extraction		
Mod. Type	Sam. Freq (Hz)	Carr. Freq (Hz)	Band Width (Hz)	Code length /cpp		Band Width (Hz)	Carr. Freq (Hz)	Code length/cpp
P2	7,000	1,000	200	16/5	P2	199	998	-/5
P2	10,000	2,000	400	25/5	P2	388	1940	-/5
P2	10,000	2,000	400	49/5	P2	393	1968	-/5
P2	10,000	1000	200	36/5	P2	185	927	-/5
P2	7,000	1,000	200	64/5	P2	199	998	-/5
P2	10,000	2,000	250	36/8	P2	248	1988	-/8
P2	10,000	2,000	400	36/5	P2	400	2001	-/5
P2	10,000	1000	200	25/5	P2	199	996	-/5
P2	16,000	5000	1000	25/5	P2	1000	5000	-/5

The below table 2.7 shows the classification results of P3 code and the parameter extraction for different values of carrier frequency, sampling frequency, bandwidth and Code length/CPP of the P3 code.

Table 2.7 Parameter Extraction of P3

Input to Classifier					Output of Classifier	Parameter Extraction		
Mod. Type	Sam. Freq (Hz)	Carr. Freq (Hz)	Band Width (Hz)	Code length /cpp		Band Width (Hz)	Carr. Freq (Hz)	Code length/cpp
P3	7,000	1,000	200	4/5	P3	193	968	4/5
P3	10,000	2,000	400	5/5	P3	396	1980	5/5
P3	10,000	2,000	400	7/5	P3	396	1980	7/5
P3	10,000	1000	200	6/5	P3	189	946	6/5
P3	7,000	1,000	200	8/5	P3	200	998	8/5

P3	10,000	2,000	250	6/8	P3	248	1988	6/8
P3	10,000	2,000	400	6/5	P3	405	2025	6/5
P3	10,000	1000	200	5/5	P3	199	996	5/5
P3	16,000	5000	1000	5/5	P3	998	4990	5/5

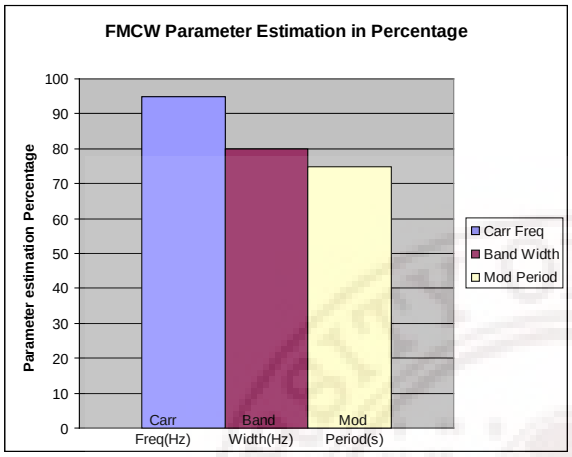
The below table 2.8 shows the classification results of P4 code and the parameter extraction for different values of carrier frequency, sampling frequency, bandwidth and Code length/CPP of the P4 code.

Table 2.8 Parameter Extraction of P4

Input to Classifier					Output of Classifier	Parameter Extraction		
Mod. Type	Sam. Freq (Hz)	Carr. Freq (Hz)	Band Width (Hz)	Code length /cpp		Band Width (Hz)	Carr. Freq (Hz)	Code length/cpp
P4	7,000	1,000	200	4/5	P4	193	968	4/5
P4	10,000	2,000	400	5/5	P4	396	1980	5/5
P4	10,000	2,000	400	7/5	P4	396	1980	7/5
P4	10,000	1000	200	6/5	P4	189	946	6/5
P4	7,000	1,000	200	8/5	P4	200	998	8/5
P4	10,000	2,000	250	6/8	P4	248	1988	6/8
P4	10,000	2,000	400	6/5	P4	405	2025	6/5
P4	10,000	1000	200	5/5	P4	199	996	5/5
P4	16,000	5000	1000	5/5	P4	998	4990	5/5

The results show that using Wigner Ville Distribution, classification and extraction of parameter of BPSK and FMCW can be measured accurately. The simulation results of various LPI radar signals and the percentage of estimation of different parameter of LPI radar signals plotted are shown below.

The below figure 2.21 and figure 2.22 gives the percentage estimation of different parameters of BPSK and FMCW using WVD respectively. The carrier frequency of both BPSK and FMCW can be extracted accurately upto 98%.



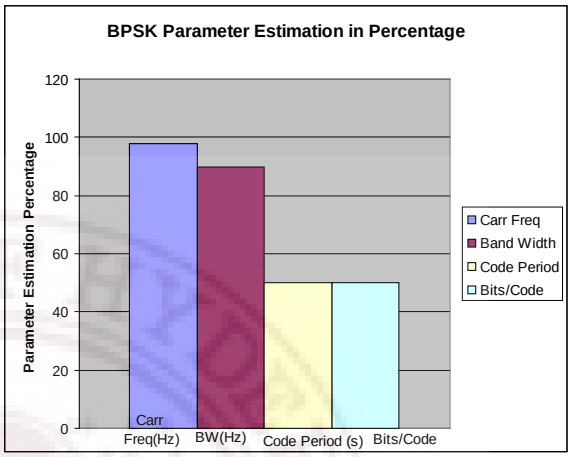
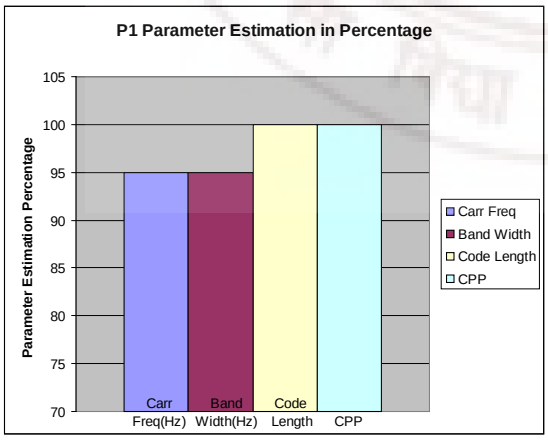


Figure 2.21 Percentage estimation of BPSK parameter

Figure2.22 Percentage estimation of FMCW parameter

The below figure 2.23 and figure 2.24 gives the percentage estimation of different parameters of P1 and P2 code respectively. The carrier frequency of both P1 and P2 codes can be extracted accurately up to 95% and the code length can be estimated accurately up to 100%.



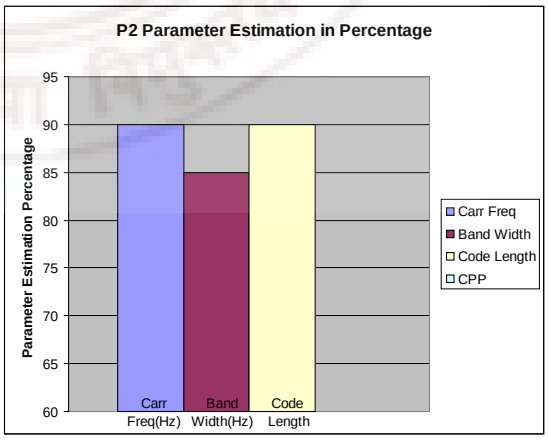


Figure 2.23 percentage estimation of P1 parameter

Figure 2.24 percentage estimation of P2 parameter

The below figure 2.25 and figure 2.26 gives the percentage estimation of different parameters of P3 and P4 code respectively. The carrier frequency of both P3 and P4 codes can be extracted accurately up to 95% and the code length can be estimated accurately up to 100%.

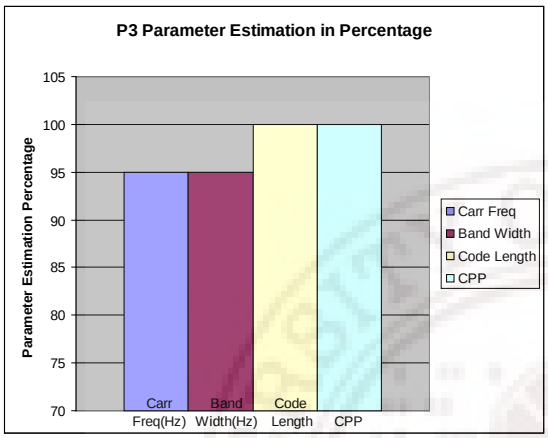


Figure 2.25 percentage estimation of P3 parameter

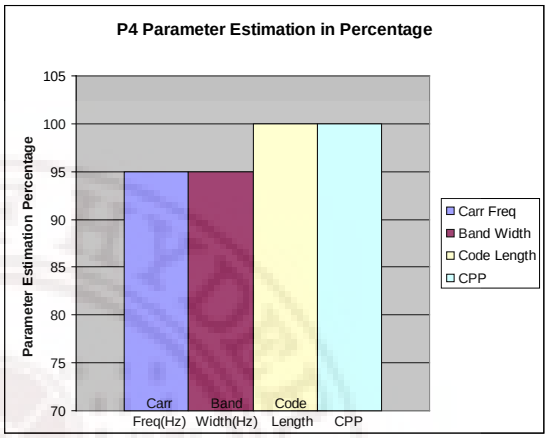


Figure 2.26 percentage estimation of P4 parameter

The below figure 2.27 gives the percentage estimation of different parameters of Frank code. The carrier frequency of Frank code can be extracted accurately up to 98%, code length and CPP can be estimated accurately up to 100%.

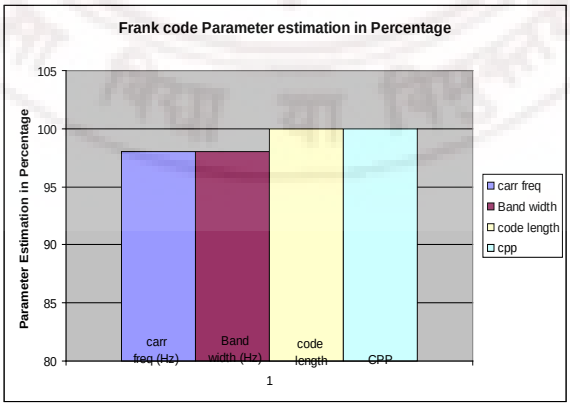


Figure 2.27 percentage estimation of Frank code Parameter

References

[1] Phillip E. Pace Detecting and classifying Low Probability of Intercept Radars. Artech House Radar Library, 2004.

[2] Nadav Levanon, Eli Mozeson Radar Signals. John Wiley, 2004.

[3] Jen-Yu Gau "Analysis of Low Probability of Intercept Radar Signals using Wigner Distribution," Sep 2002.

[4] Ram Bilas Pachori, Pradip Sircar "Analysis of Multi-Component Non-Stationary Signals using Fourier-Bessel Transform and Wigner Distribution".

[5] Jarmo Lunden, Liisa Terho1 and Visa Koivunen "Classifying Pulse Compression Radar Waveforms Using Time-Frequency Distributions" Conference on Information Sciences and Systems, The Johns Hopkins University, March 16-18, 2005.

[6] E. R. Zilberman, P.E. Pace "Autonomous Time Frequency Morphological Feature Extraction Algorithm for LPI Radar Modulation Classification".

[7] Bassem R. Mahafza *Radar system analysis and design using matlab* Chapman & Hall/CRC.

[8] MATLAB®, The Language of Technical Computing, Version 7.1.

[9] Time Frequency Analysis for Single Channel Applications, John Saunders ,High Performance Embedded Computing (HPEC) Conference September 30, 2004

CHAPTER 3

HIL OF MODULATION CLASSIFICATION USING WVD

3.1 Introduction to HIL

Hardware-in-the-loop (HIL) is a form of real-time simulation, it differs from conventional simulation by an addition of a real component (FFT) in the loop. This component may be FFT implemented in FPGA or ASIC. The current industry definition of a Hardware-In-the-Loop system is shown in Figure 3.1. It shows that the Modulation classification being simulated in MATLAB environment (inside PC) and the FFT a part of WVD is being implemented in FPGA. The purpose of the HIL simulation is to provide all the electrical stimuli needed to fully exercise the FFT via IO Blocks.

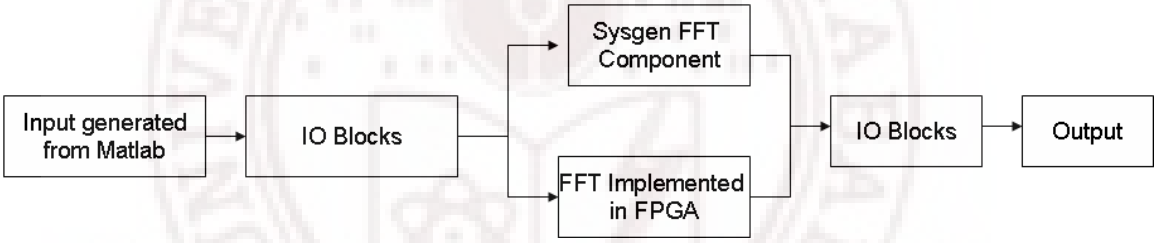


Figure 3.1 Definition of Hardware in the loop

The figure 3.2 shows the design cycle of HIL using Xilinx Spartan 3E and Matlab. The VHDL-module for FFT is implemented into Spartan 3E FPGA and the modulation classification and parameter extraction algorithm is designed in Matlab.

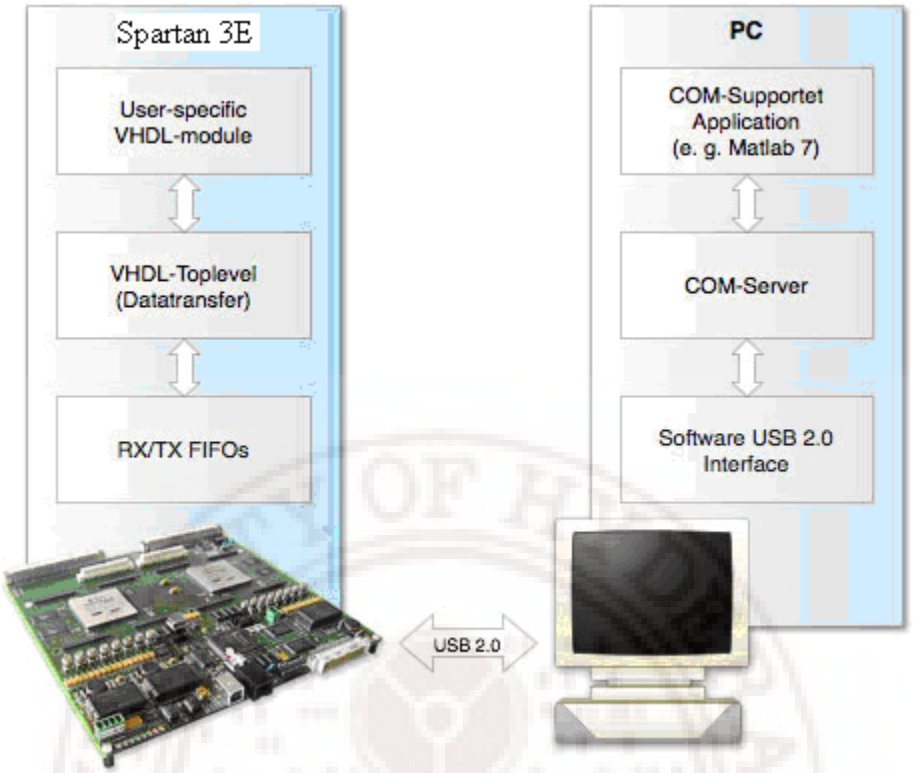


Figure 3.2 Hardware in the loop design cycle

The figure 3.3 shows the HIL simulation of FFT implemented in Spartan 3E FPGA using sysgen. The specifications of processor supporting MATLAB are Intel Pentium D processor, CPU clock speed is 2.99 GHz and 0.99 GB RAM.

3.2 Design Blocks

The simulation of WVD consists of 2 blocks as shown in figure 3.4

1) KERNEL
2) FAST FOURIER TRANSFORMATION

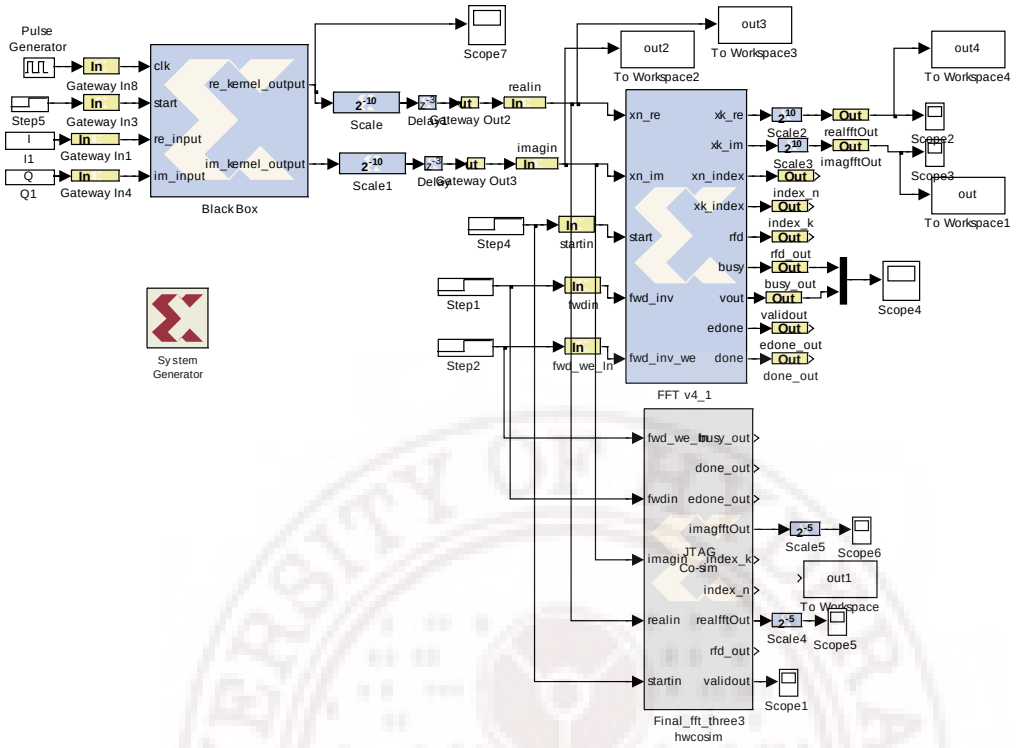


Figure 3.3 HIL simulation of WVD with FFT in Spartan 3E FPGA

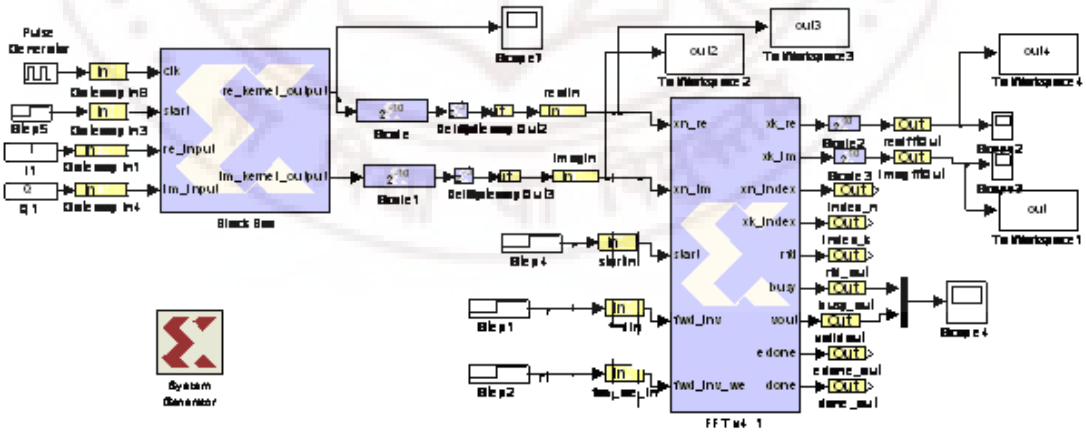


Figure 3.4 Design of WVD using sysgen

3.2.1 Kernel (Black box component)

The function $x(l+n)x^*(l-n)$ is used to calculate the instantaneous correlation of $x(n)$ and is represented by ' $f(l,n)$ ' known as Kernel function, i.e.

$$f(l,n) = x(l+n)x^*(l-n) \qquad (3.1)$$

The Kernel code is written in VHDL and using sysgen black box component, the kernel block corresponding to sysgen is created. Refer appendix A to see the procedure for the creation of Kernel Black box and its features.

3.2.1.1 Sample Periods

Sample periods in sysgen blocks are expressed as integer multiples of the system rate as specified by the Simulink System Period field on the master System Generator block. For example, if the Simulink System Period is 1/8, then the sample period of sysgen should be an integral multiple of 1/8.

3.2.2 FFT JTAG cosim block

The figure 3.5 shows the FFT sysgen component along with the gateway-in and gateway-out. The figure 3.6 shows the JTAG co-sim FFT component corresponding to sysgen FFT component. The complete procedure for creation of JTAG co-sim component is given in appendix A.

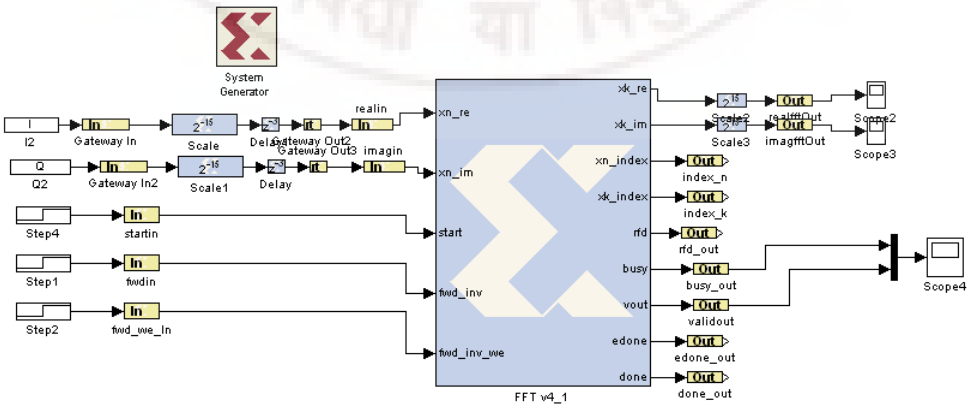


Figure 3.5 sysgen FFT block and connections

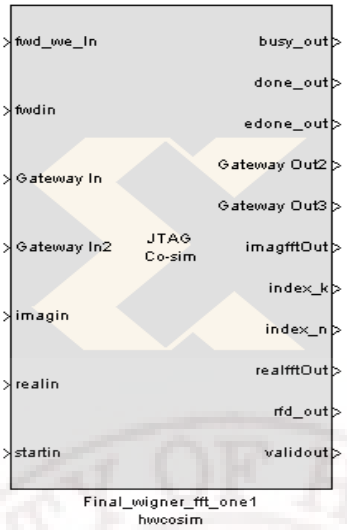


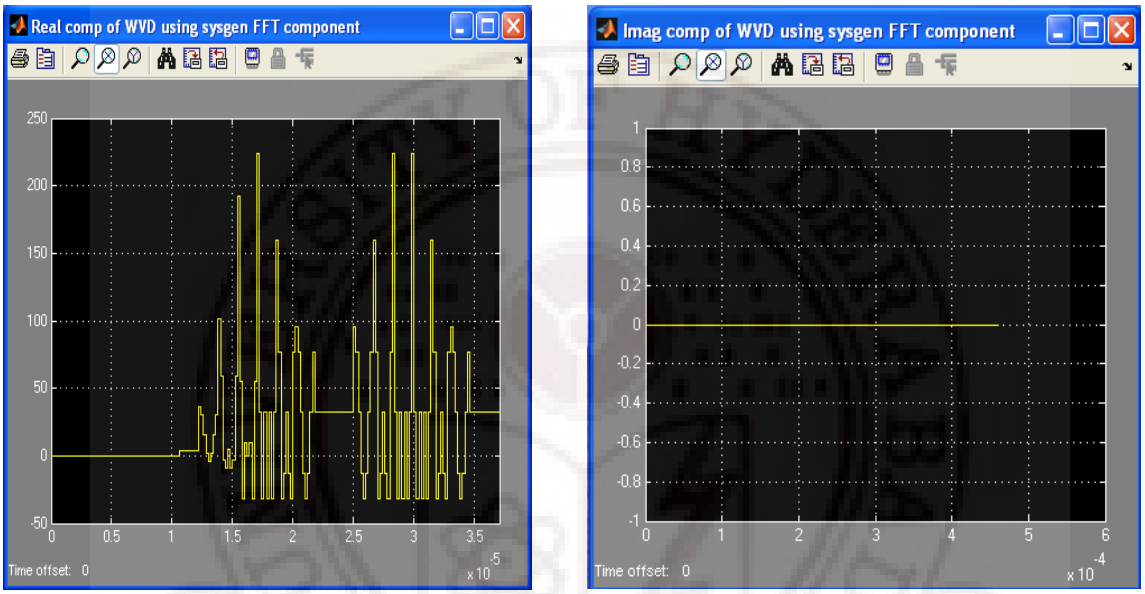
Figure 3.6 JTAG cosim block of FFT corresponding to FFT block in figure 3.5

Figure 3.3 shows both the sysgen FFT block and JTAG cosim FFT block, the design can be run to verify the functionality of FFT in FPGA.

3.3 Simulation Reports

The simulation results of WVD using HIL simulation of FFT implemented in Spartan 3E FPGA is studied. The implementation of whole WVD into both Spartan 3E and Vertex 2Pro and the resources utilized are tabulated along with the timing report are listed in table 3.1 to 3.10.

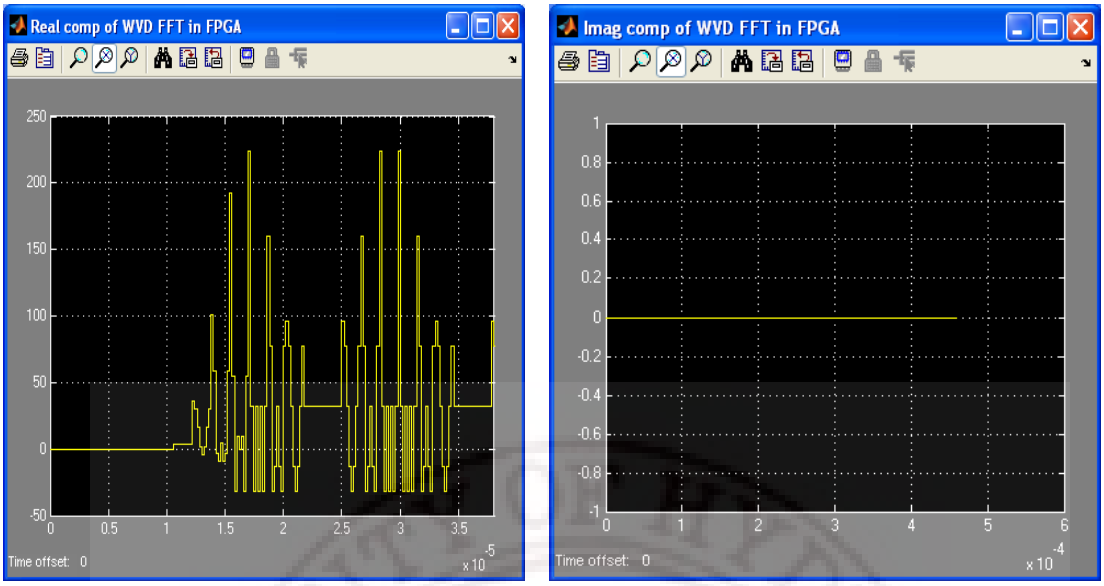
3.3.1 Comparison of WVD with FFT in Spartan 3E FPGA and using Sysgen component



(a) (b)

Figure 3.7 (a) real component (b) imaginary component of WVD using Sysgen block

The figure 3.7 (a) and (b) shows the WVD output using the sysgen FFT block of sample length 8, the input given to the block is *x(l)*={1,2,3,4,5,6,7,8}. Figure 3.8 (a) and (b) shows the WVD output with FFT of sample length 8 implemented in Spartan 3E FPGA, with the same input *x(l)*.



(a) (b)

Figure 3.8 (a) real component (b) imaginary component of WVD using FFT block in FPGA

3.3.2 Resources utilized for WVD

The different resources utilized by the WVD implemented into both Spartan 3E and vertex 2Pro is listed in table 3.1 to 3.4 for different sample length of FFT. Table 3.5 lists the timing report for WVD and its acceleration time for different sample length of FFT.

The below table 3.1 shows the slices utilized for implementing the WVD in Spartan 3E and vertex 2pro FPGAs, compared to Spartan 3E the vertex 2pro has more slices and hence the vertex 2pro is preferred for implementing the larger designs rather than Spartan 3E. As the size of FFT is increasing the slice resources utilized by both Spartan 3E and vertex 2pro are increasing.

Table 3.1 Slices utilized for WVD

No. of Samples (N)	FPGA RESOURCES (slices)	
	Spartan 3E 3s500efg320-4	Vertex2Pro (2vp30ff896-5)
8	4443 out of 4656 95%	1639 out of 13696 11%
16	4705 out of 4656 101% (*)	5163 out of 13696 37%
64	7291 out of 4656 156% (*)	9057 out of 13696 66%
128	10597 out of 4656 227% (*)	10778 out of 13696 78%
256	17634 out of 4656 378% (*)	21900 out of 13696 159% (*)
512	30972 out of 4656 665% (*)	31028 out of 13696 226% (*)

*requires more than available resources

The below table 3.2 shows the slice flip-flops utilized for implementing the WVD in Spartan 3E and vertex 2pro FPGAs, compared to Spartan 3E the vertex 2pro has more slice flip-flops. As the size of FFT is increasing the slice flip-flop resources utilized by both Spartan 3E and vertex 2pro are increasing.

Table 3.2 Slice Flip Flops utilized for WVD

No. of Samples (N)	FPGA RESOURCES (Slice Flip Flops)	
	Spartan 3E 3s500efg320-4	Vertex2Pro (2vp30ff896-5)
8	6726 out of 9312 72%	1810 out of 27392 6%
16	6843 out of 9312 73%	7112 out of 27392 25%
64	8392 out of 9312 90%	9480 out of 27392 34%
128	10456 out of 9312 112% (*)	10452 out of 27392 38%
256	14559 out of 9312 156% (*)	18696 out of 27392 68%
512	22807 out of 9312 244% (*)	22762 out of 27392 83%

The below table 3.3 shows the 4 input LUTs utilized for implementing the WVD in Spartan 3E and vertex 2pro FPGAs, compared to Spartan 3E the vertex 2pro has more 4 input LUTs. As the size of FFT is increasing the 4 input LUT resources utilized by both Spartan 3E and vertex 2pro are increasing.

Table 3.3 4 input LUTs utilized for WVD

No. of Samples (N)	FPGA RESOURCES (4 input LUTs)	
	Spartan 3E 3s500efg320-4	Vertex2Pro (2vp30ff896-5)
8	6277 out of 9312 67%	2079 out of 27392 7%
16	6753 out of 9312 72%	6962 out of 27392 25%
64	10688 out of 9312 114% (*)	13862 out of 27392 50%
128	15966 out of 9312 171% (*)	15788 out of 27392 57%
256	26269 out of 9312 282% (*)	31068 out of 27392 113% (*)
512	48029 out of 9312 515% (*)	44683 out of 27392 163% (*)

The below table 3.4 shows the No. of 18X18 Multipliers utilized for implementing the WVD in Spartan 3E and vertex 2pro FPGAs, the 18X18 Multipliers resources utilized are same for both the Spartan 3E and Vertex 2pro.

Table 3.4 MULT18X18s utilized for WVD

No. of Samples (N)	FPGA RESOURCES (MULT18X18s)	
	Spartan 3E 3s500efg320-4	Vertex2Pro (2vp30ff896-5)
8	26 out of 20 130% (*)	8 out of 136 5%
16	26 out of 20 130% (*)	26 out of 136 19%
64	26 out of 20 130% (*)	26 out of 136 19%
128	26 out of 20 130% (*)	26 out of 136 19%
256	26 out of 20 130% (*)	26 out of 136 19%
512	26 out of 20 130% (*)	26 out of 136 19%

3.3.3 Timing Report for WVD

The below table 3.5 shows the comparison of timing report for simulating the WVD in processor and implementing the WVD in FPGA. The computation time for WVD in vertex 2pro is less compared to Spartan 3E. The acceleration time decreases with the increase in the input sample length. The computational time for WVD in sysgen is calculated as

$(Latency + N^2) * synthesis\_clock\_frequency$, where N is sample length of FFT

Table 3.5 Timing report for WVD

No. of Samples (N)	MATLAB TIME (sec)	FPGA Minimum clock period (ns)		FPGA Maximum clock frequency (MHz)		Acceleration Ratio (M time/ Fpga time)	
		Spartan 3E 3s500e fg320-4	Vertex2Pro 2vp30 ff896-5	Spartan 3E 3s500e fg320-4	Vertex2Pro 2vp30 ff896-5	Spartan 3E 3s500e fg320-4	Vertex2Pro 2vp30 ff896-5
8	0.004032	23.654	19.112	42.277	52.324	1039	1286
16	0.004867	23.074	16.964	43.339	58.949	454	625
64	0.010193	23.723	19.098	42.154	52.363	47	58
128	0.065864	25.474	17.043	39.256	58.676	98	147
256	0.269735	25.808	22.555	38.748	44.336	63	72
512	3.441837	26.601	18.509	37.593	54.028	57	81

3.3.4 Resources utilized for FFT

The different resources utilized by the FFT implemented into both Spartan 3E and vertex 2Pro is listed in table 3.6 to 3.9 along with different sample length of FFT. Table 3.10 lists the timing report for WVD and its acceleration time for different sample length of FFT

The below table 3.6 shows the slices utilized for implementing the FFT in Spartan 3E and vertex 2pro FPGAs, compared to Spartan 3E the vertex 2pro has more slices and hence the vertex 2pro is preferred for implementing the larger designs rather than Spartan 3E. As the size of FFT is increasing the slice resources utilized by both Spartan 3E and vertex 2pro are increasing.

Table 3.6 Slice utilized for FFT

No. of Samples (N)	FPGA RESOURCES (slices)	
	Spartan 3E 3s500efg320-4	Vertex2Pro (2vp30ff896-5)
8	645 out of 4656 13%	703 out of 13696 5%
16	956 out of 4656 20%	1025 out of 13696 7%
64	2160 out of 4656 46%	2619 out of 13696 19%
128	3431 out of 4656 73%	4496 out of 13696 32%
256	3023 out of 4656 64%	3562 out of 13696 26%
512	4387 out of 4656 94%	5592 out of 13696 40%

The below table 3.7 shows the slice flip-flops utilized for implementing the FFT in Spartan 3E and vertex 2pro FPGAs, compared to Spartan 3E the vertex 2pro has more slice flip-flops. As the size of FFT is increasing the slice flip-flop resources utilized by both Spartan 3E and vertex 2pro are increasing.

Table 3.7 Slice Flip Flops utilized for FFT

No. of Samples (N)	FPGA RESOURCES (Slice Flip Flops)	
	Spartan 3E 3s500efg320-4	Vertex2Pro (2vp30ff896-5)
8	1059 out of 9312 11%	1066 out of 27392 3%
16	1570 out of 9312 16%	1576 out of 27392 5%
64	3328 out of 9312 35%	3333 out of 27392 12%
128	4920 out of 9312 52%	4926 out of 27392 17%
256	4751 out of 9312 51%	4757 out of 27392 17%
512	6431 out of 9312 69%	6412 out of 27392 23%

The below table 3.8 shows the 4 input LUTs utilized for implementing the FFT in Spartan 3E and vertex 2pro FPGAs, compared to Spartan 3E the vertex 2pro has more 4 input LUTs. As the size of FFT is increasing the 4 input LUT resources utilized by both Spartan 3E and vertex 2pro are increasing.

Table 3.8 4 input LUTs utilized for FFT

No. of Samples (N)	FPGA RESOURCES (4 input LUTs)	
	Spartan 3E 3s500efg320-4	Vertex2Pro (2vp30ff896-5)
8	889 out of 9312 9%	882 out of 27392 3%
16	1387 out of 9312 14%	1381 out of 27392 5%
64	3368 out of 9312 36%	3363 out of 27392 12%
128	5463 out of 9312 58%	5457 out of 27392 19%
256	4735 out of 9312 50%	4704 out of 27392 17%
512	7135 out of 9312 76%	7124 out of 27392 26%

The below table 3.9 shows the No. of 18X18 Multipliers utilized for implementing the FFT in Spartan 3E and vertex 2pro FPGAs, the 18X18 Multipliers resources utilized are same for both the Spartan 3E and Vertex 2pro.

Table 3.9 MULT18X18s utilized for FFT

No. of Samples (N)	FPGA RESOURCES (MULT18X18s)	
	Spartan 3E 3s500efg320-4	Vertex2Pro (2vp30ff896-5)
8	4 out of 20 20%	4 out of 136 2%
16	4 out of 20 20%	4 out of 136 2%
64	10 out of 20 50%	10 out of 136 7%
128	16 out of 20 80%	16 out of 136 11%
256	16 out of 20 80%	16 out of 136 11%
512	22 out of 20 110% (*)	22 out of 136 16%

3.3.5 Time Report for FFT

The below table 3.10 shows the comparison of timing report for simulating the FFT in processor and implemented in FPGA. The acceleration time for FFT in vertex 2pro is more compared to Spartan 3E. The acceleration time decreases with the increase in the input sample length. The computational time for FFT implemented in FPGA is calculated as

$(Latency + N) * synthesis\_clock\_frequency$, where N is sample length of FFT.

Table 3.10 Timing report for FFT

No. of Samples (N)	MATLAB TIME (sec)	FPGA Minimum clock period (ns)		FPGA Maximum clock frequency (MHz)		Acceleration Ratio (M time/ Fpga time)	
		Spartan 3E 3s500e fg320-4	Vertex2Pro 2vp30 ff896-5	Spartan 3E 3s500e fg320-4	Vertex2Pro 2vp30 ff896-5	Spartan 3E 3s500e fg320-4	Vertex2Pro 2vp30 ff896-5
8	0.000775	4.654	4.444	214.869	224.997	1535	1065
16	0.000883	4.654	4.444	214.869	224.997	874	915
64	0.001912	4.780	4.596	209.207	217.592	78	81
128	0.004022	4.883	5.004	204.794	214.838	81	84
256	0.010097	4.914	4.684	203.515	213.504	40	42
512	0.125886	4.986	4.860	200.563	205.772	50	51

3.4 Results

The above timing reports given in table 3.5 and table 3.10 shows that the implementation of FFT of WVD in FPGA accelerates the execution of time in the order of 50 for 512 point FFT. Where as implementation of WVD in FPGA accelerates the execution time in the order of 80 for 512 points of FFT.

Modulation classification of LPI signals using WVD (or FFT) implemented in FPGA will classify the LPI signals more quickly than the WVD implemented in processor.

The acceleration time for vertex 2pro is more compared to Spartan 3E FPGA. This is due to its higher operating frequency and dedicated functional build blocks for DSP operation. The table 3.5 and table 3.10 shows that the acceleration time for WVD implemented in FPGA is high than the acceleration time for FFT implemented in FPGA for higher sample length. Hence implementing the WVD in FPGA instead of FFT will improve the performance of the modulation classification algorithm of LPI signals using WVD.

References

[1] MATLAB®, The Language of Technical Computing, Version 7.1.

[2] XILINX ISE®, version 9.1

[3] XILINX SYSTEM GENERATOR, version 9.1

[4] zone.ni.com/devzone/cda/tut/p/id/3567

[5] www.precisionmba.com/hardware_in_the_loop.htm

[6] iaf-bs.de/products/ffp-basic.en.html

CHAPTER 4
CONCLUSION AND FUTURE WORK

4.1 Conclusion

The Modulation classification of LPI radar signals using WVD algorithm is studied and also extracted different parameters of the LPI radar signals using image processing techniques with 95% accuracy. HIL simulation of WVD is carried out using FFT implemented into Spartan 3E FPGA and the acceleration time is noted for different lengths of FFT. The acceleration time of 51 is achieved using FFT of sample length 512 implemented in Vertex 2Pro FPGA and an acceleration of 50 is achieved using Spartan 3E FPGA. The resources consumed are also presented including timing report.

4.2 Future work

The work done in this dissertation opens up several possible future explorations. The whole modulation classification and parameter extraction using WVD can be implemented into single system by using Hardware Software Co-design (HW-SW Co-design).

The classification of LPI radar signals and parameter extraction including noise signal has to be studied and to be implemented. The whole WVD is to be implemented into Hardware (FPGA) so as to further increase the acceleration time of WVD.

APPENDIX A

PROCEDURE FOR IMPLEMENTATION OF SYSGEN COMPONENTS

A.1 Procedure to create kernel black box in Sysgen

Write the VHDL code for Kernel in Xilinx ISE.

Select the sysgen black box and place it in the design.

Select the Kernel VHDL code and click ok.

Double click on black box and change the simulator to Xilinx ISE.

A.1.1Feature of Sysgen Black Box component

The System Generator Black Box block provides a way to incorporate hardware description language (HDL) models into System Generator. The block is used to specify both the simulation behavior in Simulink and the implementation files to be used during code generation with System Generator. A black box's ports produce and consume the same sorts of signals as other System Generator blocks. When a black box is translated into hardware, the associated HDL entity is automatically incorporated and wired to other blocks in the resulting design.

The black box can be used to incorporate either VHDL or Verilog into a Simulink model. Black box HDL can be co-simulated with Simulink using the System Generator interface to either ISE simulator or the ModelSim simulation software from Model Technology, Inc.

A.2 Procedure for implementing FFT into FPGA

Place the Sysgen block in the simulink design as shown in figure 3.4 (chapter 3).

Now place the sysgen FFT v4_1 block in the design with sample length=8 and enable the dynamic transform size to change the length of FFT dynamically.

The FFT v4_1 sysgen block implements an efficient algorithm for computing the Discrete Fourier Transform (DFT).

Parameters specific to the FFT v4_1 block are:

Number of Sample Points: transform length, one of N =8 to 65536.

Output Ordering: option to choose between bit reversed and natural order output.

Scaling: to select between Unscaled, Scaled output data types.

Rounding Mode: to choose between Truncation and Convergent Rounding to be applied at the output.

Phase Factor Bit Width: option to choose a value between 8 and 24, inclusive to be used as bit widths for phase factors.

Enable Dynamic Transform Size: option to have optional input ports nfft and nfft_we for dynamically varying the point size of input data frames.

Now connect the gateway in and gateway out to FFT block as shown in figure 3.5 (chapter 3), save the design. Double-click on the System Generator token and select **Hardware Co-Simulation → New Compilation Target…** under the **Compilation** field as shown in figure A.1

Enter the target board information and clock sections as follows:

- Board Name: Spartan 3E
- Frequency (MHz): 50 (This is the system clock frequency on the board)
- Pin Location: c9 (This is the pin location of the system clock)

Connect the USB cable to the board and turn the power on.

Click **Detect** to fill in the JTAG options

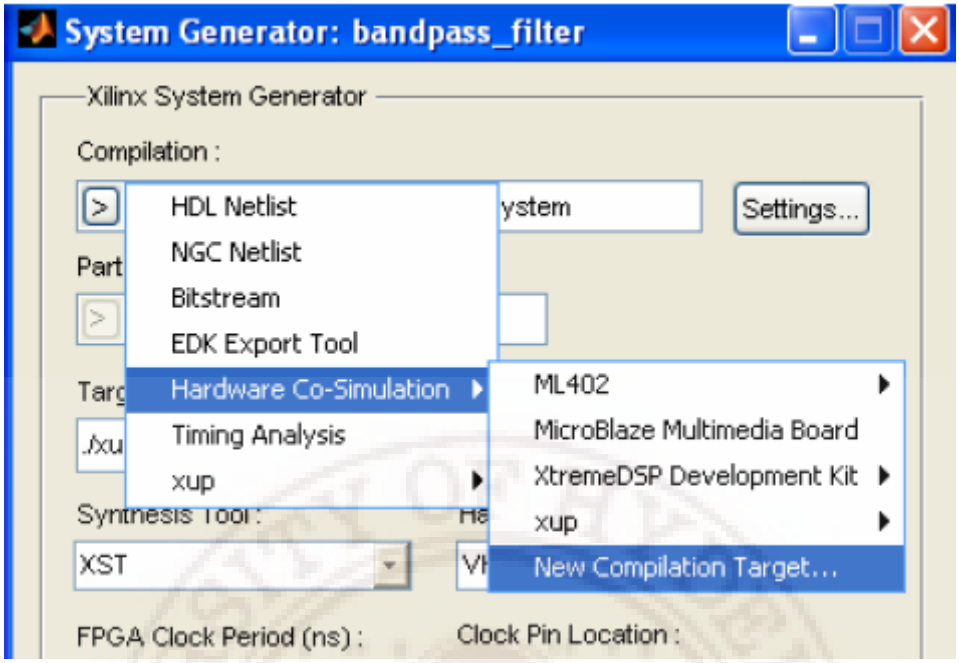


Figure A.1 to Create New Hardware Compilation Target

Under the **Targetable Devices** section, add the 3s500efg320-4

Click the Save Files button and save the files

Double click on the system generator token to verify that the following options are selected:

- Compilation: HDL Co-Simulation → spartan3E
- Target Directory: ./hwcosim
- Synthesis Tool: XST
- Hardware Description Language: VHDL

Click the generate button in the System Generator token to generate the hardware model as shown in figure 3.6 (chapter 3)

A.2.1Features of Fast Fourier Transformation (JTAG Cosim block)

The Xilinx JTAG Co-Simulation block supports HIL simulation using JTAG and a Parallel Cable IV or Platform USB. When a design is implemented by JTAG hardware co-simulation block, a new library is created that contains a custom JTAG co-simulation block with ports that match the gateway names (or port names if the subsystem is not the top level) from the original model. The co-simulation block interacts with the FPGA

hardware platform during the Simulink simulation. Simulation data that is written to the input ports of the block are passed to the hardware by the block. Conversely, when data is read from the co-simulation block's output ports, the block reads the appropriate values from the hardware and drives them on the output ports so they can be interpreted in Simulink.