Design and FPGA Implementation of an Adaptive Demodulator

by

Sandeep Mukthavaram

B.S. E.E Osmania University , Hyderabad, India, 1997

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science

Professor in Charge

Committee Members (2)

Date of Acceptance

Acknowledgments

”As the going gets tough, tough gets going”. At the end of two years at the ACS lab working with some amazing folks I would re-phrase it to ”As the going gets tough, you get stronger, more mature”. To begin with I would like to thank my advisor Dr. Joe Evans who has helped me through my graduate studies at KU and now with my thesis. Thank you particularly for amazingly fast email responses including the ones in the midnight. Thanks also to Dr. Gary Minden for being so patient when results didnot quite come by at the right intervals in the ACS project. The course I enjoyed most at KU was ”DSP for Communications” offered by Dr. Glenn Prescott. Thanks Prescott for making it so interesting and informative. My special thanks to Dr. Arvin Agah who has obliged to be on my committee at such a short notice. I cannot forget the long hours spent in the ACS lab with my buddies Karthik, Sarin and Hemang brainstorming for more ideas to deal with problems both software and hardware. They have been an amazing team to work with. My special thanks to Dan Depardo and Artur who have extended such support in dealing with the hardware related problems in the lab especially considering how frustrating we were sometimes.

Last but not the least I would like to thank my family and friends. Thankyou Mom and Dad for everything. You guys are amazing. My special friends Monica and Satish have made the Lawrence experience rich. Thanks guys.

Abstract

Signal Processing systems for communications will have to operate in rapidly changing environments. To suitably adapt to the varying requirements, control strategies targeted at selecting and tuning the signal processing algorithms need to be developed. The work being reported in this thesis is part of the bigger initiative by DARPA to develop and exploit reconfigurable computing for evolving defense systems. This work focuses on the use of automatic recognition of modulation type of the input signal to suitably reprogram the FPGA as a particular demodulator. The modulation schemes considered are PSK2 PSK4 and FSK2. This is used as a case study to demonstrate how reconfigurable computing can be a promising choice for building more adaptable and robust signal processing and communication systems.

This thesis first surveys the existing algorithms of Automatic Modulation Recognition (AMR). These algorithms are briefly analyzed to consider the feasibility of implementing them in hardware. A novel algorithm of automatic modulation is proposed and its FPGA implementation discussed in detail. It is also discussed how the run-time reconfigurability offered by the FPGA can aid in building a universal demodulator.

The complete design flow followed for synthesizing the designs for FPGAs is presented. Design and implementation details of the individual demodulators is discussed in great detail. Certain hardware optimizations that exploit the available FPGA architectures are documented. The test setup used for testing these radios is briefed.

Finally all the demodulators and the modulation recognizer are integrated into a run-time reconfigurable set up. Results from testing are reported and discussed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

There has been lot of research on the use of reconfigurable computing based on Field Programmable Gate Arrays (FPGAs) to accelerate computation intensive applications like Digital Signal Processing (DSP). A few aspects of DSP considered as part of the ”Adaptive Computing Systems” project at the University of Kansas are radio processing functions and their FPGA implementation. To dynamically reconfigure the same piece of silicon to perform different radio processing functions as required by changing requirements forms the basic motivation for the work being presented here. Changing requirements are identified to be different modulation types that may be present in the input signal. To be able to detect the change in the input modulation scheme while demodulating some other modulation type and suitably reconfigure the FPGA is the end goal. For this we need an algorithm for recognizing the modulation type. There is enough literature on this and this is a fairly old problem. However most of the algorithms that we came across have been either too expensive propositions for hardware implementations or good off-line solutions. To address this novel algorithm of modulation recognition is proposed, that can recognize PSK2, PSK4 and FSK2 modulations. Extensions to this algorithm can be useful to accommodate other types of modulation.

1.2 Thesis Overview

This chapter begins to describe the motivation for the work. The later part of this chapter will discuss the challenges posed by real-time processing of digital signals and why FPGAs make such a good candidate choice for implementing signal processing functions. There are three very distinct and important aspects of this work: reconfigurability, FPGAs, and signal processing of communication signals. Each of these are introduced in this chapter.

The second chapter titled ”Automatic Modulation Recognition” begins with explaining the need for automatic modulation recognition. Then we do a brief survey of the existing algorithms and discuss their possible hardware realizations and the problems involved therein. We then propose a novel algorithm of modulation recognition for the modulation of the input signal being PSK2, PSK4 or FSK2. We discuss its FPGA implementation.

Chapter three discusses in depth the FPGA implementation of the demodulators for PSK2, PSK4, and FSK2. The design flow that starts from hand coding in VHDL to generating configuration data to download onto the FPGA is explained. The design, implementation and results of the sub-modules are reported. We discuss certain optimizations that lead to better performance in terms of space and time. To conclude this chapter we explain the test set up we used to validate the designs.

In chapter four we integrate the modulation recognizer and the individual demodulators into a reconfigurable set up. The idea here is to have a single demodulator on silicon at a given time, while sensing for any changes in the modulation type.

The chapter titled ”results” reports the results we obtained from testing. In the concluding chapter some limitations to the algorithm are mentioned with possible explanations .The thesis ends with some suggestions for future work.

1.3 Reconfigurability, FPGAs, DSP

System performance is a major concern while designing complicated systems. The maximum performance can be achieved when circuits are optimized for single problem. As a new problem arises with minor changes to the old one, the whole system has to be redesigned and re optimized. Reconfigurable computing addresses this problem by allowing dedicated circuits to be built on to FPGAs and modifying these circuits by reprogramming the chips again. This greatly improves system flexibility and functional density. The circuits are loaded into the hardware and unloaded from it dynamically during the operation of the system. This was a brief introduction to reconfigurable computing.

The high throughput computational requirements of real-time digital signal processing (DSP) systems typically dictate hardware intensive solutions. And by increasing the system density configurable computing can deliver functionality of a device many times more than its size by dynamically reconfiguring the system. In digital signal processing (DSP) applications, the emphasis is most often on performance, in terms of real-time requirements as well as throughput. For radio signal processing in particular, the high throughput requirements dictate that a hardware solution is almost always required. Currently available DSP microprocessors are suited for baseband waveform processing. Conventional DSP microprocessors are implemented using inherently serial architectures. A DSP chip operating at 40 million instruction second (MIPS) has a useful bandwidth of less than 500 kHz. [12] DSP microprocessors are thus inadequate for the implementation of most stages of a radio system.

Field Programmable Gate Arrays (FPGAs) provide a rapid prototyping platform, which can be reprogrammed for different hardware functions without incurring the non-recurring engineering costs typically associated with custom IC fabrication. Implementing DSP functions in FPGA provides the several advantages over conventional DSP hardware: *Reconfigurability* and *Parallelism*.

Chapter 2

Automatic Modulation Recognition

This chapter introduces the concepts of automatic modulation recognition of communication signals. This is a fairly old problem and we here do a brief survey of the existing algorithms. Novel algorithm and its FPGA implementation is presented towards the later part of this chapter.

2.1 Need for Automatic Modulation Recognition

Signal Processing systems for communications will operate in open environments, where it is required that signals of different typologies be processed, which come from different sources, hence with different characteristics and for different user requirements. [4] Communication signals traveling in space with different modulation types and different frequencies fall in very wide band. Usually, it is required to identify and monitor these signals for many applications, both defense and civilian. Civilian applications may include monitoring the non-licensed transmitters, while defense applications may be Electronic surveillance systems. [2]

Modulation recognition is extremely important in COMINT applications for several reasons. Firstly, applying the signal to an improper demodulator may partially or completely damage the signal information content. Secondly, knowing the correct modulation type help recognize the threat and to determine suitable jamming waveform.

2.2 Existing Algorithms of Automatic Modulation Recognition

Survey of literature in related areas reveal that broadly there are two major approaches to modulation recognition: decision theoretic approach and statistical pattern recognition approach. The decision- theoretic approach is based on probability and hypotheses. In statistical pattern recognition approach, the classification system is divided into two subsystems. First is a feature extraction subsystem, which extracts the pre-defined features from the stored data. The second system is a pattern recognition subsystem that actually determines the modulation of the signal. Again the pattern recognizer works in two phases. The training phase to configure the classifier network followed by a test phase that gives the classification decision. Clearly the second approach is a good off-line solution to the problem at hand.

It has been mentioned in [2] that there are five techniques for solving the modulation recognition problem. These are 1. Spectral Processing 2. Instantaneous amplitude, phase and frequency 3. Histograms of instantaneous amplitude, phase and frequency 4. Combinations of the above three. 5. Universal Demodulator.

2.2.1 Decision Theoretic Approaches

The basic concept involved in these approaches is that likelihood function (LF) or, equivalently, the log-likelihood function (LLF) of the observed waveform, contains all the necessary information for a variety of inference tasks (signal detection, classification and parameter estimation). In the paper [7] Huang and Polydoros have generalized a likelihood method for classifying any MPSK signal in AWGN. Soliman et al [13] developed an automatic modulation classification algorithm utilizing the statistical moments of the signal phase and used it to classify the modulation type of general M-ary PSK signals. Callagan et al [14] using zero crossing techniques classified different modulation types such as CW, AM, FM, SSB, FSK, and ASK. Hsue and Soliman also use zero-crossing techniques to classify PSK and FSK signals.

Assaleh et al [9] proposed a new method of modulation classification for digitally modulated signals. This method utilizes a signal representation known as the modulation model. The modulation model provides a signal representation that is convenient for subsequent analysis, such as estimating modulation parameters. The modulation parameters could be the carrier frequency, modulation type, and bit rate. The modulation model is formed via autoregressive spectrum modeling.

2.2.2 Pattern Recognition Techniques

Another popular method for the modulation recognition process is the application of Artificial Neural Networks (ANNs). The basic advantage claimed by this method over the decision theoretic approach is that a threshold for each neuron is chosen automatically and adaptively unlike the other approach. [2] The modulation recognizers based on the ANN approach have three stages : 1. Pre-processing in which key features from every signal are extracted 2. training and learning phase to configure the network. 3. test phase to classify the modulation type.

Key Feature Extraction - A communication signal can be completely defined by its amplitude, frequency and phase. Now based on certain experiments are chosen the statistics related to these parameters. The idea is to create a mapping between these key features and the modulation scheme. Choice of these features and the network architecture define the speed of convergence and the accuracy of the results. The basic advantage claimed by the pattern recognition techniques compared to the decision theoretic approach is that the threshold for the key features is set automatically in the first case. Thus there has been lot of work by various people who have come up with numerous sets of key features to identify the modulation scheme. However though some of them have excellent identification capabilities all of them are rather cumbersome options for real time identification systems.

Training Phase - Once the key features are identified and extracted the network is trained with the input-output pairs. This is done iteratively till the network ”settles” to the minimum error state. The speed of convergence again depends on the key features and the ambiguity associated with them. For example in case of PSK2 signal and a FSK2 signal mean instantaneous amplitude is a very ambiguous feature to distinguish the two. The simple fact being that both are constant amplitude signals and have differences in tones. Once the error falls below a specified limit the network is said to have converged. The weights or the interconnects are assigned the fixed values to be used for the testing phase.

Testing Phase - The network with pre-computed weights is used to identify the modulation in the given input signal.

2.3 New Algorithm of Automatic Modulation Recognition

The modulation types considered are Binary Phase Shift Keying (BPSK) - PSK2, Quadrature Phase Shift Keying (QPSK) - PSK4, and Binary Frequency Shift Keying (BFSK) - FSK2. All these signals can be characterized to have constant amplitude. It can be easily proved that by introducing M^{th} degree non-linearity into an M-ary PSK signal will remove the modulation present in the signal. Consequently, squared PSK2 signal should provide a spectral line at twice the carrier frequency $2f_c$, while PSK4 signal would yield a PSK2 signal, a fact that can be used to discriminate between the two. Number of clock cycles between two zero crossings would be different for different tones, a characteristic of the FSK2 signal. PSK signal would have a constant number of clock ticks between zero crossings. This is used to distinguish the PSK signals from the FSK signals.

The algorithm is explained diagrammatically in the figure 2.1. The modulated signal is squared. This gives a unmodualted waveform for the input being a PSK2 signal and a PSK2 signal for the PSK4 input. Time averaging is performed on the squared signal. A PSK2 signal would result in a dc value whereas a PSK4 signal would give spikes at places where modulation is still present in the signal. Differentiating this

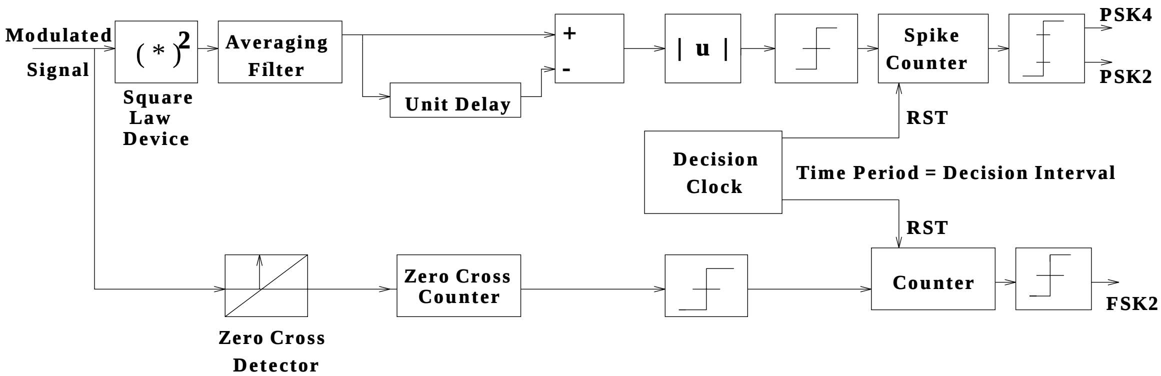

Figure 2.1: New Algorithm of Automatic Modulation Recognition

time-averaged signal helps capture the discontinuities in the PSK4 signal. A threshold is specified for the height of the spike. Number of spikes in a predetermined decision interval distinguishes PSK2 from PSK4. The modulation recognition process is a two parallel stage process. One parallel arm classifies the PSK2 and PSK4 while the other arm distinguishes FSK from PSK signals. It is assumed that the carrier frequency of the PSK signals lies in between the mark and the space frequencies of the FSK signal. To determine the frequency of the carrier, the number of clock cycles between zero crossings are counted. A threshold is fixed for the number of clock cycles between zero crossings. Cross overs of the threshold in a predetermined decision interval is used to determine if the signal has single or multiple tones.

2.4 FPGA Implementation of the AMR Algorithm

Certain optimizations need to be done to make the algorithm space efficient while implementing it in hardware. Here are a few of them.

Square law device is implemented as a table lookup. If the input modulated signal is digitized then we have finite number of bits that are used to represent each sample value. We used 8 in this case as the input bit width. If we take two's complement as a format to represent the data with no integer bits and 7 bits to represent the fraction then we have only 128 discrete possible outcomes for the squared value of the input signal. This is stored as a value in the ROM table. This is a much efficient way of squaring a signal compared to using a general purpose multiplier in terms of space.

Averaging Filter is implemented as a FIR low pass filter. The cut off is set to the data rate which is about 100 kHz. The filter is designed using the kaiser window with 25 taps. In the multiply and accumulate units of the FIR the multipliers are reduced to look up tables since the coefficients are known before hand. More detailed implementation details regarding the constant coefficient multipliers are discussed in the chapter 3.

Decision Clock. The decision interval is calculated by rigorous simulation. This block is implemented as a counter that resets at the end of every decision interval.

Zero Crossing Counter. Since the MSB of the input signal gives the zero crossings, counting the number of clock cycles between two rising edges of the MSB will give count between zero crossings which in turn gives a measure of the carrier frequency.

The individual blocks are integrated as shown in figure 2.1. At anytime only one of the output signals will be high depending on the input modulation. However it must be noted that change in the modulation type at the input will be reflected only at the end of the next decision interval. It is assumed that changes in the input modulation occur much slower than the decision clock.

In the above discussed blocks the averaging filter is the most expensive in terms of space. The results from the FPGA implementation of the proposed algorithm of modulation recognition have been very promising and in chapter 4 it will be discussed how this algorithm was used to build a run-time reconfigurable demodulator on FPGAs.

Chapter 3

Design and FPGA Implementation of PSK and FSK demodulators

This chapter begins with an in depth description of the design flow used to synthesize FPGAs. Later the design and implementation of the three demodulators, namely, PSK2, PSK4 and FSK2 is discussed. Certain domain specific optimizations that helped improve circuit performance in terms of space and time are documented. Also the hardware test setup is explained.

3.1 Design Flow for FPGA Synthesis

The design cycle starts with verifying the behavior of the block diagram that we are interested in implementing. We typically used *Simulink* from *Mathworks* for this purpose. The parameters for various lower-level components based on the specifications for the high-level modules are then derived. There are various software tools that support design of individual components and then integration into the system to verify the design using simulation. Most of the tools use floating point formats. Unfortunately, when we actually implement the design, we are constrained with finite length registers. The truncated values due to finite word lengths are typically fed back into the simulation tools in order to give a better approximation to the actual implementation.

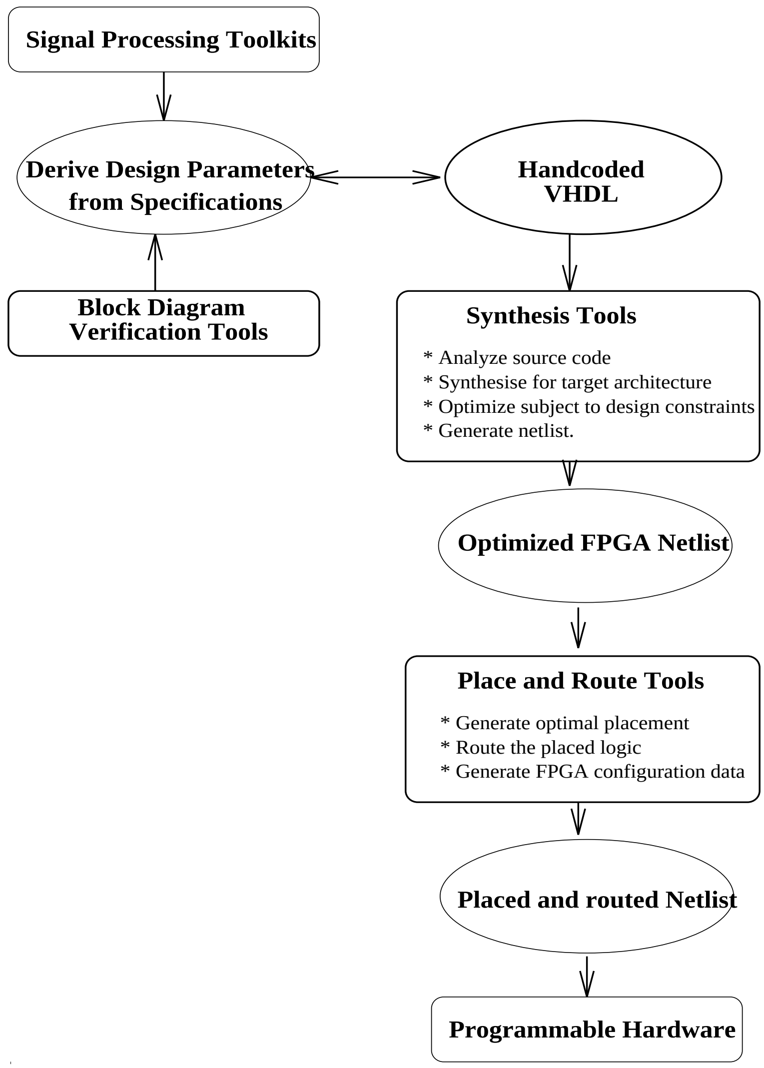

Figure 3.1: Design flow for FPGA synthesis

Once the ”magic numbers” from this phase of the design are obtained, the design is coded in VHDL. The synthesis involves analyzing the VHDL code, synthesizing for the target architecture, optimizing subject to design constraints such as placement directives or delay specifications, and generating an optimized FPGA netlist.

Placement and routing tools generate an optimal placement subject to delay constraints and then interconnect the logic using the available routing resources on the particular FPGA. A bit file containing FPGA configuration data that can be downloaded onto the chip is finally generated. The figure 3.1 explains this flow graphically.

3.2 Binary Phase Shift Keying Demodulator

The demodulator has a data rate of 100 kbps, although higher rates are quite feasible. The carrier frequency was selected to be 500 kHz. The binary PSK modulated signal is sampled at 8 MHz and fed as a digital input to the design. The PSK signal has the characteristic that the phase of the carrier wave changes at the data rate. The phase of the signal will be one of the M values for an M-ary PSK, where the M phases differ by 360/M^o. In this implementation of BPSK we had the carrier changing phase by 180^o. The phase switching occurs based on the data bit transmitted. The demodulator should be able to distinguish between a one or a zero based on the modulated input.

3.2.1 Theoretical Background

Phase Shift Keying is a widely used form of data transmission, well suited for synchronous data communications. [8] For unrestricted bandwidth PSK gives the lowest bit error rate for a given transmitted energy per bit. It is also efficient in the use of bandwidth [11]. The basic PSK system, for binary data, transmits one of the two phases of a carrier signal, depending on the sense of the bit transmitted. Thus one may be transmitted by the symbol $Acosw_ct$, while a zero is transmitted as $-Acosw_ct$. The sign reversal corresponds to a 180 degree phase shift, hence the name phase shift keying. The received modulated signal therefore is

The received modulated signal is

$$s(t) = k * d(t)cos(2\pi f_c t + \theta) \tag{3.1}$$

$$d(t)\epsilon\{-1, +1\} \tag{3.2}$$

The basic function of the BPSK demodulator is illustrated in Figure 3.2. At the receiver, a reference carrier is created from the input signal. This recovered carrier is at the same frequency as that of the original carrier, but devoid of any phase changes in the sense that the reference carrier has a constant phase. The recovered carrier is mixed with the input modulated signal to bring the signal to baseband. The baseband signal is passed through a low pass filter, which will filter out the higher frequency components arising a a result of mixing. A zero/one decision is made based on this output.

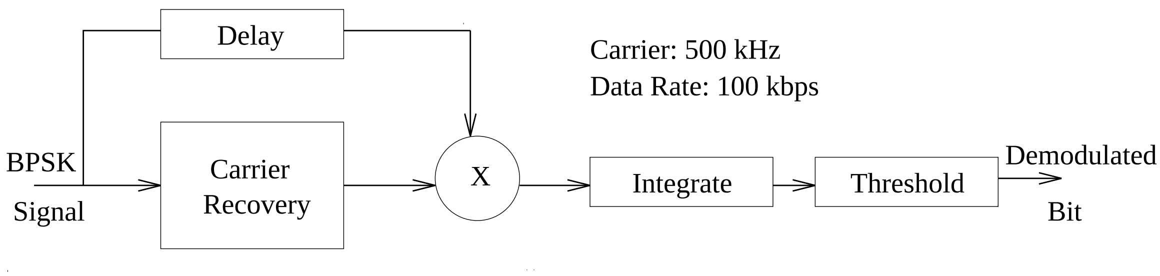

Figure 3.2: Block diagram for BPSK demodulator

There are numerous methods for generating the reference carrier from the input modulated signal. We chose an open loop carrier recovery scheme for implementation. The goal of the carrier recovery scheme is to generate a reference carrier with exactly the same frequency as that of the original carrier and having a constant phase, but it may have a constant phase difference with the original carrier. When data changes occur, the carrier will be in phase for one of the symbols (zero or one) and totally out of phase for the other symbol, so that when we mix the two signals we can have two different levels of amplitude from which we can make a decision.

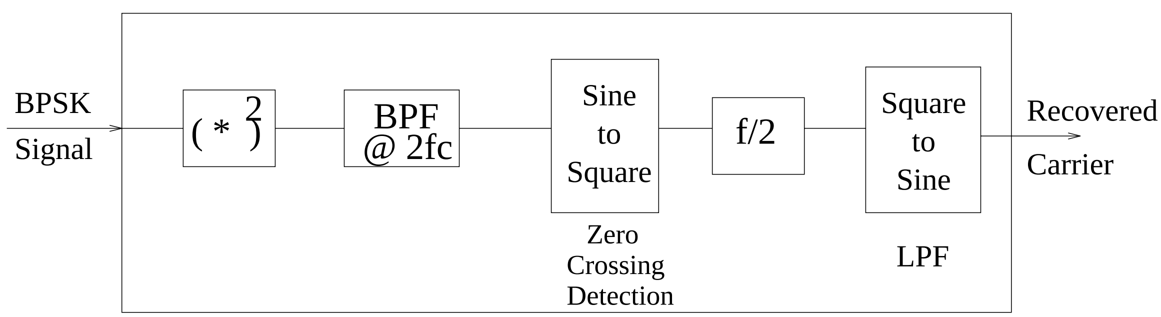

Figure 3.3: Block diagram for carrier recovery

The carrier recovery scheme takes the input in the form of a phase-modulated wave. The first module in the carrier recovery is a square law device. This eliminates the modulation present in the input signal and produces frequency components at twice the carrier frequency. By passing this output through the bandpass filter we select twice the carrier component. The output of the bandpass filter will then be

$$c(t) = k * d^2(t) sin(2\pi f_c t + 2\theta) \tag{3.3}$$

To have a clean reference at the carrier frequency we need to divide the frequency of the wave at the bandpass filter's output. To do this we convert the sine wave into a square wave by zero crossing transformation, since it is easier to half the frequency of the square wave. We then have a square wave of frequency same as that of the carrier. By passing it through a low pass filter which has a cutoff at the carrier frequency, we can exploit the fact that a square wave of a particular frequency can be viewed as a superposition of sine waves of different frequencies and pick out the sine wave of the fundamental frequency. The output of the carrier recovery circuit will always be locked to the input frequency for minor drifts in the frequency of the input. The drift the circuit can tolerate without losing synchronization will depend on the bandwidth of the bandpass filter.

We then have the phase-modulated input and a recovered carrier that have the same frequency. A delay is used to compensate for the delay through the tap line in the carrier recovery circuit so that the reference signal can be exactly in phase with one of the symbols. We then multiply these two signals. For symbols interval for which the input is in phase with the reference carrier, the multiplication will yield exactly the same as the squaring of the input signal would yield. The resulting wave would be all positive and have frequency components primarily at twice the carrier. For symbol intervals for which the input is totally out of phase with respect to the reference carrier, the multiplication would always yield a negative result and have similar frequency components as in the first case. By passing this output through a low pass filter, a square wave switching between positive and negative sides of the mean can be detected. The mean is used as a threshold to make a bit decision.

3.2.2 Implementation Details

3.2.2.1 Data Formats

We chose two's complement format to represent the signals in digital domain due to its ability to handle negative numbers inherently without an extra sign bit. Positive and negative numbers can be distinguished by the most significant bit of the given number.

3.2.2.2 Constant Coefficient Multipliers

One major class of DSP blocks used for communications is the filter. Filter implementation reduces to delay, multiply, and accumulate. The multipliers used in most filters are reduced multipliers, in the sense that one of the operand is fixed. Using constant coefficient multipliers instead of general multipliers yields significant savings in hardware resources for the circuit.

A constant coefficient multiplier can be implemented as a lookup table. [3] The output is determined for all possible values of the input and stored as a ROM. For example, for an 8 bit input and an 8 bit constant, there would be 256 entries in the ROM table, each entry being 16 bits wide. To take a practical approach, we use a hybrid technique, where the look up table is stored only for k*0,... k*F, where k represents the constant input. The input bits are grouped into groups of 4 bits. The lookup is performed with each group of four to obtain partial products, which are added to get the final product. For the above example, the 8 bit input would be split into two groups of 4 bits each (the upper nibble and the lower nibble), and two table lookups are performed to get two partial sums. The partial product resulting from the upper nibble is left shifted by 4 and added to the partial product resulting from the lower nibble to get the final product.

Two's complement multiplication is also easily implemented, by using a signed lookup table.

3.2.2.3 Square Law Device

The first block of the carrier recovery scheme is the square law device. This is implemented as a normal signed multiplier. The input is an 8 bit two's complemented number. This is the input to the whole system as such. Output for full precision would be 15 bit wide but is truncated to 8 bit to make it ideal for cascading it with the bandpass filter that follows it. The effect of squaring removes the modulation present in the input. The frequency spectrum of the output ideally would contain only the dc component and double the frequency component but in practice due to some finite amount of dc that may be present in the signal the carrier frequency also persists. The implementation details are tabulated in table 3.1.

Parttype	4013PG223-5		
CLB Usage	62 of 576 available ... 10%		
Max. Clock	17.75 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	1s	24s	26s

Table 3.1: Implementation details for the module *Square Law Device*

3.2.2.4 Bandpass Filter

The frequency response of the signal that is the input to bandpass filter is shown in the figure 3.4. The frequency of interest is double the carrier frequency. In this case for a carrier of 500 kHz the bandpass should be centered at 1 MHz. To implement this a Remez exchange window is used. The resulting filter is a 21 tap FIR. The filter design was done using Signal Processing Workshop (SPW). The 6dB bandwidth for the filter is 500 kHz and a null-null bandwidth of 950 kHz. The design takes a 8 bit two's complement input and gives a 12 bit wide output which also is two's complemented. The multiplication in the filter is done using constant coefficient multipliers which gives a lot of saving in space as compared to a generic multiplier. The theoretical versus implementation plots for the magnitude response is shown in the figure 3.4.

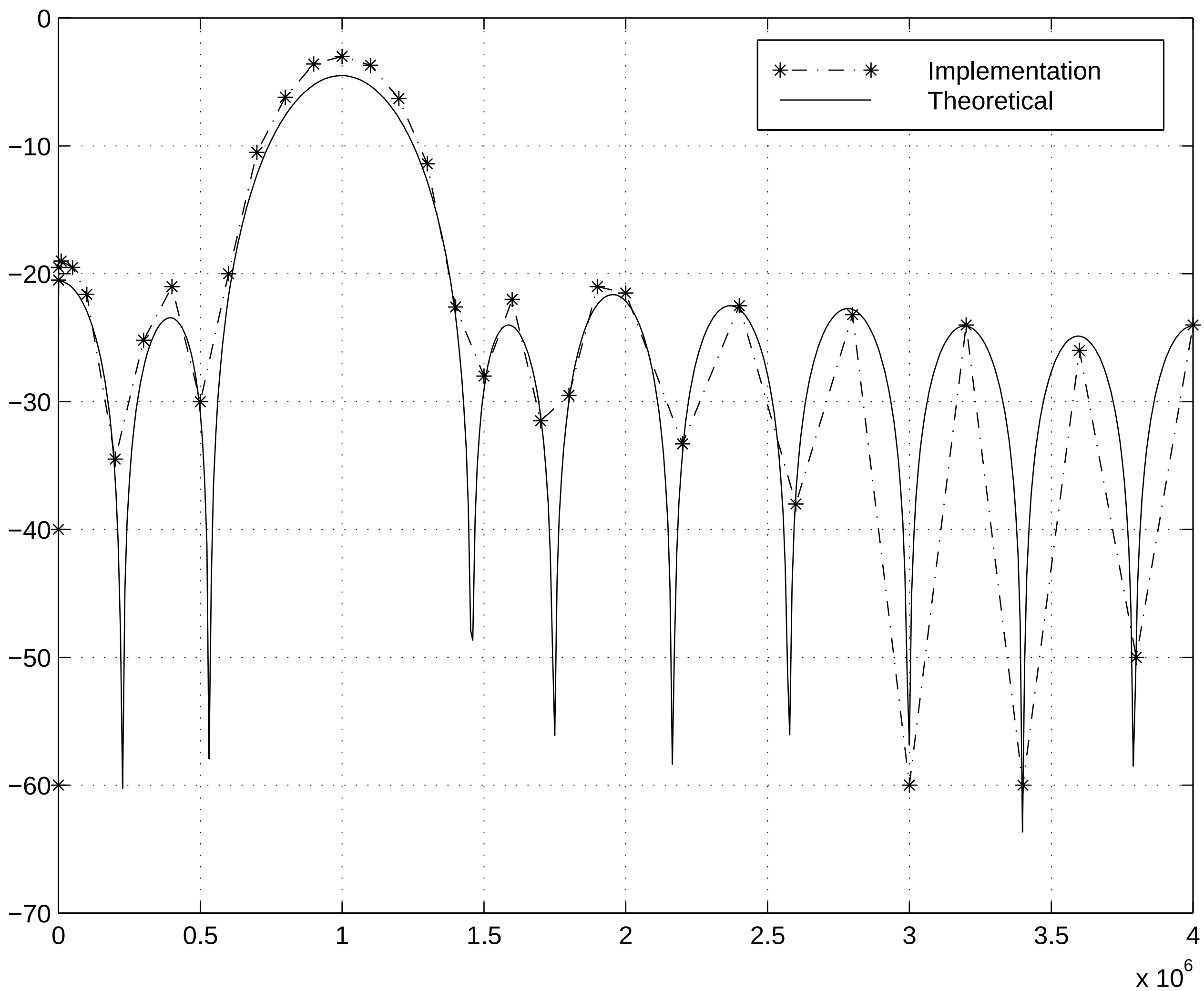

Figure 3.4: Magnitude response of Bandpass filter : Theory Vs Implementation

Here are the implementation details for the design.

Parttype	4013PG223-5		
CLB Usage	351 of 576 available ... 60%		
Max. Clock	10.32 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	5s	3m 1s	1m 49s

Table 3.2: Implementation details for the module *Bandpass Filter*

3.2.2.5 Frequency Divider

The sine wave from the bandpass is converted to a square wave by taking the most significant bit. This single bit square wave is used input to the frequency divider. The logic used for frequency division is as follows. The rising edge of the input wave is tracked and transition from the current state of the output enforced. The output stays latched for the falling edge of the input, hence doubling the time period or dividing the frequency by two. The implementation details are tabulated in table 3.3.

Parttype	4013PG223-5		
CLB Usage	4 of 576 available ... 1%		
Max. Clock	74.6 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	1s	13s	1s

Table 3.3: Implementation details for the module *Frequency Divider*

3.2.3 Square to Sine Converter

The output of the frequency divider is a square wave that has exactly the same frequency as that of the carrier. The goal is to generate a sine wave from the square wave. The input to the square to sine converter therefore is the square wave of carrier frequency. The input is a single bit and an output of 8 bits. The square wave can be visualized to be a superposition of sine waves of fundamental frequency along with other odd harmonics. By applying low pass filtering to such a frequency spectrum would yield a sine wave of fundamental frequency. As far as the implementation is concerned the multiplies can be done away with. This is because the input is a single bit and hence the product terms will be multiplication of the coefficients with either 1 or -1. The filter has a cutoff just more than the fundamental frequency, which is 500 kHz in our case. The filter chosen for implementation is a 16 tap kaiser window.

Parttype	4013PG223-5		
CLB Usage	193 of 576 available ... 33%		
Max. Clock	18.9 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	1s	1m	1s

Table 3.4: Implementation details for the module *Square to Sine Converter*

3.2.3.1 Carrier Recovery

The whole carrier recovery is integrated and tested. The table here shows the implementation details.

Parttype	4025EHQ240-4		
CLB Usage	530 of 1024 available ... 51%		
Max. Clock	10.14 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	8s	4m 45s	8m 7s

Table 3.5: Implementation details for the module *Carrier Recovery*

3.2.3.2 Data Filter

The function of the data filter is to smoothen the baseband waveform and reject the higher frequency components that result due to mixing. The input is 8 bit and output is 12 bit. The filter is designed to have a 6dB cutoff of 150 kHz and a null-null bandwidth of 500 kHz. The resulting filter is a 19 tap FIR filter. The window used was kaiser. The following plot shows a comparison of the magnitude responses of the filter designed and the implemented on the FPGA.

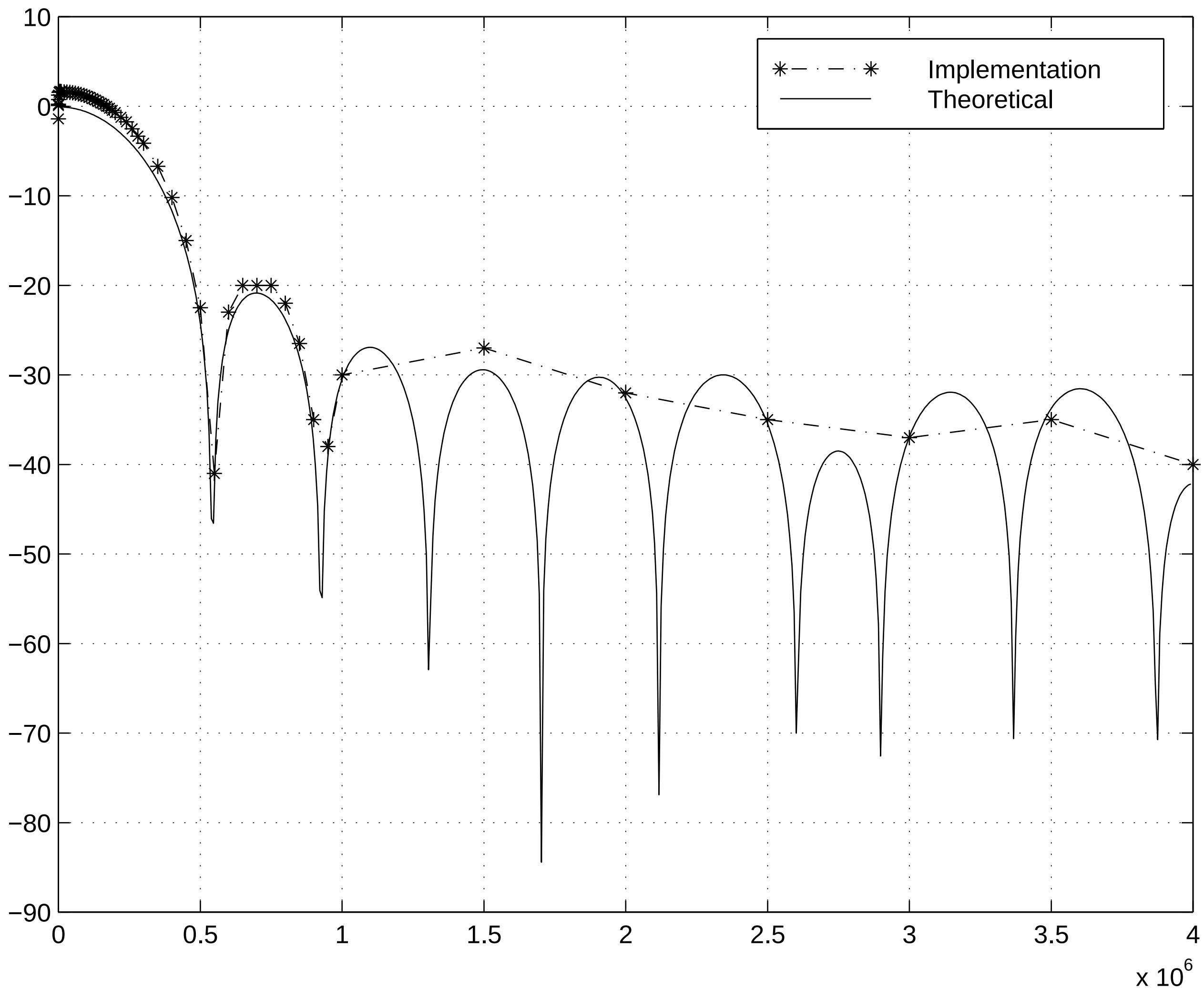

Figure 3.5: Magnitude response of Data filter : Theory Vs Implementation

Here are the implementation details.

Parttype	4013PG223-5		
CLB Usage	378 of 576 available ... 57%		
Max. Clock	10.3 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	5s	2m 46s	1m 57s

Table 3.6: Implementation details for the module *Data Filter*

3.2.3.3 Integration and Testing

All the above discussed modules are integrated as in the block diagram discussed in section 2. For generating the BPSK modulated signal Stanford Telecom's STEL-9231 is used. This takes as input the carrier frequency from the signal generator, serial data from the Bit Error Rate Tester (BERT) and gives out a modulated signal. This analog input is sampled using an A / D converter and the digital output is fed an input to the demodulator. The demodulation is performed which outputs a single bit data stream which is fed back into the received data terminal of the BERT. A detailed test setup is shown illustrated below:

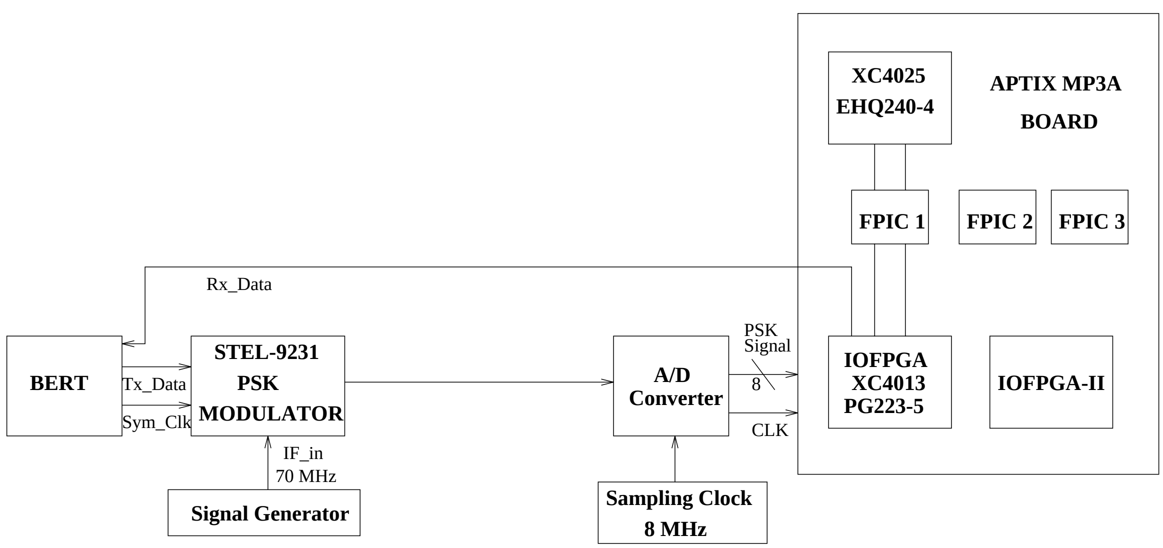

Figure 3.6: Test setup for the BPSK / QPSK demodulator

The IOFPGA on the APTIX board is used to route signal to and from the actual chip XC4025 on which the demodulator is implemented. The implementation details are presented below.

<table>
<tr><td>Parttype</td><td colspan="3">4025EHQ240-4</td></tr>
<tr><td>CLB Usage</td><td colspan="3">898 of 1024 available ... 87%</td></tr>
<tr><td>Max. Clock</td><td colspan="3">10.07 MHz</td></tr>
<tr><td rowspan="2">CPU Times On Ultra Sparc 1</td><td>Partition</td><td>Placement</td><td>Routing</td></tr>
<tr><td>35s</td><td>11m 44s</td><td>11m 23s</td></tr>
</table>

Table 3.7: Implementation details for the BPSK demodulator

The above implementation yielded a bit error rate of 10^{-9} or higher. The testing was done for more than 3 hours without a single bit error. The performance of the demodulator will be discussed in the *Testing and Results* chapter. To compare the implementation discussed above here are few other FPGA implementations and benchmarks [6]. The design used for bench marking was built for a data rate of 512 kpbs, a 5 MHz modulated carrier sampled at 20 MHz.

Vendor	Device	Required Logic	Available Logic	% of Available Logic
Altera	10K100	1682 LE's	4992 LE's	33.7 %
Lucent	2C40	355 PLC's	900 PLC's	39.4 %
Altera	10K50	1682 LE's	2880 LE's	58.4 %
Lucent	2C26	355 PLC's	576 PLC's	61.6 %
Lucent	2C15	355 PLC's	400 PLC's	88.8 %
Xilinx	4025E	919 CLB's	1024 CLB's	90.0 %

Table 3.8: Benchmarkings for the BPSK demodulator on FPGA

3.3 Quaternary Phase Shift Keying Demodulator

3.3.1 Introduction

The design was extended to 4-ary PSK and this section documents the design and implementation details. The date rate was doubled to 200 kbps or 100 k symbols/sec with two bits for each symbol. Most of the modules developed for the BPSK design were reusable for this design. The modules are discussed in the following section.

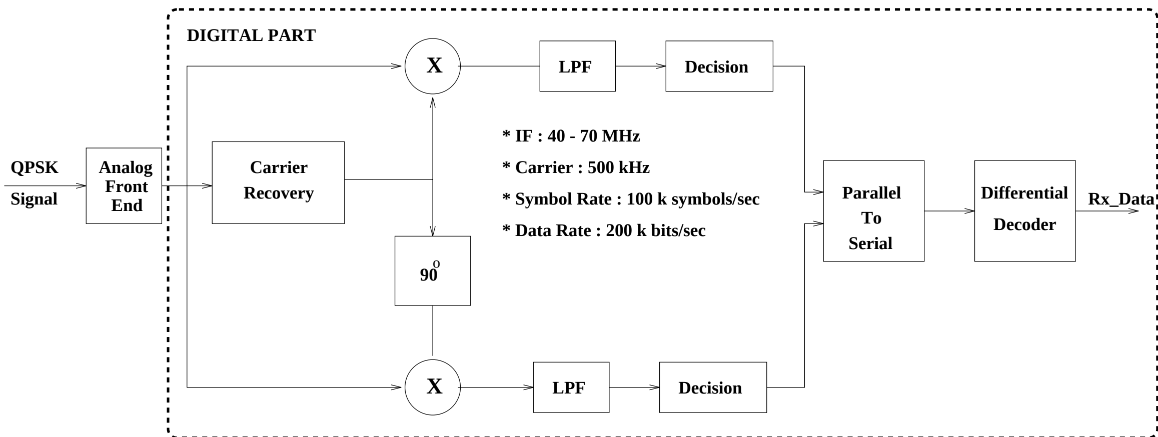

Figure 3.7: Block diagram for the QPSK demodulator

3.3.2 Carrier Recovery

Essentially all the blocks used for the BPSK design are reused. The square law device is replaced with a fourth power device. The reason for introducing a fourth power non-linearity is to get rid of the four phases contained in the modulated wave and get a pure sine wave. The output of the fourth power device now will have components at 500 kHz, 1 MHz, and 2 MHz. But only the components at 2 MHz are the one's without any phase changes. So in order to extract a 2 MHz signal we need to shift the center frequency of the band pass filter to 2 MHz. Since a carrier of 500 kHz is needed, the 2 MHz signal needs to be divided in frequency by 4. So the 2 MHz sine wave is converted into a square wave of 2 MHz and the square wave is divided by four in frequency. This is done by cascading two D-flip flops. The square wave is converted into a sine wave the same way as was done in BPSK. A QPSK demodulator can be built using two separate BPSK channels which are orthogonal to each other. For this purpose the carrier recovered needs to be phase shifted by 90o to implement a channel orthogonal to the first one. This is implemented using a Hilbert transform filter. Hilbert transform filter is realized as a 4-tap FIR structure with anti-symmetric coefficients. [5] One carrier locks in phase to one of the four possible phases and other carrier locks in phase to an orthogonal phase. The carrier recovery and the Hilbert transformer are fit into a single FPGA 4025EHQ240-4.

3.3.3 Upper Channel

The upper channel is implemented exactly as the in the BPSK design. The carrier recovered is mixed with the incoming QPSK signal that is delayed using an external delay elements. Then the data is demodulated. The lower channel that uses the orthogonal is also implemented on similar lines. The data filter is redesigned for a cutoff of 250 kHz to accommodate higher data rate.

3.3.4 Recombination of data

The I and Q data that are obtained on two channels need to be recombined before fed back into the BERT. Data is time multiplexed using a clock double the speed of the data in each channel. This is accomplished by suitably dividing the global clock by 40 again to synchronize with the data clock. Before feeding to the BERT the recombined data is decoded differentially. This is done simply by storing the previous bit and XORing with the current data bit.

3.3.5 Partitioning the design across multiple FPGAs

The general test set up shown in Figure is used for testing QPSK demodulator. The whole design is split across three different FPGAs. The IOFPGA contains the clock generator, data generator and the encoder along with the Front end, and the lower channel. The implementation details are tabulated in table 3.9.

Parttype	4013PG223-5		
CLB Usage	530 of 576 available ... 92%		
Max. Clock	10.06 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	4s	2m 25s	2m 23s

Table 3.9: Implementation details of the QPSK demodulator : IOFPGA

The FPGA1 which is 4025EHQ240-4 has the carrier recovery along with the Hilbert transformer implemented in it. The implementation details are tabulated table 3.10.

Parttype	4025EHQ240-4		
CLB Usage	835 of 1024 available ... 81%		
Max. Clock	9.41 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	8s	4m 32s	4m 15s

Table 3.10: Implementation details of the QPSK demodulator : FPGA-I

The FPGA2 has the upper channel in it. The input to this FPGA is the recovered carrier from the FPGA1 and the output of the front end of from the IOFPGA. The other implementation details are tabulated in table 3.11.

Parttype	4025EHQ240-4		
CLB Usage	538 of 1024 available ... 52%		
Max. Clock	9.34 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	4s	2m 43s	2m 51s

Table 3.11: Implementation details of the QPSK demodulator : FPGA-II

The performance of the demodulator will be discussed in the chapter 5

3.4 Binary Frequency Shift Keying Demodulator

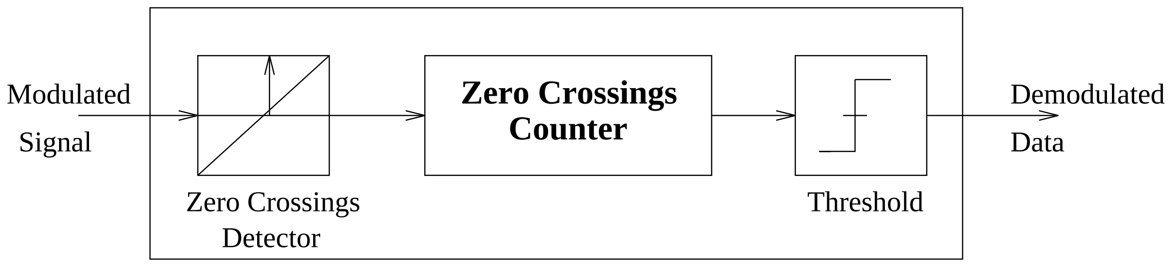

Figure 3.8: Block diagram for the BFSK demodulator

FSK signals have multiple tones in them. Each tone can be characterized by the number of clock cycles it takes between two zero crossings for a given sampling frequency. For example a 400 kHz signal sampled at 8 MHz will have 20 samples between two zero crossings and a 600 kHz signal would have about 13 samples. So a counter that can resets for every zero crossing would have two discreet values at the output. A threshold somewhere in between 20 and 13 would hard limit the output to digital levels. Thus demodulating the FSK signal. The block diagram for the FSK demodulator is shown in the figure To demonstrate this the two carriers are chosen to be 400 and 600 kHz for Mark and Space frequencies. To increase the tolerance to noise the sampling can be increased so that the values from the counter are separated further apart and decreasing the probability of an error. The test set up is very similar to the BPSK test set up. The implementation details are presented in the following table.

Parttype	4013PG223-5		
CLB Usage	4 of 576 available ...1%		
Max. Clock	63.4 MHz		
CPU Times On Ultra Sparc 1	Partition	Placement	Routing
	1s	4s	5s

Table 3.12: Implementation details of the BFSK demodulator

Chapter 4

Reconfigurable Demodulator

This chapter describes the reconfigurable platform used to implement the adaptive demodulator.

4.1 Introduction

The WILDFORCE reconfigurable computing engine is a Commercial Off the Shelf (COTS) product from Annapolis Micro Systems, Inc. It has Xilinx 4000 series of FPGAs as Parallel Processing Elements (PEs). It is a PCI based parallel high speed processing board.

4.2 Architecture

The general architecture is shown in the figure 4.1. [1] The board has 5 Xilinx 4085XLA FPGAs on them. Each Processing Element (PE) consists of a Xilinx 4000 Series FPGA programmable by the user, a memory controller and shared memory. WILDFORCE has a crossbar switch which allows non-adjacent processing elements to communicate. The communication between the host computer and the board is made possible using FIFOs which are in turn implemented on xilinx 4010 FPGAs or Direct Memory Access both using the 'C' API interface. For real time processing the board is equipped with external IO interface. It is also possible to get signals in and out of the and other control logic part from the functionality specified by the user. The interface to the memory or the other mezzanine cards is implemented on the Xilinx of the Processing Element being used along with the user code also called as ”Logic Core”.

4.2.1 FIFO Interface

The FIFO interface is meant to provide a simple means of file I/O to the FIFO. The actual FIFO is implemented on a xilinx 4010 FPGA. This is configured as a FIFO when the drivers are loaded. The FIFO interface to the user program however is implemented on the part of the Xilinx that user will program. The host to FIFO interface is provided by a ”C” Application Programmer’s Interface (API) calls. The clock for the circuit may be specified again by the C APIs. The three FIFOs used on the WILDFORCE board are 36 bit wide, consisting of a 32 bit wide bi-directional data word and a 4-bit wide tag, that is input only from the host.

4.2.2 Memory Interface

The Dual Port Memory Controller (DPMC) allows both the host and the processing elements to access the external memory cards on the WILDFORCE mezzanine expansion connectors. The memory accesses performed by the processing elements or the host are pipelined. All the accesses are arbitrated by the dual port memory controller. The external memory card is of size 1 MB.

4.2.3 Crossbar Model

The crossbar network is a set of Xilinx FPGAs used to implement processing element interconnection network. The data port between the processing elements and the crossbar is 36 bits wide, and is divided into nine four-bit nibbles. The crossbar can be configured between to establish connections between any same nibble of the two distinct PE ports. Each processing element has one connection to the crossbar, except for pro-

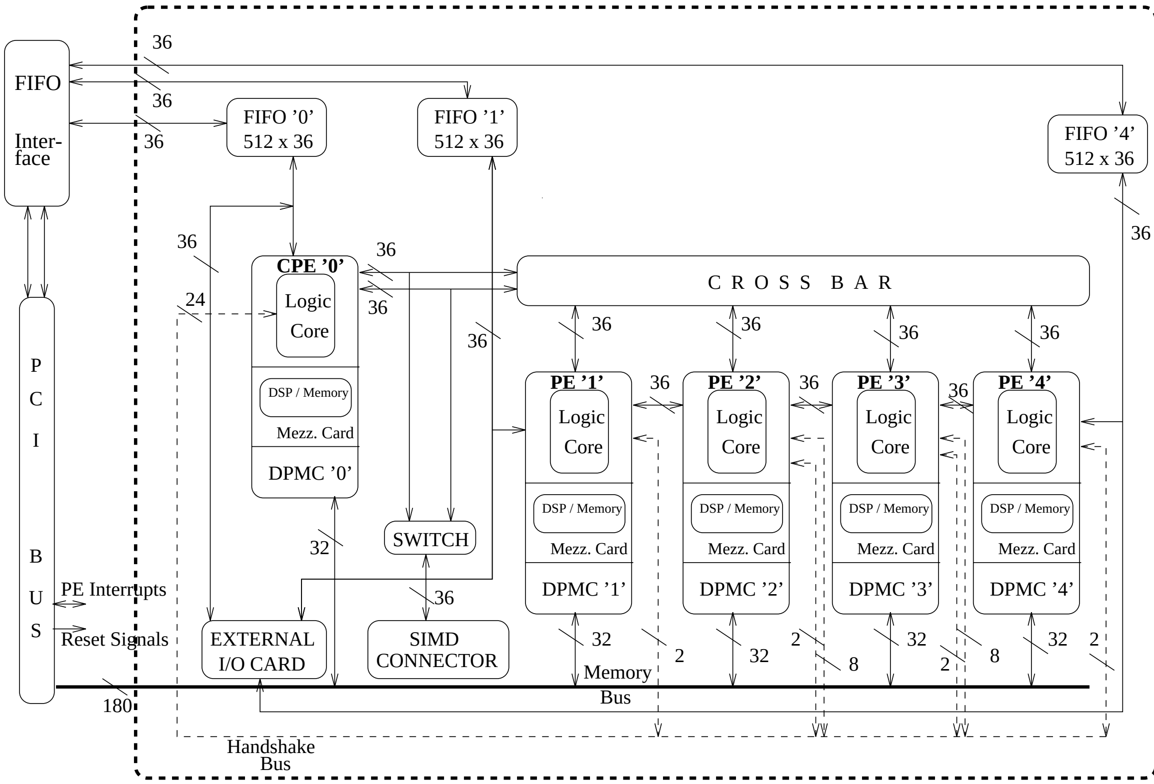

Figure 4.1: WILDFORCE : Architecture

cessing element 0 (CPE0), which has two such connections to port zero and ports five on the crossbar. There are lot of options in which crossbar can be configured. The crossbar allows for a reconfigurable, bi-directional set of data paths that enable any processing element to develop a set of connections to any other processing elements on the WILDFORCE board. Like the crossbar, another powerful means of communication between processing elements 1 through 4 is the systolic interface. Each processing element is connected to its neighboring element by this systolic bus. The systolic bus is bi-directional.

4.3 Programming the WILDFORCE

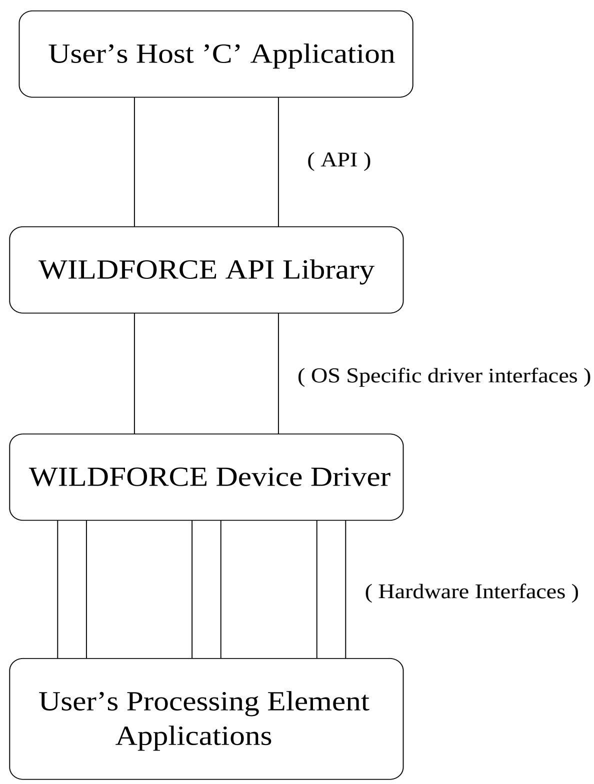

Figure 4.2: WILDFORCE : Software Hierarchy

The API routines provide high-level operations by performing combinations of

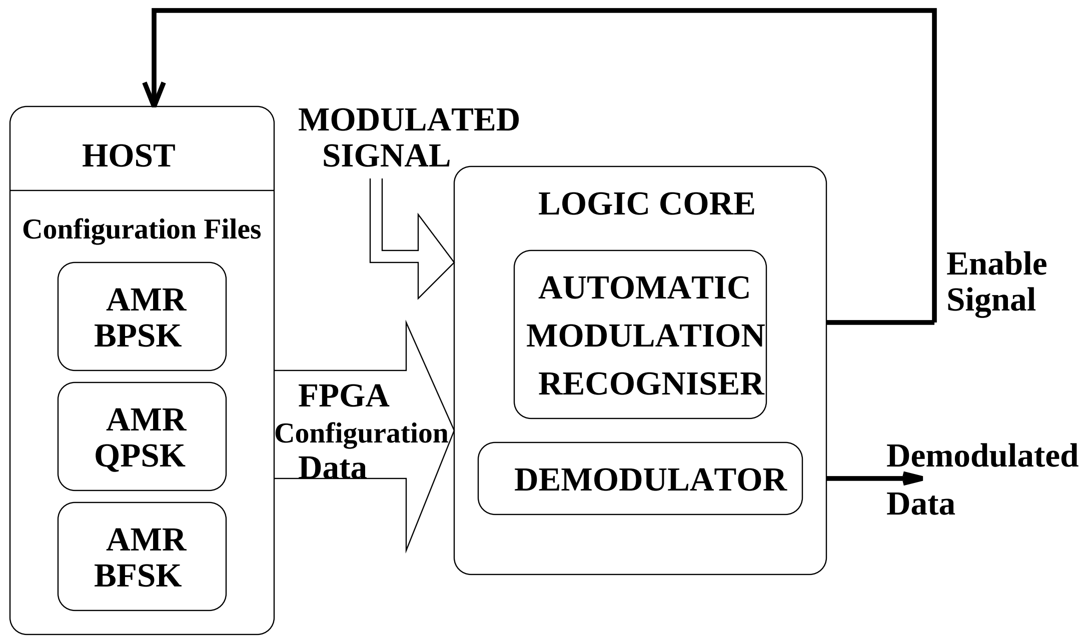

Figure 4.3: Adaptive Demodulator

low-level hardware interfaces. The software hierarchy is explained in the figure 4.2. [1] The input data file can be read using the C API and the data fed to the FPGA using the FIFO interface explained earlier. Similarly the processed data from the FPGA is dumped to the FPGA from where the C API can read it back. The C API accesses the FPGA via the WILDFORCE API library which in turn communicates with the hardware device drivers. This gives tremendous possibilities to the user to prototype, debug and implement complex systems. The application we are interested is to dynamically reconfigure the FPGAs depending on the type of modulation present in the input signal so as to demodulate accordingly. The user C program can be written to accordingly to download a different configuration file as and when required. Since the download time is extremely small to the order of less than a second, the interface could be acceptable for real time data processing. The modifications to extend the idea to process real signals is minimal. The fifo interface will be replaced by the external IO interface.

4.4 Adaptive Demodulation

The algorithm for the automatic modulation recognition was explained in chapter 2. This algorithm is used to build a reconfigurable demodulator. The modulation recognizer constantly monitors the type of the modulation present in the input signal and generates distinct enable signals for PSK2, PSK4 and FSK4 modulation type. Each of the demodulator is equipped with a modulation recognizer at the front end so that it can track the changes in the modulation of the input signal. While the data is being demodulated the enable signals are read using the C API. The C program keeps the state of the FPGA configuration as to what the current demodulator is. If the modulation changes the FPGA is reconfigured by downloading the appropriate bit file. So the ”Logic Core” (explained in section 4.2) can be either a PSK2, PSK4 or a FSK4 demodulator with a modulation recognizer hooked onto it. The general block diagram for the Logic Core is explained in the figure 4.3.

Chapter 5

Testing and Results

5.1 Performance Analysis of the Demodulators in Presence of Additive White Gaussian Noise

The general setup to test the demodulators in the presence of noise is illustrated in the figure 5.1.

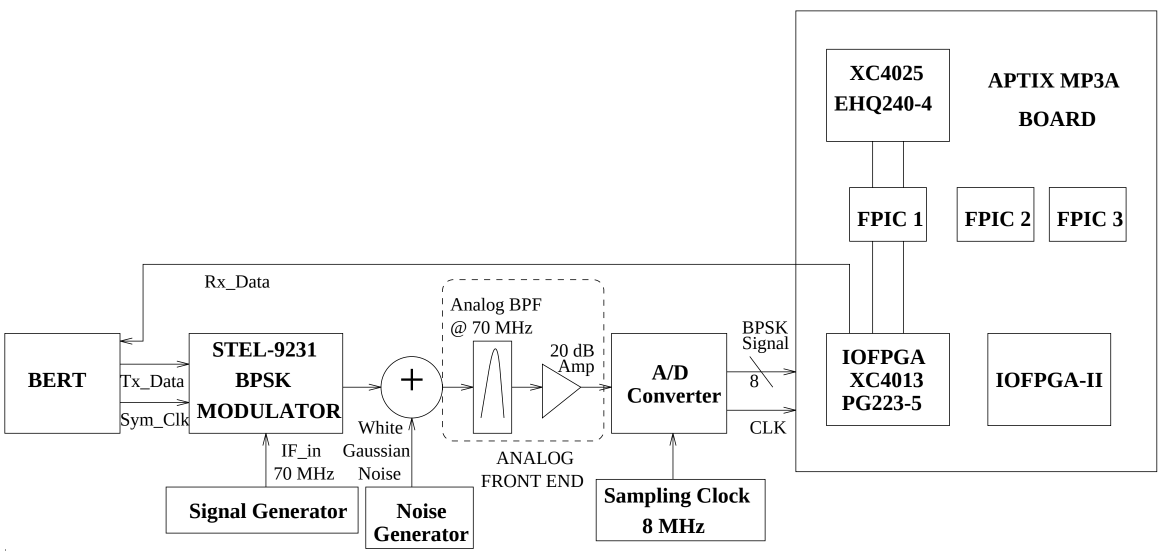

Figure 5.1: Test setup for characterizing PSK demodulators in the presence of AWGN

5.1.0.1 Analog Front End

In order to test the receiver in the presence of Additive White Gaussian Noise an analog front end was built. The Signal-to-Noise ratio is calculated at the input of the front end. The front end comprises a bandpass filter at 70 MHz with a 5 MHz bandwidth (3 dB bandwidth) and a 20 dB amplifier to push the amplitude levels into the operating range of the digital demodulator. The signal is brought down to the second IF stage that is 500 kHz by suitably sub sampling the 70 MHz IF by a 8.75 MHz clock so that it forms images at every 500 kHz and the first image is used for the rest of the processing.

5.1.0.2 E_b/N_0 calculation

We are interested in plotting the variation of the average Bit Error Rate (BER) with changing E_b / N_0. The E_b / N_0 is calculated the following way. [10]

$$E_b = PT_b \tag{5.1}$$

where P represents the *received signal data power* in watts and N_0 the one-sided PSD level of the noise. For the tests performed the signal power was about -25 dB which is about -75 dBm / Hz for a data rate of 100 kbps. The noise power is varied accordingly so as to get a ratio of 0 to 15 dB.

5.1.0.3 BER Vs E_b/N_0

The standard BER Vs E_b / N_0 curves are plotted and compared with the theoretical ones in the figure 5.2 and figure 5.3. The theoretical expression for the probability of BER as a function of E_b / N_0 is given by the formula [10]:

For Antipodal (BPSK) Signals) -

$$P(E) = Q(\sqrt{\frac{2E_b}{N_0}}) \tag{5.2}$$

For Bi-Orthogonal (QPSK) Signals) -

$$P(E) = Q(\sqrt{\frac{E_b}{2N_0}}) \tag{5.3}$$

where Q is the error function.

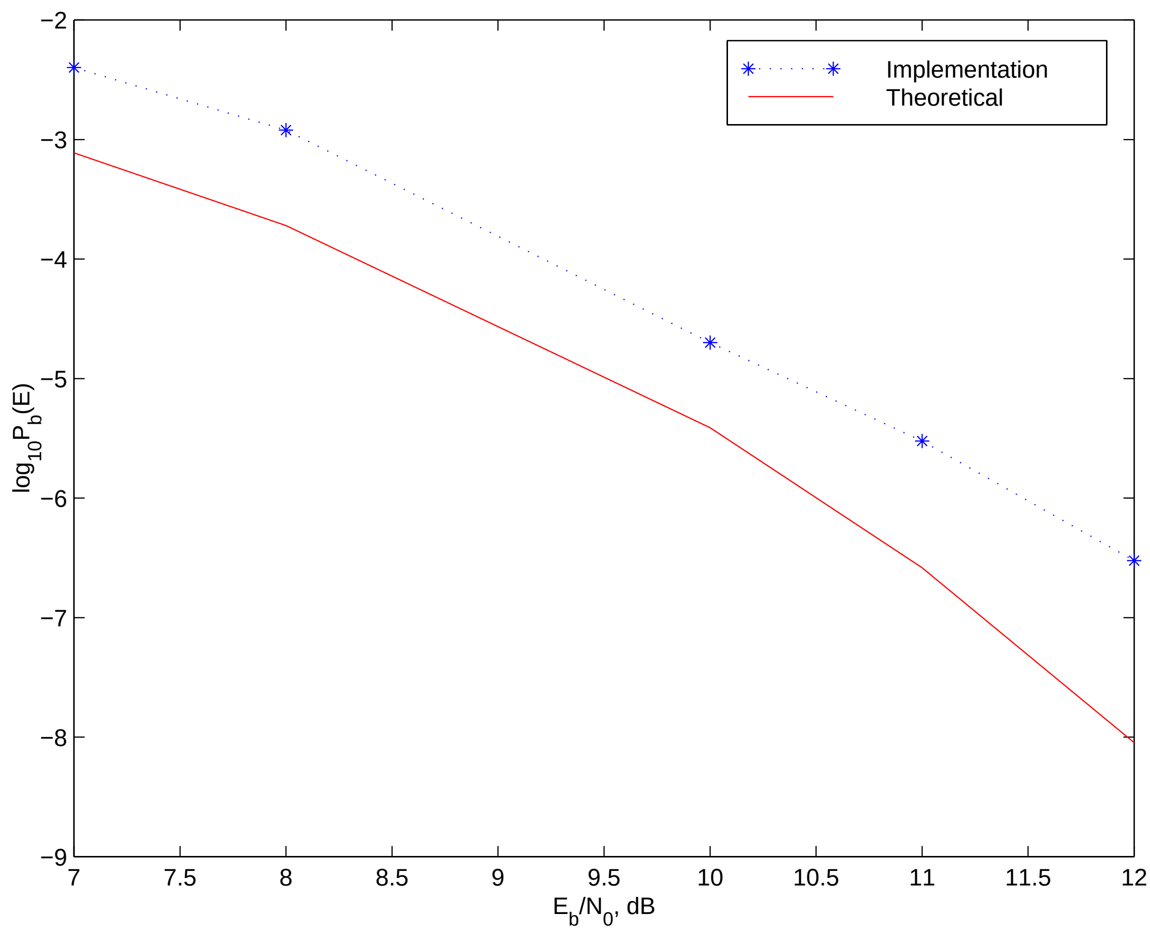

Figure 5.2: BER Vs E_b / N_0 curves for BPSK : Theoretical Vs Implementation

To compare the performance of QPSK receiver vis-a-vis the BPSK receiver both the curves are plotted in the figure 5.4.

As expected the performance of BPSK receiver is better than that of the QPSK. The-

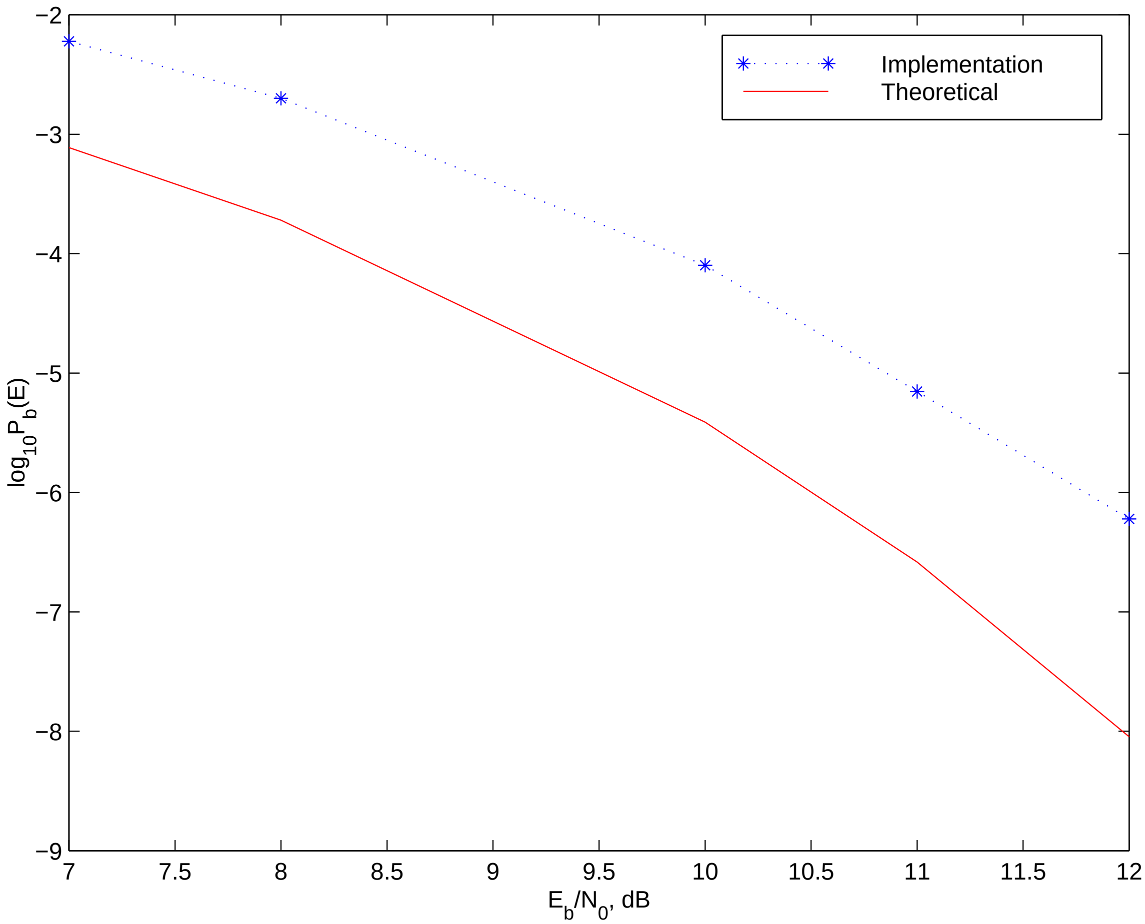

Figure 5.3: BER Vs E_b / N_0 curves for QPSK : Theoretical Vs Implementation

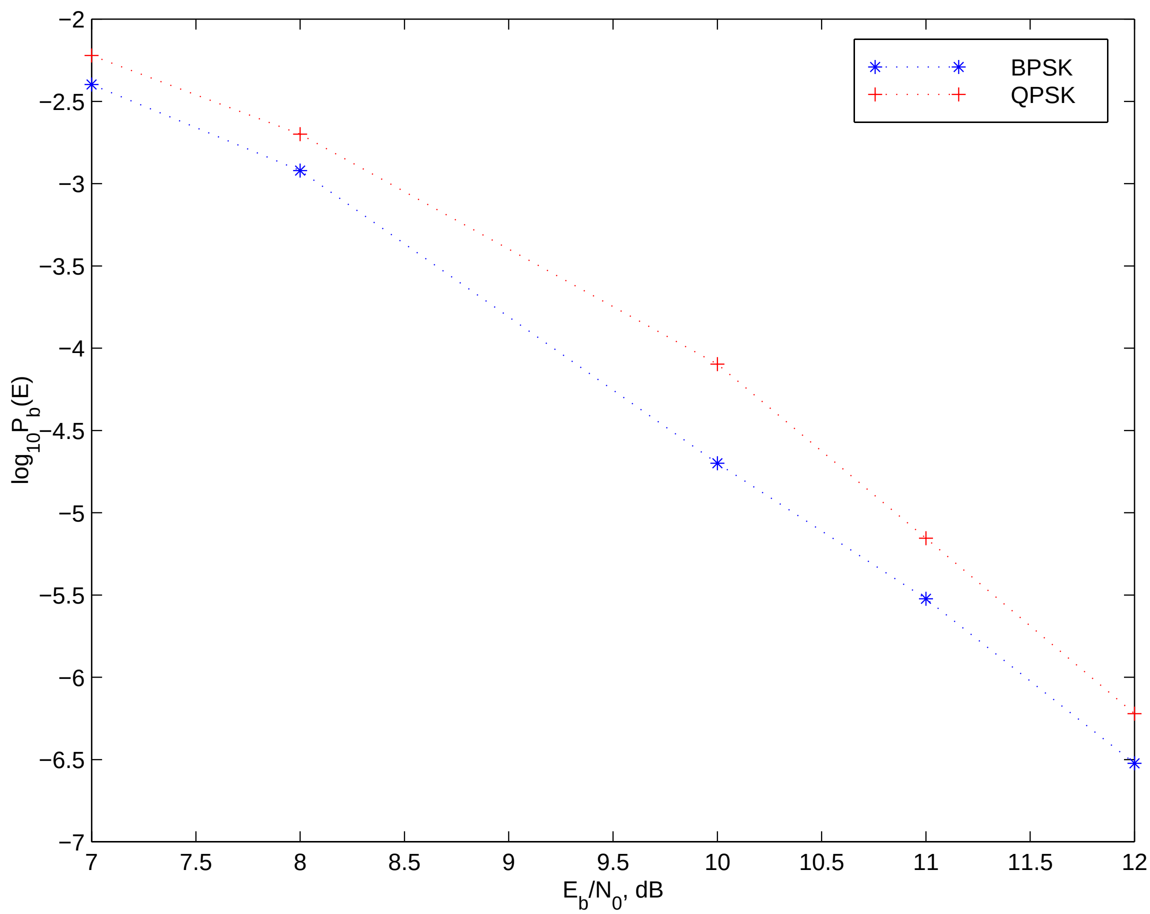

Figure 5.4: BER Vs E_b / N_0 curves for PSK : BPSK Vs QPSK

oretically the performance is expected to drop by about 3dB of SNR which means that for the same probability of error QPSK requires that the SNR or equivalently E_b/N_0 3 dB more than that would be needed for the BPSK receiver.

5.2 Testing the Reconfigurable Demodulator

The functioning of the adaptive demodulator was tested on the WILDFORCE platform about which was explained in the ”Reconfigurable Demodulator” chapter. This platform is ideally suited for reconfiguring the FPGA based on the modulation detected at the input by the modulation recognizer hooked to each of the demodulators. The testing strategy adopted is that the FIFO interface of the WILDFORCE setup is used to pump in data to the FPGA from a file and the enable signals read from the FPGA using the same interface. The state of the FPGA as to what demodulator is currently being used is maintained by the used C program and when ever the modulation changes the FPGA is reconfigured by downloading the appropriate bit file. So three different bit files are generated one for each demodulator. At start only the modulation recognizer resides in the FPGA and monitors the input signal to determine the modulation type. The decision interval is 2048 clock cycles. This figure is arrived at after extensive tests to decide on an optimum value. Though the modulation at the input signal may change between two decision intervals the reconfiguration takes place only at the end of the decision interval. This is to avoid misfiring of the download program for false changes in the sense that unless a particular modulation type is sensed for the sometime as determined by the decision interval the changes are ignored and considered false. This is important for testing because the signal should possess a particular modulation in it for the time posed by the decision interval. The data is generated using matlab and simulink and stored into a file.

5.2.1 Noise Tolerance levels for the AMR algorithm

Testing was done with the strategy explained in the previous section. This was a noise free environment. The results are as expected. To elaborate, the modulation recognizer could successfully identify the modulation type and also initiate the reconfiguring the FPGA as a appropriate demodulator. However a more useful metric while comparing the algorithm with the existing ones is to quote the noise tolerance level which usually is mentioned in dB of Signal-to-Noise ratio. The reconfigurable demodulator was tested in the presence of noise. For successfully classifying the modulation types among FSK2, PSK2 and PSK4 the proposed algorithm required a signal-to-noise ratio of 20 dB or more which is not particularly bad. Most of the classifiers referenced in this work require about 10-20 dB of SNR for 100% success rate of classification. It would be interesting to see the variation of the thresholds with SNR.

Chapter 6

Conclusions and Future Work

To summarize, this thesis documents the design and FPGA implementation of certain radio processing functions. It also demonstrates how the reconfigurability offered by FPGAs can be exploited to implement adaptive algorithms of signal processing and communications. A novel algorithm of modulation recognition has been proposed and implemented. This aided the integration of the individual demodulators into an adaptive demodulator. As opposed to more common universal demodulators which switch between resident demodulators based on the input from the modulation recognizer, it has been shown how the same FPGA can be reconfigured as a desired demodulator on the fly as the input modulation scheme changes thus saving extra silicon. The capabilities offered by the reconfigurable platform have been demonstrated which can be a promising choice for a more robust signal processing or communication system.

6.1 Future Work

Future work may extend in different directions which may include the following:

- Extension of the modulation recognition algorithm to accommodate other modulation types.

- Analysis of the effects of Signal to Noise ratio on the thresholds in the AMR algorithm to make them more noise tolerant than 20 dB.

- Exploit the reconfigurability of FPGAs to partially reconfigure them, thereby making the demodulators adapt to changing environments. Supporting a wide range of data rates would be a good example in this direction.

Bibliography

[1] Annapolis Micro Systems Inc. *WILDFORCE Reference Manual*, 1999. revision 3.4.

[2] Elsayed Elsayed Azzouz and Asoke Kumar Nandi. *Automatic Modulation Recognition of Communication Signals*. Kluwer Academic Publishers, 1996.

[3] Ken Chapman. Constant coefficient multipliers for the xc4000e. Application Note, december 1996. Xilinx Inc.

[4] Sebastiano B. Serpico Fabio Roli and Gianni Vernazza. Intelligent control of signal processing algorithms in communications. *IEEE Journal on Selected Areas in Communications*, pages 1553–1565, 1994.

[5] M. E. Frerking. *Digital Signal Processing in Communication Systems*. Chapman & Hall, 1994.

[6] Thad Genrich. Bpsk demodulator/bit synchronizer fpga implmentation and benchmarking.

[7] Chung-Yu Huamg and Andreas Polydros. Likelihood methods for mpsk modulation classification. *IEEE Transactions on Communications*, 43(2/3/4):1493–1504, 1995.

[8] D. Duponteil J. C. Bic and J. C. Imbeaux. *Elements of Digital Communications*. John Wiley and Sons, 1991.

[9] K. Farrell K. Alssaleh and R. J. Mammone. A new method of modulation classification for digitally modulated signals. *MELCOM*, pages 30.5.1–30.5.5, 1992.

[10] William C. Lindsey Marvin K. Simon, Sami M. Hinedi. *Digital Communication Techniques*. Prentice Hall, 1995.

[11] Steve P. Nicoloso. An investigation of carrier recovery techniques for psk modulated signals in cdma and multipath mobile environments. Master's thesis, Virginia Polytechnic Institute and State University, June 1997.

[12] Glenn E. Prescott. Rapid prototyping of software radio systems using fpgas and dsp microprocessors. Technical report, University of Kansas, Lawrence, 1997.

[13] Samir S. Soliman and Shue-Zen Hsue. Signal classification using statistical moments. *IEEE Transactions on Communications*, 40(5):908–916, 1992.

[14] J. L. Perry T. G. Callaghan and J. K. Tjho. Sampling and algorithms aid modulation recognition. *Microwaves RF*, 24(9):117–119, 121, 1985.

Appendix A

Filter Details

A.1 Automatic Modulation Recognizer

A.1.1 Averaging Filter

Filter Structure : FIR

Sampling Frequency : 8 MHz

Window Method : Kaiser

Tap Length : 25

-6 dB cut-off : 100 kHz

Trade Off Factor : 3.5

Coefficients :

$b_0 = b_24 = 0.008738$

$b_1 = b_23 = 0.013567$

$b_2 = b_22 = 0.019045$

$b_3 = b_21 = 0.025014$

$b_4 = b_20 = 0.031278$

$b_5 = b_19 = 0.037611$

$b_6 = b_18 = 0.043771$

$b_7 = b_17 = 0.049510$

$b_8 = b_16 = 0.054590$

$b_9 = b_15 = 0.058792$

$b_10 = b_14 = 0.061935$

$b_11 = b_13 = 0.063878$

$b_12 = b_12 = 0.063878$

A.2 BPSK Receiver

A.2.1 Carrier Recovery

A.2.1.1 Bandpass Filter

Filter Structure : FIR

Sampling Frequency : 8 MHz

Window Method : Remez Exchange

Tap Length : 21

Center Frequency : 1 MHz

Passband Width : 400 kHz

Stopband Width : 800 kHz

Stopband Weight : 8

Passband Weight : 8

Coefficients :

$b_0 = b_21 = 0.008738$

$b_1 = b_19 = 0.049727$

$b_2 = b_18 = 0.044479$

$b_3 = b_17 = 0.034744$

$b_4 = b_16 = 0.001235$

$b_5 = b_15 = -0.041227$

$b_6 = b_14 = -0.064711$

b_7 = $b_1$3 = -0.049342

b_8 = $b_1$2 = -0.000223

b_9 = $b_1$1 = 0.0.052975

$b_1$0 = $b_1$0 = 0.075775

A.2.1.2 Square to Sine Converter

Filter Structure : FIR

Sampling Frequency : 8 MHz

Window Method : Kaiser

Tap Length : 16

-6dB cut-off : 600 kHz Trade Off Factor : 3.39 Coefficients :

b_0 = $b_1$5 = 0.003834

b_1 = $b_1$4 = 0.012247

b_2 = $b_1$3 = 0.026880

b_3 = $b_1$2 = 0.047420

b_4 = $b_1$1 = 0.071685

b_5 = $b_1$0 = 0.095893

b_6 = b_9 = 0.115520

b_7 = b_8 = 0.126520

A.2.2 Data Filter

Filter Structure : FIR

Sampling Frequency : 8 MHz

Window Method : Kaiser

Tap Length : 19

-6 dB cut-off : 150 kHz

Trade Off Factor : 2.0

Coefficients :

b_0 = $b_1$8 = 0.023438

b_1 = $b_1$7 = 0.031250

b_2 = $b_1$6 = 0.039063

b_3 = $b_1$5 = 0.046875

b_4 = $b_1$4 = 0.054688

b_5 = $b_1$3 = 0.062500

b_6 = $b_1$2 = 0.062500

b_7 = $b_1$1 = 0.070313

b_8 = $b_1$0 = 0.070313

b_9 = b_9 = 0.070313

A.3 QPSK Receiver

A.3.1 Analog Front End Bandpass Filter

Center Frequency : 70 MHz

-6 dB Bandwidth : 5 MHz

A.3.2 Carrier Recovery

A.3.2.1 Bandpass Filter

Filter Structure : FIR

Sampling Frequency : 8 MHz

Window Method : Rectangular

Tap Length : 25

-6 dB Bandwidth : 800 kHz

Center Frequency : 2 MHz Coefficients :

b_0 = $b_2$4 = -0.028095

b_1 = $b_2$3 = 0.000000

b_2 = $b_2$2 = 0.000000

b_3 = $b_2$1 = 0.000000

b_4 = $b_2$0 = 0.042142

b_5 = $b_1$9 = 0.000000

b_6 = $b_1$8 = 0.090916

b_7 = $b_1$7 = 0.000000

b_8 = $b_1$6 = 0.013637

b_9 = $b_1$5 = 0.000000

$b_1$0 = $b_1$4 = -0.168570

$b_1$1 = $b_1$3 = 0.000000

$b_1$2 = $b_1$2 = 0.180190

A.3.2.2 Square to Sine Converter

Filter Structure : FIR

Sampling Frequency : 8 MHz

Window Method : Kaiser

Tap Length : 16

-6dB cut-off : 600 kHz

Trade Off Factor : 3.39

Coefficients :

b_0 = $b_1$5 = 0.003834

b_1 = $b_1$4 = 0.012247

b_2 = $b_1$3 = 0.026880

b_3 = $b_1$2 = 0.047420

$b_4 = b_1 1 = 0.071685$

$b_5 = b_1 0 = 0.095893$

$b_6 = b_9 = 0.115520$

$b_7 = b_8 = 0.126520$

A.3.3 Hilbert Transform Filter

Filter Structure : FIR

Sampling Frequency : 8 MHz

Tap Length : 4

Coefficients :

$b_0 = 0.671875$

$b_1 = 0.406250$

$b_2 = -0.406250$

$b_3 = -0.671875$

A.3.4 Data Filter

Filter Structure : FIR

Sampling Frequency : 8 MHz

Window Method : Kaiser

Tap Length : 19

-6 dB cut-off : 150 kHz

Trade Off Factor : 2.0

Coefficients :

$b_0 = b_1 8 = 0.023438$

$b_1 = b_1 7 = 0.031250$

$b_2 = b_1 6 = 0.039063$

$b_3 = b_15 = 0.046875$

$b_4 = b_14 = 0.054688$

$b_5 = b_13 = 0.062500$

$b_6 = b_12 = 0.062500$

$b_7 = b_11 = 0.070313$

$b_8 = b_10 = 0.070313$

$b_9 = b_9 = 0.070313$