COGNITIVE MIMO RADIO

By

NC Abeera Naveed

NC Zaib un Nisa Hafeez

GC Haider Ali

Project Supervisor: Lt. Col Dr. Imran Rashid

Submitted to the Faculty of Electrical Engineering Department,
National University of Sciences and Technology, Islamabad in partial fulfillment for
the requirements of a B.E Degree in Telecommunication Engineering

JUNE 2013

ABSTRACT

COGNITIVE MIMO RADIO

The supply of available radio frequency spectrum is in a state of shortage. Whereas demand for spectrum in the most useful frequencies exceeds supply and some statically allocated spectrum bands experience low utilization. Currently the spectrum assignment for the exclusive use to licensed services is highly inefficient. This is due to the high variability of the traffic statistics across time, space and frequency.

Basically the spectrum usage is typically concentrated over certain portions of the spectrum, while a significant amount of the licensed bands (or idle slots in static time division multiple access (TDMA) systems with bursty traffic) remains unused or under-utilized for 90% of time.

A Cognitive Radio (CR) is a technology that is capable of exploiting unused or lightly used spectrum and is a promising mechanism to achieve efficient use of the frequency resource by allowing the coexistence of licensed (primary) and unlicensed (secondary) users in the same bandwidth.

This project was proposed with the aim to design a prototype for unlicensed secondary users (SUs) to dynamically access the unused white spaces in the licensed spectrum held by primary users (PUs) in order to increase the efficiency of spectrum utilization. Multiple Input Multiple Output (MIMO) technology is used in our Cognitive Radio system as MIMO system prominently increases the spectral efficiency of a wireless system.

In our project, we consider a simple cognitive MIMO system, in which the cognitive link has two transmission antennas and two receive antennas. By monitoring the activity of the primary user through spectrum sensing technique, the cognitive link lets the secondary user to communicate when it senses an idle slot. MIMO uses spatial

multiplexing as the transmission technique to provide additional data capacity.

This prototype enables us to use the available natural frequency resource efficiently and allows the secondary user to transmit on the same available bandwidth without paying revenue.

CERTIFICATE OF CORRECTNESS AND APPROVAL

It is certified that the work contained in this thesis titled “Cognitive MIMO Radio”, carried out by Abeera Naveed, Zaib un Nisa Hafeez and Haider Ali under the supervision of Assoc. Prof. Dr. Imran Rashid in partial fulfillment of the Bachelors of Telecommunication Engineering, is correct and approved.

Approved By

____

Asst. Prof. Dr. Imran Rashid

Project Supervisor

Military College of Signals, NUST

Dedication

Almighty Allah for His blessings,

Teachers and friends for their help,

And Our Parents for their support and prayers

ACKNOWLEDGEMENT

All praises for ALLAH All Mighty who enlightened us with the requisite knowledge on portion of this subject enabling us to accomplish this extremely challenging and gigantic task.

We would like to express our gratitude towards our advisor Dr. Imran Rashid for all his help, invaluable guidance, critics and generous support throughout our final year project.

Special acknowledgements to all teachers at MCS, NUST for helping us. Their interest in this project was very beneficial and helped design many vital parts of the project.

We would also like to thank all the lab staff, our friends and all those, whoever has helped us either directly or indirectly, in the completion of our final year project and thesis.

LIST OF ABBREVIATIONS

GUI	Graphical User Interface
MCS	Military College of signals
CR	Cognitive Radio
USRP	Universal Software Radio Peripheral
TX	Transmitter
RX	Receiver

Table of Contents

LIST OF ABBREVIATIONS vii

CHAPTER 1 1

1.1 Introduction 1

1.2 Problem Statement and Solution 1

1.3 Project Objectives and End Goals 2

1.4 Academic Objectives 3

1.5 Project Scope 3

1.6 Approach and Platform Used 3

1.6.1GNU Radio 4

1.6.2 MATLAB/C 5

1.6.3 Operating System 5

1.6.3 Tools and Technologies 5

1.6.4 Requirements 5

CHAPTER 2 6

2.1 Design 6

2.2 Procedure 7

2.3 MIMO 8

2.3.1 Introduction 8

2.3.2 Conventional Radio Systems (SISO) 9

2.3.3 Multiple Antenna Systems 9

2.4 Implementing 2x2 MIMO on USRP 12

CHAPTER 3 37

3.1 IMPLEMENTATION ON MATLAB 37

3.1.1 SISO 37

3.1.2 SIMO 37

3.1.3 MISO 38

3.1.4 MMSE, ZF Comparison 38

3.1.5 VBLAST DETECTION 39

3.1.6 VBLAST –ZF DETECTION OSIC AND ZF DETECTION 40

3.1.7 COMPARISON OF FPS OF SISO AND MIMO LINK 40

3.2 Receiver Operating Characteristics (ROC) 41

CHAPTER 4 43

4.1 SUMMARY 43

4.2 FUTURE WORK 44

4.3 CONCLUSION 45

BIBLIOGRAPHY 46

APPENDIX –A 47

Table of Figures

Figure 1-1 Approach and Platform used 4
Figure 2-1 Transmitter Side 6
Figure 2-2 Receiver Side 6
Figure 2-3 SISO 9
Figure 2-4 MIMO 10
Figure 2-5 2x2 MIMO Link 10
Figure 2-6 OpenCV Structure andContents 14
Figure 2-7 QPSK Mapping 15
Figure 2-8 VBLAST Technique 17
Figure 2-9 VBLAST Detection Diagram 24
Figure 2-10 ZF VBLAST and ML VBLAST Comparison 26
Figure 2-12 Flow Chart for Cognitive Transmitter 29
Figure 2-13 Parallel Detection and Transmission 31
Figure 3-1 SISO AWGN and Rayleigh Comparison 37
Figure 3-2 SIMO 38
Figure 3-3 MISO 38
Figure 3-4 ZF and MMSE Comparison 39
Figure 3-5 VBLAST-OSIC Detection 39
Figure 3-6 VBLAST ZF-OSIC and ZF Detection Comparison 40
Figure 3-7 Receiver Operating Characteristic (ROC) 42

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter will give a brief overview of the problem statement and the identification of the solution to that problem through the implementation of this project. It will also brief about the environment in which the implementation of Cognitive MIMO Radio is done.

1.2 Problem Statement and Solution

Current wireless networks are characterized by a static spectrum allocation policy, where governmental agencies assign wireless spectrum to license holders on a long-term basis for large geographical regions. Recently, because of the increase in spectrum demand due to the introduction of new wireless communication technologies (3G, 4G etc) and with the growing popularity of smart phones and the increasing use of applications designed to make use of their capabilities, traffic is rising dramatically. The resource requirements vary greatly over time and between cells and frequency layers. At any one time, many parts of the network have significant free resources, while other parts need to deliver high data speeds. Underused resources are common in a typical network. This is inefficient for network operators.

Thus this policy faces spectrum scarcity in particular spectrum bands. In contrast, a large portion of the assigned spectrum is used sporadically, leading to underutilization of a significant amount of spectrum.

Dynamic spectrum access technique is proposed to solve these spectrum inefficiency problems.

The key enabling technology of dynamic spectrum access techniques is cognitive radio (CR) technology, which provides the capability to share the wireless channel with licensed users in an opportunistic manner. CR envisions to provide the secondary user the opportunity to communicate on the same bandwidth when it detects a spectrum hole through spectrum sensing technique. A spectrum hole is defined as a band of frequencies assigned to a primary user, but, at a particular time and specific geographic location, the band is not being utilized by that user.

The MIMO technology is used because it solves the problem of multipath fading, disallowing the signals to distort to great extent. While in case of the SISO system, it allows data rates to be normal and does not allow correction for the fading of signals. On the other hand, in case of MIMO the data rates are increased and on the whole the communication becomes better than the SISO system. Multiple Input Multiple Output (MIMO) uses the physical presence of multiple transmits and receive antennas to achieve higher data rates and reliability by fighting off fading and multipath interference effects.

1.3 Project Objectives and End Goals

The aim of the project is to develop a 2x2 MIMO based Cognitive Radio. Spectrum sensing technique is used for detection of spectrum holes. The prototype works on UHF frequency (around 2.4 GHz).

Performance of the system will be compared with a SISO system and practically will be shown that 2x2 MIMO has double the data rate than that of SISO. Also, Cognitive technique (spectrum sensing) implementation will be shown.

1.4 Academic Objectives

1. To apply the knowledge gained about DSP, Programming, hardware designing and other related subjects in a practical system.
2. To know how real world communication is done and how it is affected by different factors and learn how to minimize the effect of those factors which degrade the quality of communication.
3. Learn how to design a complete end to end wireless communication system.

1.5 Project Scope

A 2 x 2 MIMO based system was first developed in MATLAB. VBLAST technique was utilized as a reception technique. A working model was checked for its BER performance.

Energy detection based cognitive radio sensing algorithm was developed which dynamically sensed various frequency channels of a primary user’s licensed band. The algorithm indicated the free (underutilized) frequency channels to the system.

MIMO system chose one of the channels for its cognitive transmission.

This complete algorithm is now developed in Python and is embedded in 2 x USRP kits. The system uses low transmission power for its MIMO link keeping the interference aspects very low towards the main primary user.

An underutilized channel reuse helps in better management of precious frequency utilization.

1.6 Approach and Platform Used

First of all the literature review for the project was done. The project was understood and methods were devised for the implementation of the project.

Secondly the implementation of various techniques was done in MATLAB and comparison of those techniques was carried out.

Then finalization of the technical design and design requirement was done.

The application of the final design is done in GNU Radio environment and integration is done on the USRPs.

Finally the hardware-software integrated prototype model is created.

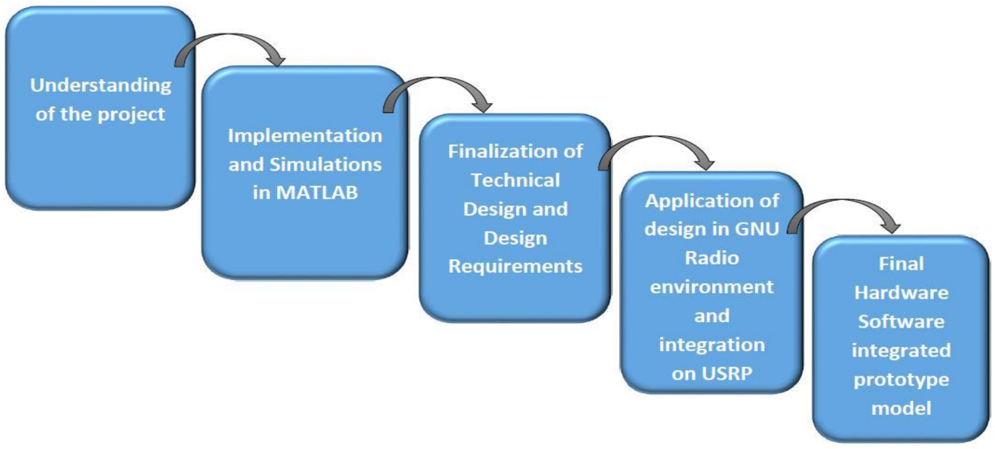

Understanding of the project
Implementation and Simulations in MATLAB
Finalization of Technical Design and Design Requirements
Application of design in GNU Radio environment and integration on USRP
Final Hardware Software integrated prototype model

Figure 1-1 Approach and Platform used

1.6.1GNU Radio

Various open ended operating systems have provided the provision for developers to write blocks for specialized purposes. Ubuntu is one such Linux based open ended operating system in which a block for communications has been developed named GNU RADIO. The block has been written in Python which is a widely acclaimed programming language. Simulation of implemented systems can be done using sub-blocks from GNU. These have the advantage of being specially developed for communications. Required fields can be appended with existing blocks using Python.

1.6.2 MATLAB/C

The simulation can also be carried out in a MATLAB environment which has an un parallel library of mathematical and communication functions.

1.6.3 Operating System

Microsoft Windows and Linux (UBUNTU).

1.6.3 Tools and Technologies

MATLAB, GNURADIO, Python and Microsoft Visual C++.

1.6.4 Requirements

1. Programming skills (C++, Python).
2. Knowledge about GNU radio and USRP.
3. OPENCV
4. Linux (Ubuntu)
5. Knowledge about modulation schemes.
6. Details of the equipment required to develop the subject facility is as under

Sr. no	Equipment	Quantity
1.	USRP1	2
2.	Daughter cards	4
3.	General Purpose Computers	2

Table 1-1 Equipment required for Project

CHAPTER 2

Design and Procedures

2.1 Design

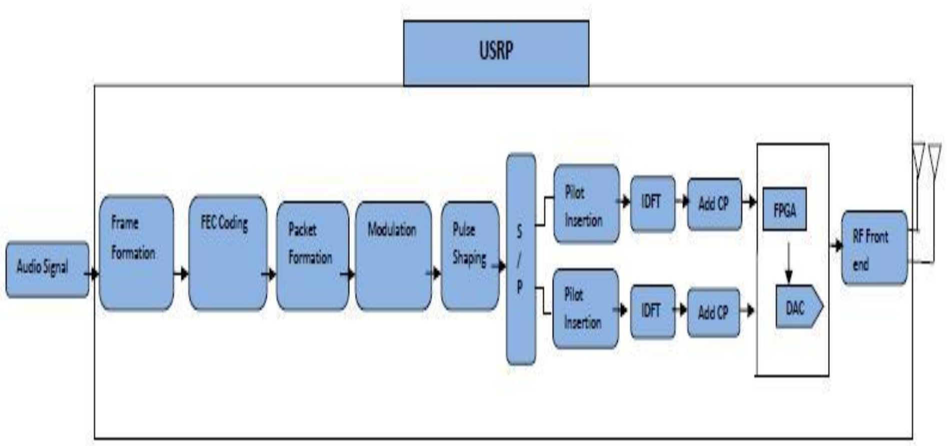


Figure 2-1 Transmitter Side

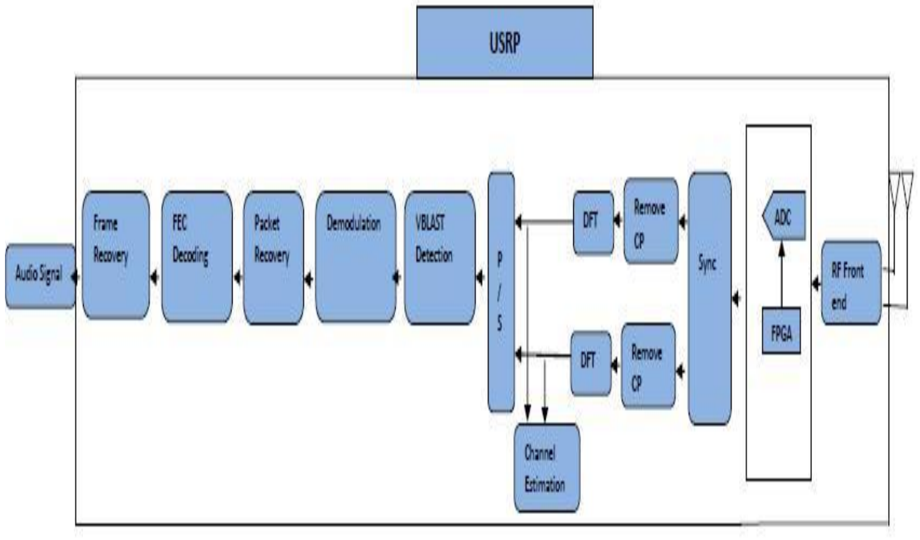


Figure 2-2 Receiver Side

2.2 Procedure

The main goal of this project is to achieve real-time video transmission over a free frequency channel, detected as a result of spectrum sensing technique, employing MIMO transmission which enables high data rate. The basic idea is that sensing of the free frequency channel is done using energy detection technique and then for the transmission on the free available channel, the input video sequence is divided into frames. Each frame is then converted to bit stream. The resulting bit stream is processed to be transmitted over the MIMO link.

Then we will setup a practical MIMO platform for the real-time transmission analysis of the video.

USRPs will be used as front ends and thus are needed to complete this platform.

The modulation scheme used will be QPSK.

Cognitive radios intelligently optimize their own performance in response to user requests. Cognitive radio transmits on a piece of spectrum found not utilized by the primary user.

VBLAST technique will be applied for detection. Using ZF VBLAST Detection, the computational and implementation complexities are removed as one can see from the the ZF equation for removing the channel affect

$$\hat{a} = (H*H)-1Hx = H+x$$

The ZF receiver converts the joint decoding problem into M single stream decoding problems thereby significantly reducing receiver complexity.

A well thought model of the hardware implementation is based on the USRP boards with protocol development in C/Python in GNU radio. The project once completed would provide students and faculty for further research and development in MIMO and Cognitive Networks.

Now we will discuss different topics along with the procedures involved in the project in the next sections.

2.3 MIMO

Regardless of whether mobile radio networks like 3GPP related technologies or wireless radio networks like WLAN, all radio communications systems must continually provide higher data rates. In addition to conventional methods, such as introducing higher modulation types or providing larger bandwidths, this is also being achieved by using multiple antenna systems (Multiple Input, Multiple Output – MIMO). This application note gives an introduction to basic MIMO concepts and terminology and explains how MIMO is implemented in the different radio communications standards.

2.3.1 Introduction

It is obvious that the main goals in developing next generations of wireless communication systems are increasing the link throughput (i.e., bit rate) and the network capacity. Since radio propagation conditions appear to limit the realm of wireless and mobile communication systems to the range around 1 GHz to 6 GHz, the available frequency spectrum is limited. So to fulfill the above goals, future systems should be characterized by improved spectral efficiency. The use of multiple transmit and receive antennas can improve the performance of wireless communication systems significantly by providing higher data rates and higher spectral efficiency.

2.3.2 Conventional Radio Systems (SISO)

Conventional systems use one transmit and one receive antenna. In MIMO terminology, this is called Single Input, Single Output (SISO, in figure 2.3-1).

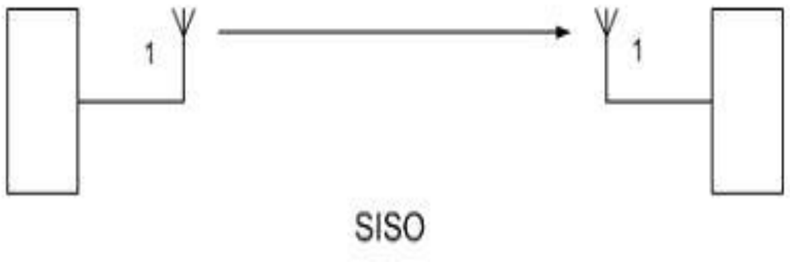


Figure 2-3 SISO

2.3.3 Multiple Antenna Systems

A MIMO system typically consists of m transmit and n receive antennas. By using the same channel, every antenna receives not only the direct components intended for it, but also the indirect components intended for the other antennas. A time-independent, narrowband channel is assumed. The direct connection from antenna 1 to 1 is specified with h11, etc., while the indirect connection from antenna 1 to 2 is identified as cross component h21, etc. From this is obtained transmission matrix H with the dimensions n x m.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{..} & h_{1m} \\ h_{21} & h_{22} & h_{..} & h_{2m} \\ h_{..} & h_{..} & h_{..} & h_{.m} \\ h_{n1} & h_{n2} & h_{n.} & h_{nm} \end{bmatrix}$$

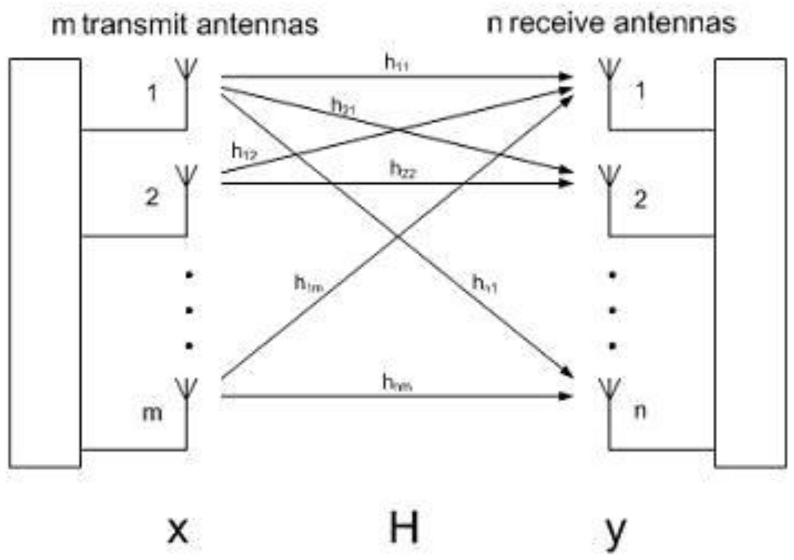


Figure 2-4 MIMO Link

2.3.3.1 2x2 MIMO

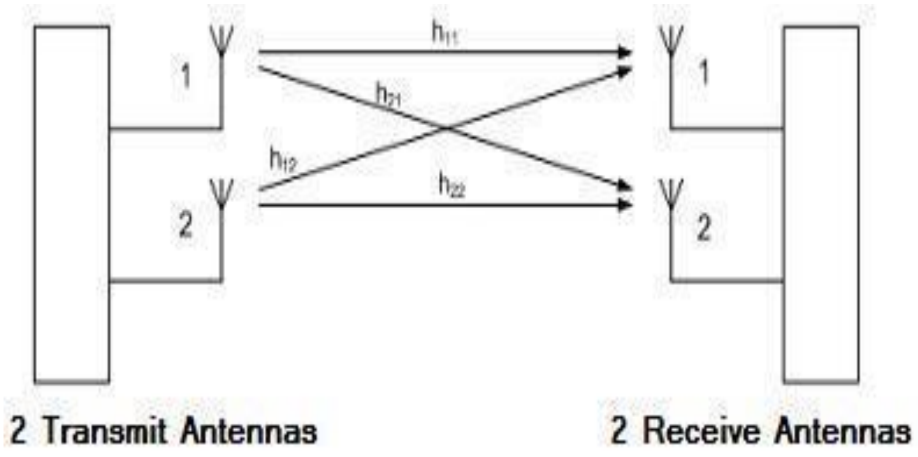


Figure 2-5 2x2 MIMO Link

This is a 2x2 MIMO system in which there are two transmitting and two receiving antennas. During the transmission both transmitting antennas send their respective data to both of the receiving antennas hence forming four channels, two direct channels identified as h11 and h22, and indirect channels are identified as h12 and h21. Comparing this system with a Single Input Single Output (SISO) system, it can be proven that the data rate increases two times theoretically and 1.8 to 1.9 times practically in a 2x2 MIMO system.

2.3.3.2 Advantages of MIMO

MIMO Systems are more Resistive to Fading

Advantages of MIMO are increased throughput and robustness of the data link. MIMO systems are more resistive to fading which results in better Quality of Service. Other advantages include increased coverage, increase capacity, increased data rate, improved spectral efficiency and reduced power consumption.

Increases Throughput

MIMO supports enhanced data throughput even under conditions of interference, signal fading, and multipath. The demand for higher data rates over longer distances has been one of the primary motivations behind the development of communications systems.

For years, engineers assumed that the theoretical channel capacity limits were defined by the Shannon- Hartley theorem illustrated in Eq.:

$$\text{Capacity} = BW \times \log 2\ (1 + SNR)$$

As the above equation shows, increase gains in channel throughput. As a result, the traditional way to achieve higher data rates is by increasing the signal bandwidth. Unfortunately, increasing the signal bandwidth of a communications channel by increasing the symbol rate of a modulated carrier increases its susceptibility to multipath fading. System throughput is increased due to antenna systems used at transmitter and receiver ends, which means higher obtainable data rates with reduced overall transmitted power and reduced necessary receiver sensitivity. This causes significant reduction of emission and enables more frequent reuse of frequency.

Spectral Efficiency

MIMO communications channels provide an interesting solution to the multipath challenge by requiring multiple signal paths. In effect, MIMO systems use a combination of multiple antennas and multiple signal paths to gain knowledge of the communications channel. By using the spatial dimension of a communications link, MIMO systems can achieve significantly higher data rates than traditional single-input, single-output (SISO) channels. Using channel knowledge, a receiver can

recover independent streams from each of system produces two spatial streams to effectively double the maximum data rate of what might be achieved in a traditional SISO communications channel.

2.4 Implementing 2x2 MIMO on USRP

1. A video is taken as input from the OpenCV software inside the Python code.

OpenCV

Handling of this video data and frames is not directly supported in python. So another tool was required for this purpose. After a bit of researching into this field it was found that OpenCV is a great tool that can be used. OpenCV is an open source computer vision library. The library is written in C and C++ and runs under Linux, Windows and Mac OS. There is active development on interfaces for Python, Ruby, MATLAB, and other languages. Primarily developed for the purpose of image processing, the library was utilized as an advantage in this very project. The main issue was how to integrate python with C++ as it is the language of OpenCV. A wrapper which integrates the modules of OpenCV with those of python is SWIG or Simplified Wrapper Interface Generator can generate the interface between C++ and Python programming languages. It generates the equivalent C++ code in python (.py) files. In addition to that, to make the OpenCV modules fully functional in python the GNU radio package named python-OpenCV was installed, available on the PPA Launch pad standard repository.

Why OpenCV?

Importing video in the GNU radio python was a difficult task as python itself did not provide with the option to do it straight away. Initially the g streamer built in player was used to manipulate with the video data but that turned out to be too complex. OpenCV proved to be the perfect tool to import video in GNU radio python. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. OpenCV is written in optimized C and can take advantage of multi-core processors.

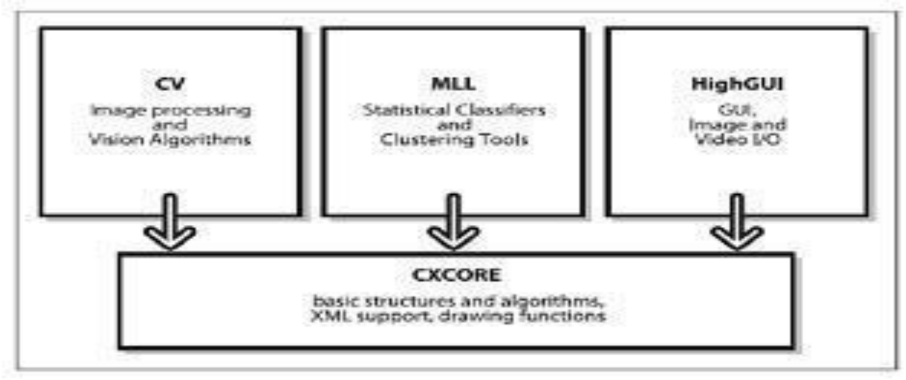

CV
Image processing and Vision Algorithms
MLL
Statistical Classifiers and Clustering Tools
HighGUI
GUI, Image and Video I/O
CXCORE
basic structures and algorithms, XML support, drawing functions

Figure 2-6 OpenCV Structure and Contents

1. The video is divided into frames.
2. The frames are divided into bits.
3. Bits are converted to QPSK symbols and are transmitted using two 2.4 GHz daughter cards, as it is 2x2 MIMO.
4. The symbols with the strongest SNR are detected (coming from the same antenna).
5. Symbols are demodulated to get the vectors of bits.
6. Channel is estimated and is used for VBLAST Detection.
7. We get two columns of the channel matrix and get the received vector having the noise portion as well.
8. The vectors are ordered first and then moved to the VBLAST phase and go through the VBLAST Detection.

QPSK Modulation

Quadrature Phase Shift Keying (QPSK) is the digital modulation technique. Quadrature Phase Shift Keying (QPSK) is a form of Phase Shift Keying in which two bits are modulated at once, selecting one of four possible carrier phase shifts (0, Π/2, Π, and 3Π/2).

QPSK perform by changing the phase of the In-phase (I) carrier from 0° to 180° and the Quadrature-phase (Q) carrier between 90° and 270°. This is used to indicate the four states of a 2-bit binary code. Each state of these carriers is referred to as a Symbol.

Quadrature Phase-shift Keying (QPSK) is a widely used method of transferring digital data by changing or modulating the phase of a carrier signal. In QPSK digital data is represented by 4 points around a circle

which correspond to 4 phases of the carrier signal. These points are

called symbols.

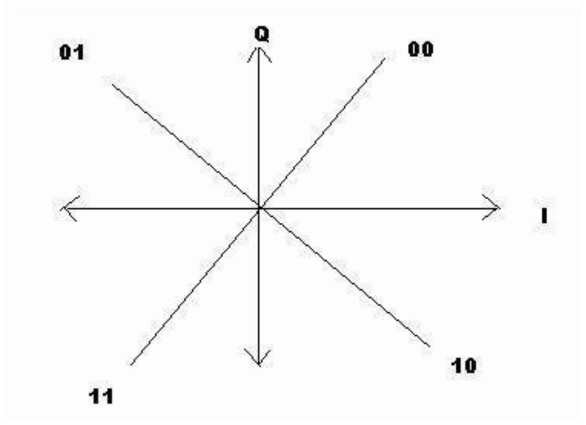


Figure 2-7 QPSK Mapping

VBLAST DETECTION

INTRODUCTION

During the past years, theoretical evaluations have shown that the multipath wireless channel is capable of having enormous capacities only if the multipath scattering is richly provided and is properly used by using appropriate architecture. Foschini proposed the diagonally-layered space-time architecture, now known as diagonal BLAST (Bell Laboratories Layered Space Time) or DBLAST, is one such way of approach. DBLAST uses multi antenna arrays at both transmitter and receiver and diagonally layered coding structure. In this structure the code blocks are dispersed across diagonals in space-time. In a Rayleigh scattering environment, this diagonal structure leads to rates (theoretical) which grow linearly as the number of antennas increase (assuming equal numbers of transmit and receive antennas), with these rates extending to 90% of Shannon capacity. However, the diagonal procedure comes across some implementation problems which make it inappropriate for initial implementation. A simplified version of BLAST known as vertical BLAST or VBLAST, which has been implemented in real-time in the laboratory, is described.

System Overview

A high-level block diagram of a BLAST system is shown in the figure. A single data stream is distributed into M sub streams, and each sub stream is then transformed into symbols and given to its respective transmitter. All transmitters operate co-channel at symbol rate 1/T sym/sec, with synchronized symbol timing. Each transmitter is itself an ordinary QAM transmitter.

The transmitters comprise, in effect, a vector-valued transmitter, where elements of each transmitted M-vector are symbols drawn from a QAM constellation. Assuming that the same constellation is used for each sub stream, and that transmissions are organized into bursts of some symbols. The power radiated by each transmitter is proportional to 1/M so that the total launched power is constant and independent of M.

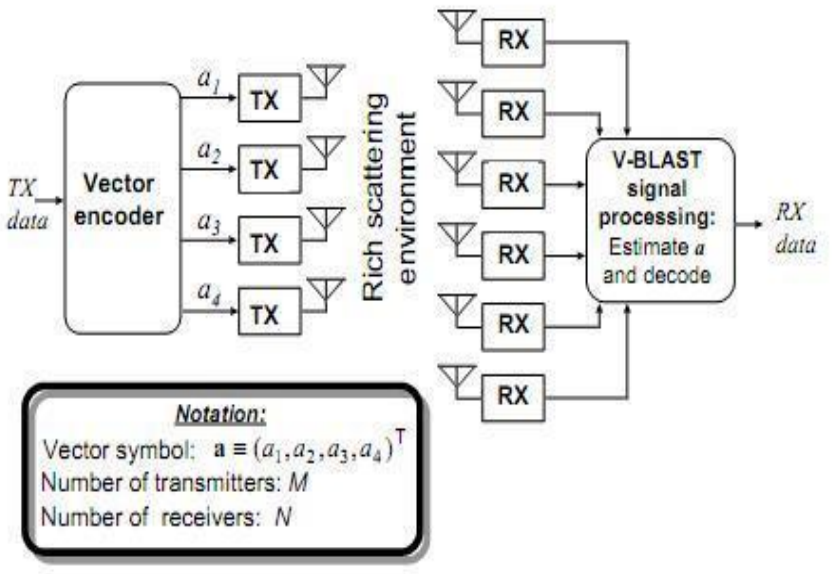

TX data
Vector encoder
a_1
a_2
a_3
a_4
TX
TX
TX
TX
Rich scattering environment
RX
RX
RX
RX
RX
RX
V-BLAST signal processing: Estimate a and decode
RX data
Notation:
Vector symbol: $\mathbf{a} \equiv (a_1, a_2, a_3, a_4)^T$
Number of transmitters: M
Number of receivers: N

Figure 2-8 VBLAST Technique

All receivers are, individually, QAM receivers. These receivers also operate co-channel, each receiving the signals radiated from each M transmit antennas. For simplicity in the process, fading flat is assumed, and the matrix channel is depicted as HNxM, where hij is the (complex) transfer function from transmitter j to receiver i, and M less or equal to N. The main difference between DBLAST and VBLAST lies in the vector encoding process. In DBLAST, redundancy is introduced between the sub streams through the use of specialized inter-sub stream block coding.

Code blocks of DBLAST are organized along diagonals in space-time. It is this coding that leads to DBLAST's higher spectral efficiencies for a given number of transmitters and receivers. In VBLAST, however, the vector encoding process is simply a demultiplex operation followed by bit-to-symbol mapping of each sub stream. No inter-sub stream coding, or coding of any kind, is required; however conventional coding of the individual sub streams may certainly be applied.

Although VBLAST, as shown above, is essentially a single-user system which uses multiple transmitters, one can naturally ask in what ways the BLAST approach differs from simply using traditional multiple access techniques in a single-user fashion, i.e. by driving all the transmitters which has been split into from sub streams. Some of these differences are worth pointing out: First, unlike code division or other spread-spectrum multiple access techniques, the total channel bandwidth utilized in a BLAST system is only a small fraction in excess of the symbol rate, i.e. similar to the excess bandwidth required by a conventional QAM system. Second, unlike FDMA, each transmitted signal occupies the entire system bandwidth. Finally, unlike TDMA, the entire system bandwidth is used simultaneously by all of the transmitters all of the time.

Taken together, these differences together are precisely what give BLAST the potential to realize higher spectral efficiencies than the multiple-access techniques. In fact, an essential feature of BLAST is that no explicit orthogonalization of the transmitted signals is imposed by the transmit structure at all. Instead, the propagation environment itself, which is assumed to exhibit significant to achieve the signal decorrelation necessary to separate the co-channel, signals. V-BLAST utilizes a combination of old and new detection techniques to separate the signals in an efficient manner, permitting operation at significant Shannon capacity and achieving large spectral efficiencies in the process.

V-BLAST Detection

In what follows, we take a discrete-time baseband view of the detection process for a single transmitted vector symbol, assuming symbol-synchronous receiver sampling and ideal timing. Letting a = (a1, a2, . . .,aM)^T denote the vector of transmit symbols, then the corresponding received N-vector is:

$$r1 = Ha1 + \nu$$

where ν is a noise vector with components drawn from wide-sense stationary processes with variance σ2.

Conceptually, each sub stream in turn is considered to be the desired signal, and the remainder are considered as "interferers". Nulling is performed by linearly weighting the received signals so as to satisfy some performance-related criterion, such as minimum mean-squared error (MMSE) or zero-forcing (ZF). For example, zero-forcing nulling can be performed by choosing weight vectors wi where, i = 1, 2, . . ., M, such that

$$wi^{T} (H)j = \delta ij$$

where (H)j is the j-th column of H, and δ is the Kronecker delta. Thus, the decision statistic for the i–th substream is

$$yi = wiTr1.$$

Use symbol cancellation as well as linear nulling to perform detection. Using symbol cancellation, interference from already-detected components of a is subtracted out from the received signal vector, resulting in a modified received vector in which, effectively, fewer interferers are present. This is somewhat analogous to decision feedback equalization.

When symbol cancellation is used, the order in which the components of a are detected becomes important to the overall performance of the system. We first discuss the general detection procedure with respect to an arbitrary ordering. Let the ordered set

$$S \equiv 1\{k,2,. . ., kM\}$$

be a permutation of the integers 1, 2, . . . , M specifying the order in which components of the transmitted symbol vector a are extracted. The detection process proceeds generally as follows:

***Step 1*:**

Using nulling vector w_{k1}, form decision statistic y_{k1}:

$$y_{k1} = w_{ki}^{T} r_1$$

***Step 2*:**

Slice y_{k1} to obtain $\hat{a}_{k1}$:

$$\hat{a}_{k1} = Q\,(y_{k1})$$

where Q(.) denotes the quantization (slicing) operation.

***Step 3*:**

Assuming that $\hat{a}_{k1} = a_{k1}$, cancel a_{k1} from the received vector r_1, resulting in modified vectorreceived$_2$:

$$r_2 = r_1 - \hat{a}_{k1}\,(H)_{k1}$$

where $(H)_{k1}$ denotes the k_1-th column of H. Steps 1 - 3 are then performed for components k_2, . . . , k_M by operating in turn on the received vectors r_2, r_3, . . . ,r_M.

The full ZF V-BLAST detection algorithm can now be described compactly as a recursive procedure, including determination of the optimal ordering, as follows:

```
initialization:
    $i \leftarrow 1$
    $\mathbf{G}_1 = \mathbf{H}^+$
    $k_1 = \underset{j}{\operatorname{argmin}} \|(\mathbf{G}_1)_j\|^2$

recursion:
    $\mathbf{w}_{k_i} = (\mathbf{G}_i)_{k_i}$
    $y_{k_i} = \mathbf{w}_{k_i}^{\mathsf{T}} \mathbf{r}_i$
    $\hat{a}_{k_i} = Q(y_{k_i})$
    $\mathbf{r}_{i+1} = \mathbf{r}_i - \hat{a}_{k_i} (\mathbf{H})_{k_i}$
    $\mathbf{G}_{i+1} = \mathbf{H}_{\overline{k_i}}^+$
    $k_{i+1} = \underset{j \notin \{k_1 \cdots k_i\}}{\operatorname{argmin}} \|(\mathbf{G}_{i+1})_j\|^2$
    $i \leftarrow i + 1$
```

We implemented this algorithm on MATLAB and then on Python to get the results. The results of ZF and ZF VBLAST were plotted and compared.

Implementing ZF VBLAST having MIMO on USRP

We have implemented ZF VBLAST Detection described above. Considering from the start of the procedure:

1. A video is taken as input from the OpenCV software inside the Python code.
2. The video is divided into frames.
3. The frames are divided into bits.
4. Bits are converted to QPSK symbols and are transmitted using two 2.4 GHz

daughter cards as it is 2x2 MIMO.

5. The symbols with the strongest SNR are detected (coming from the same antenna).

6. Symbols are demodulated to get the vectors of bits.
7. Channel is estimated and is used for VBLAST Detection.
8. We get two columns of the channel matrix and get the received vector having the noise portion as well.
9. The vectors are ordered first and then moved to the VBLAST phase.
10. Following procedure takes place within the VBLAST department:

    a. First the weight symbols are found make the decision vectors out of them.

    b. Slicing (Quantization) takes place of the decision vectors.

    c. The sliced vector is multiplied by k-th columns of the channel matrix and subtracted from the received vector to get the desired vector.

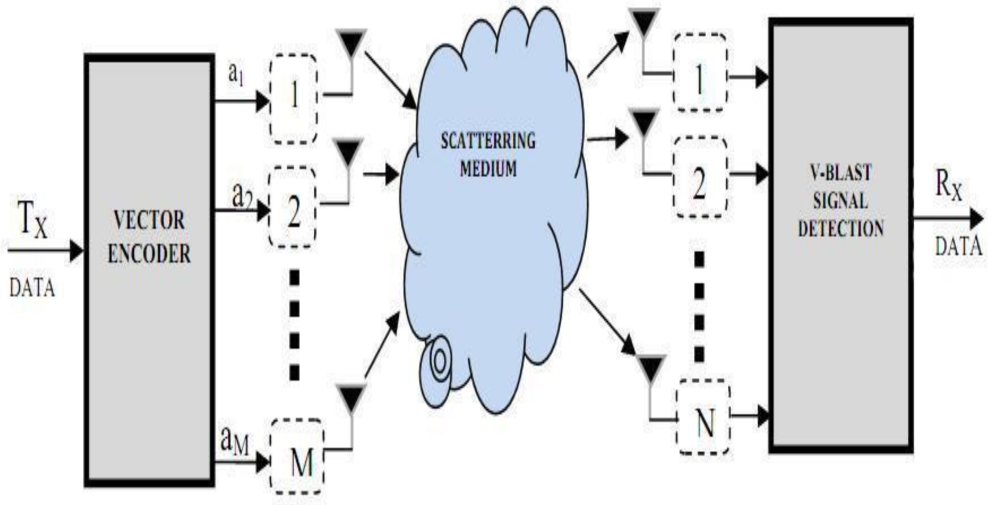


Figure 2-9 VBLAST Detection Diagram

ADVANTAGE OF ZF VBLAST DETECTION

Complexities are Removed

Using ZF VBLAST Detection, the computational and implementation complexities are removed as one can see from the the ZF equation for removing the channel affect

$$\hat{a} = (H^*H)^{-1}Hx = H^{+}x$$

The ZF receiver converts the joint decoding problem into M single stream decoding problems thereby significantly reducing receiver complexity.

Removes Noise and Channel's Effect

ZF VBLAST involves cancellation process which is equivalent to equalization. Hence the interferers (noise and channel) are removed.

DISADVANTAGE OF ZF VBLAST DETECTION

High Noise due to Complexity Reduction

The complexity reduction occurs, although, at the expense of noise enhancement which in results in a significant performance decline (compared to the ML decoder).

BER Increases as compared to ML Decoder

If BER vs Eb/No plots are given we can easily see that ZF VBLAST shows more BER than other techniques reason is because of the noise enhancement.

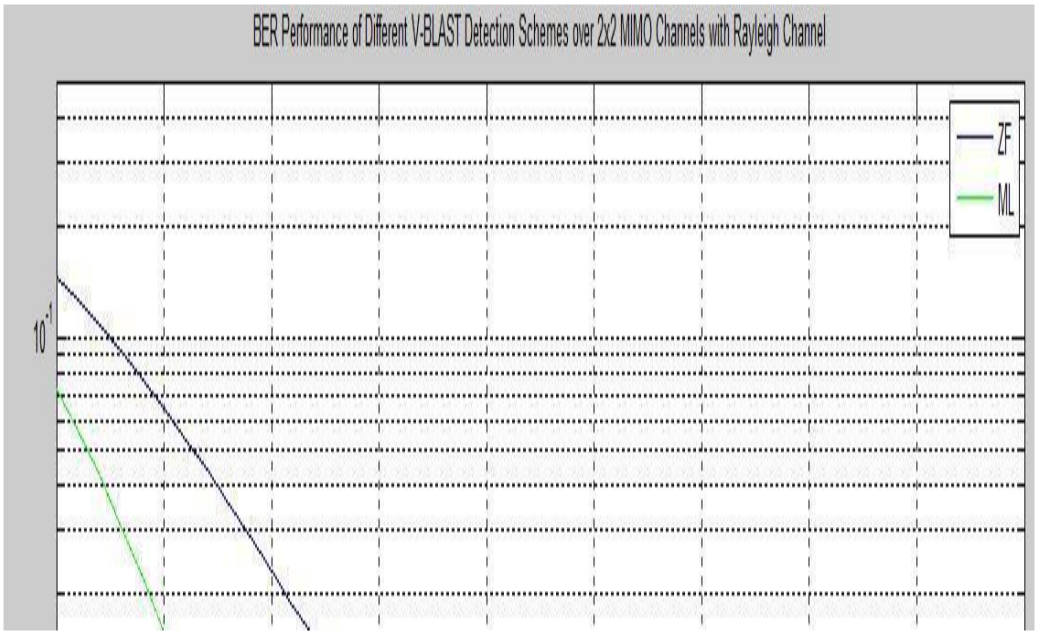


Figure 2-10 ZF VBLAST and ML VBLAST Comparison

COGNITIVE TECHNIQUE

Cognition is an upcoming technology and is now being experimented all over the world. This technique has not yet been applied in the world's communication but is going to be a part of telecommunication soon. In this technique a user buys a frequency from frequency allocation board and that particular frequency band is allotted for that user. This user is termed as the primary or licensed user. Another user who does not buy any band and tries to use the frequency band completely or partially of others is termed as secondary or unlicensed user.

In this technique the secondary user uses the frequency of the primary user in an intelligent way ensuring there is no interruption between the users' transmissions. This process involves measuring the Receiver Operating Characteristics (ROC) of the detector experimentally.

There are free channels allocated to the cognitive transmitter to jump to just in case a primary user is detected. The cognitive transmitter must sense the spectrum regularly and jump to a new channel as soon as the primary user is detected. The cognitive receiver must make sure that it changes the band as the cognitive transmitter changes its band in order for continuous reception of packets. The receiver must constantly resynchronize itself with the cognitive transmitter.

INTRODUCTION

Every user is assigned a particular bandwidth when he buys it from the frequency allocation board. But nowadays the spectrums are being used in irregular intervals, almost 15% to 85% according to the FCC. The remaining frequency is rarely used and hence the cognitive approach takes place.

COGNITIVE RADIO CYCLE

The cognitive radio cycle is shown in the figure:

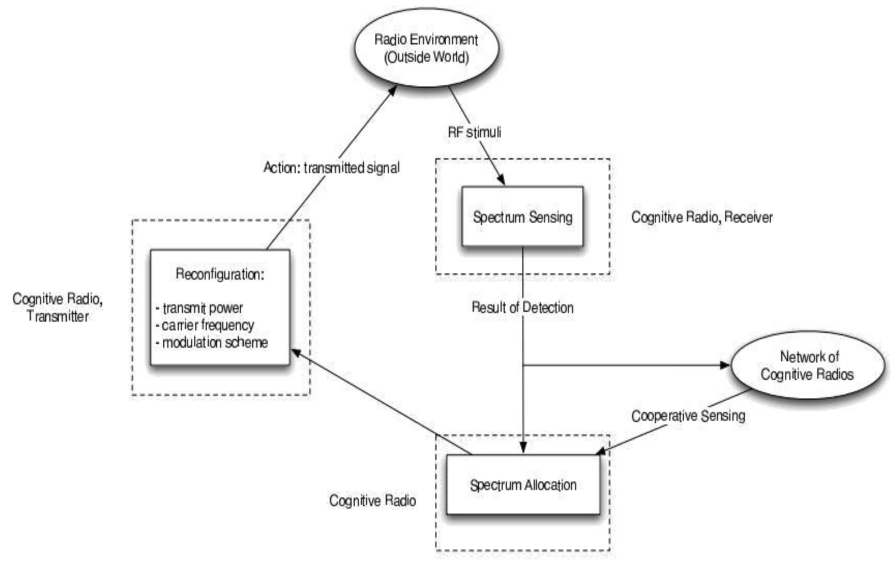


Figure 2-11 Cognitive Radio Cycle

Cognitive technique has four basic steps to complete its cycle:

1. Spectrum Sensing
2. Spectrum Allocation
3. Reconfiguration
4. Transmission

The first step in the cycle involves spectrum sensing. As its name suggests, spectrum sensing is a process in which it is detected whether the primary user is present on the band or not, that is, whether the primary user is transmitting over his band or not. After the spectrum sensing, the results are shared with a cluster of cognitive radios connected together. The cooperative sensing tells the cognitive radios about the holes in the spectrum and also tells which band to use.

The cooperative sensing helps detecting the primary users that are far away from the cognitive receivers and hence tells the cognitive receivers to lock onto a certain band in the absence and presence of the primary user.

The second step involves the spectrum allocation. In this step a band is allocated to the secondary user. That band is allocated which is free of the primary user. In case, where there are more than one secondary users, the band is shared among them.

The third step involves reconfiguration of the cognitive transmitter. This process involves changing in various mechanisms such as change in the carrier frequency, transmission power and modulation scheme.

The fourth and the last step involve the transmission over that band which was found free and is ready to use now.

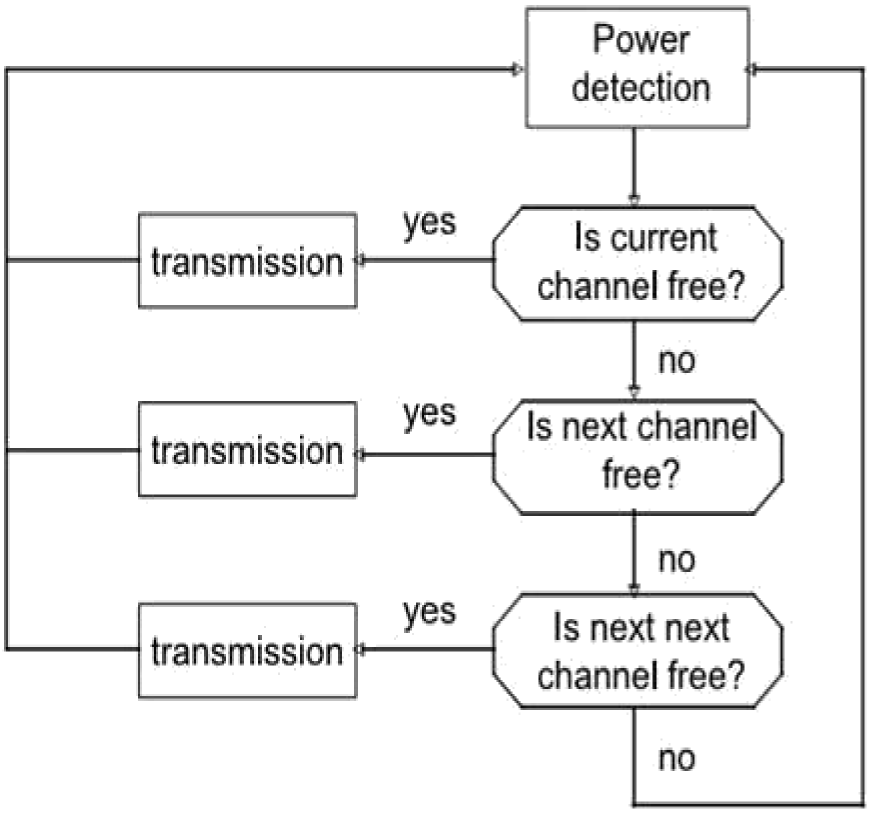


Figure 2-12 Flow Chart for Cognitive Transmitter

The figure above is the flow chart for cognitive and simple to understand if one understands the process described before. Energy is detected on the spectrum (to find the primary user) after which the channel switching takes place.

ADVANTAGES OF COGNITIVE TECHNIQUE

Causes Jamming

Cognitive radio is an intelligent technique to utilize the bandwidth to the fullest if there are holes present. It must not interfere with the primary user in any case. However, if an enemy is trying to send an information against us, that can be jammed by using cognitive technique that the secondary user starts transmitting over his complete bandwidth, interfering with his transmission at all times.

Utilizes Holes In The Spectrum

As already discussed the cognitive transmitter makes sure that the holes in the band are readily used if the primary user is not using those bands. It also moves away from over occupied bands.

Does not let the Primary User know about its presence

Cognitive transmitter does not let the primary user know about the usage of licensed user’s own bandwidth. It is a smart radio considering all the factors regarding interference.

All of the benefits of software defined radio

Cognitive is smart radio technique. Every command is gone through a software. If there needs to be a change we just need to make amendments in the software.

Improves link performance

If the channels are bad and noisy, it adapts away from those channels and increases the data rate where it finds the channels to be in good conditions.

DISADVANTAGE OF COGNITIVE TECHNIQUE

If there are no standards then it is illegal

Cognitive is a way of using someone’s allocated bandwidth without letting the licensed user know. So there should be standards which justify the use of empty bands so that the spectrum does not get the chance of being wasted.

COGNITIVE RADIO IMPLEMENTATION ON USRP

1. A cognitive transmitter first scans the channels and jumps onto the channels depending upon the presence or absence of the primary user.
2. After selecting a channel, the cognitive transmitter locks onto that band.

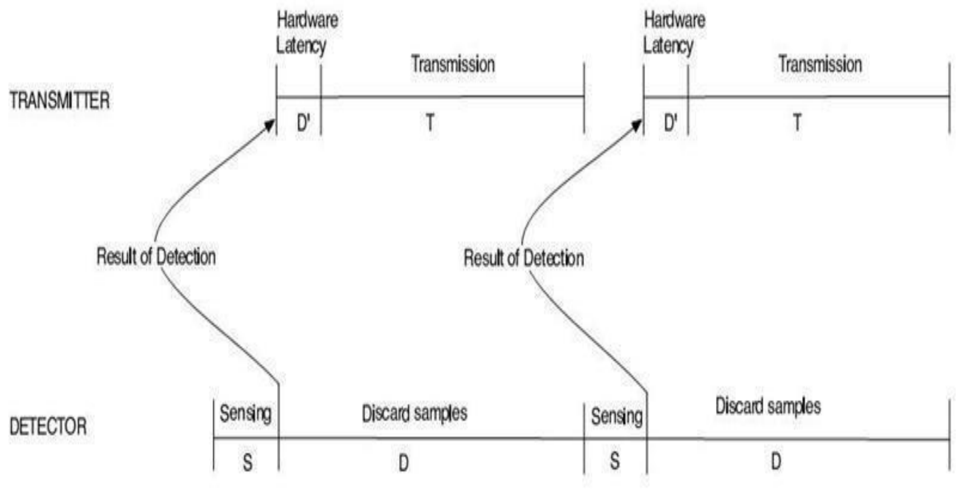


Figure 2-10-3 Parallel Detection and Transmission

This is the simple diagram which reflects the work being done through the USRP for cognition. You can see here that when detection takes place there is no transmission, that is, the transmission stops and only spectrum sensing takes place. After the detection phase the results are taken into account and the cognitive transmitter starts transmitting on the new channel.

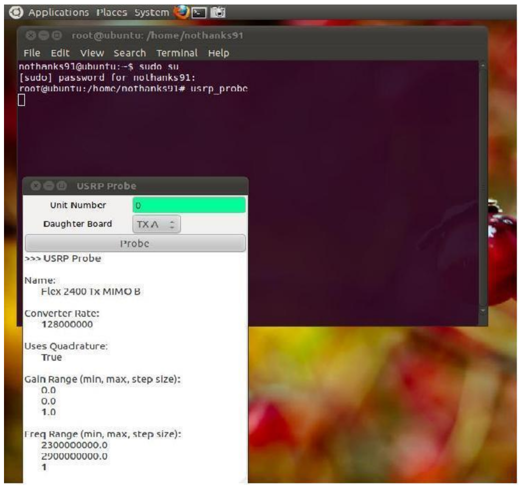
3. A video is taken as input from the opencv software inside the Python code.
4. The video is divided into frames.
5. The frames are divided into bits.
6. Bits are converted to QPSK symbols and are transmitted using two 2.4 GHz daughter cards, as it is 2x2 MIMO.
7. The symbols with the strongest SNR are detected (coming from the same antenna).
8. Symbols are demodulated to get the vectors of bits.
9. Channel is estimated and is used for VBLAST Detection.
10. We get two columns of the channel matrix and get the received vector having the noise portion as well.
11. The vectors are ordered first and then moved to the VBLAST phase and go through the VBLAST Detection.

After the VBLAST Detection the process starts again. If there is a primary user and he starts transmitting on his desired frequency, the cognitive transmitter changes its frequency and the receiver synchronizes itself with the secondary user.

The pictures show that when scanning takes place what is the result shown:

1) First we checked whether the USRP was connected to the TX console and RX console using a command usrp_probe

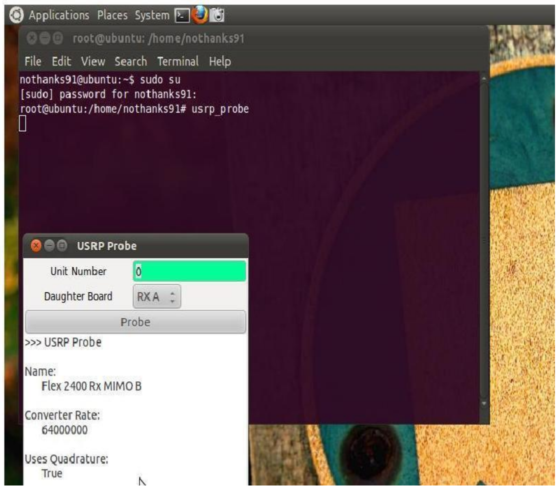
TX CONSOLE **RX CONSOLE**


Applications Places System
root@ubuntu: /home/nothanks91
File Edit View Search Terminal Help
USRP Probe
Probe

```
nothanks91@ubuntu:~$ sudo su
[sudo] password for nothanks91:
root@ubuntu:/home/nothanks91# usrp_probe
```

Unit Number: 0
Daughter Board: TX A

```
>>> USRP Probe

Name:
    Flex 2400 Tx MIMO B

Converter Rate:
    128000000

Uses Quadrature:
    True

Gain Range (min, max, step size):
    0.0
    0.0
    1.0

Freq Range (min, max, step size):
    2300000000.0
    2900000000.0
    1
```


Applications Places System
root@ubuntu: /home/nothanks91
File Edit View Search Terminal Help
USRP Probe
Probe

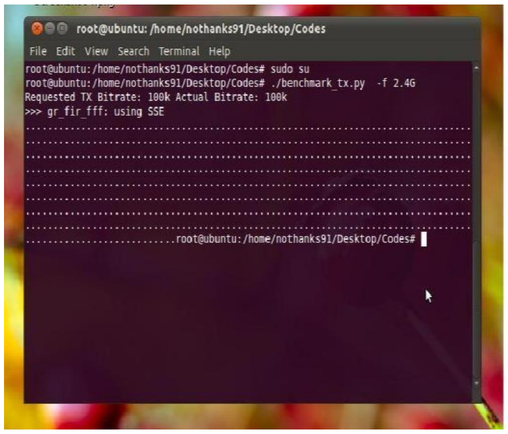
```
nothanks91@ubuntu:~$ sudo su
[sudo] password for nothanks91:
root@ubuntu:/home/nothanks91# usrp_probe
```

Unit Number: 0
Daughter Board: RX A

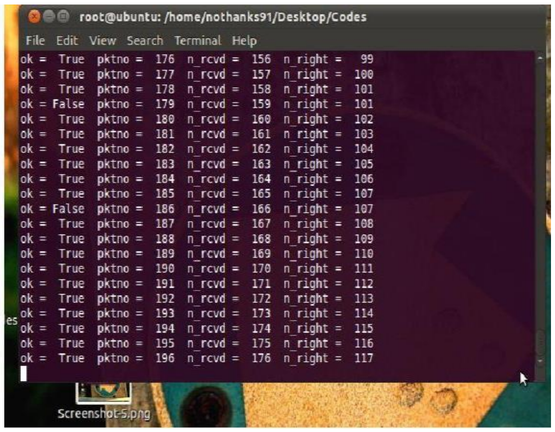
```
>>> USRP Probe

Name:
    Flex 2400 Rx MIMO B

Converter Rate:
    64000000

Uses Quadrature:
    True
```

2) While the transmission was being done at the transmitter at 2.4G the receiver showed the following FFT plot at 2.4GHz.

TX CONSOLE **RX CONSOLE**


root@ubuntu: /home/nothanks91/Desktop/Codes
File Edit View Search Terminal Help

```
root@ubuntu:/home/nothanks91/Desktop/Codes# sudo su
root@ubuntu:/home/nothanks91/Desktop/Codes# ./benchmark_tx.py  -f 2.4G
Requested TX Bitrate: 100k Actual Bitrate: 100k
>>> gr_fir_fff: using SSE
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..............................................................................
..........................root@ubuntu:/home/nothanks91/Desktop/Codes#
```


root@ubuntu: /home/nothanks91/Desktop/Codes
File Edit View Search Terminal Help
Screenshot-5.png

```
ok =  True  pktno =  176  n_rcvd =  156  n_right =   99
ok =  True  pktno =  177  n_rcvd =  157  n_right =  100
ok =  True  pktno =  178  n_rcvd =  158  n_right =  101
ok = False  pktno =  179  n_rcvd =  159  n_right =  101
ok =  True  pktno =  180  n_rcvd =  160  n_right =  102
ok =  True  pktno =  181  n_rcvd =  161  n_right =  103
ok =  True  pktno =  182  n_rcvd =  162  n_right =  104
ok =  True  pktno =  183  n_rcvd =  163  n_right =  105
ok =  True  pktno =  184  n_rcvd =  164  n_right =  106
ok =  True  pktno =  185  n_rcvd =  165  n_right =  107
ok = False  pktno =  186  n_rcvd =  166  n_right =  107
ok =  True  pktno =  187  n_rcvd =  167  n_right =  108
ok =  True  pktno =  188  n_rcvd =  168  n_right =  109
ok =  True  pktno =  189  n_rcvd =  169  n_right =  110
ok =  True  pktno =  190  n_rcvd =  170  n_right =  111
ok =  True  pktno =  191  n_rcvd =  171  n_right =  112
ok =  True  pktno =  192  n_rcvd =  172  n_right =  113
ok =  True  pktno =  193  n_rcvd =  173  n_right =  114
ok =  True  pktno =  194  n_rcvd =  174  n_right =  115
ok =  True  pktno =  195  n_rcvd =  175  n_right =  116
ok =  True  pktno =  196  n_rcvd =  176  n_right =  117
```

3) In order to establish a wireless link for communication, we did a simple

transmission of random data using SISO.

Initially we did spectrum sensing of three channels pre-defined by us. These three channels had the frequency

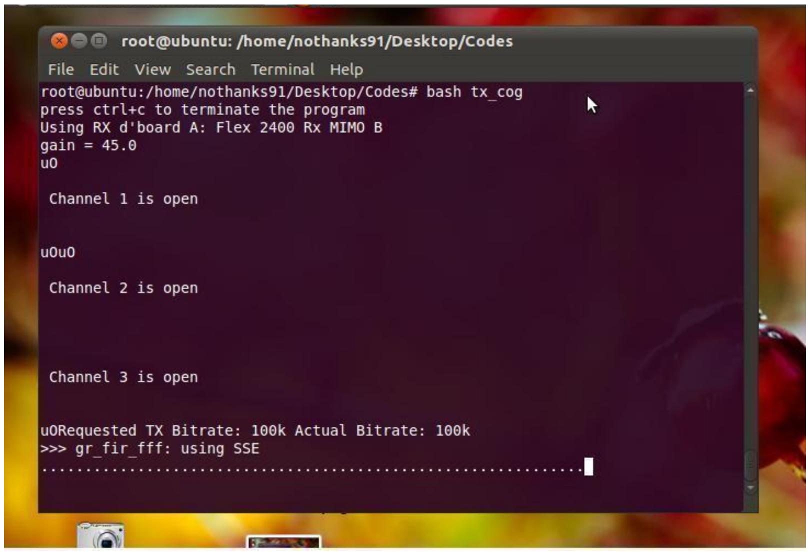
- 2.399GHz -Ch1
- 2.4GHz-Ch2
- 2.401GHz-Ch3

Each channel is of approx. 1MHz bandwidth calculated by its FFT plot.

The reason we are only sensing three channels instead of all frequencies of USRP-1 (RFX2400, 2.3–2.9 GHz for transceiver) is that it will take too much time for sensing and so transmission period will decrease. Since there is a tradeoff between sensing period and transmission period we opt to take more time for transmission.

We found from our transmission that for a 3MHz bandwidth (3 channels: 1 MHz each) sensing time is 6 seconds. If we start spectrum sensing over all the BW available for transmission on a USRP i.e 8MHz it will take approx. a minute to sense the channels before transmitting on them. In the meantime the available white space may get used by the primary user and the opportunity for transmission by the secondary user is lost.

4) When no transmission is being done by TX-1 at 2.4G (Channel 2), all the three channels are sensed open by TX-2 and the transmission starts from the lowest ascending open channel.


root@ubuntu: /home/nothanks91/Desktop/Codes
File Edit View Search Terminal Help

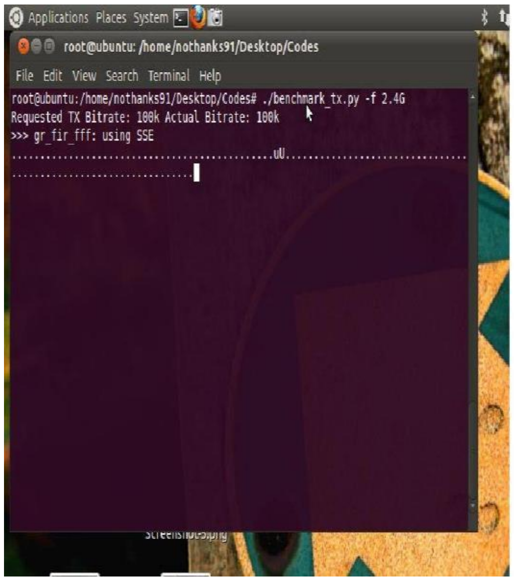
```
root@ubuntu:/home/nothanks91/Desktop/Codes# bash tx_cog
press ctrl+c to terminate the program
Using RX d'board A: Flex 2400 Rx MIMO B
gain = 45.0
uO

 Channel 1 is open


uOuO

 Channel 2 is open



 Channel 3 is open


uORequested TX Bitrate: 100k Actual Bitrate: 100k
>>> gr_fir_fff: using SSE
..............................................................
```

5) There is a transmission going on over one of the three channels, that is, there is a primary user using his frequency to transmit.

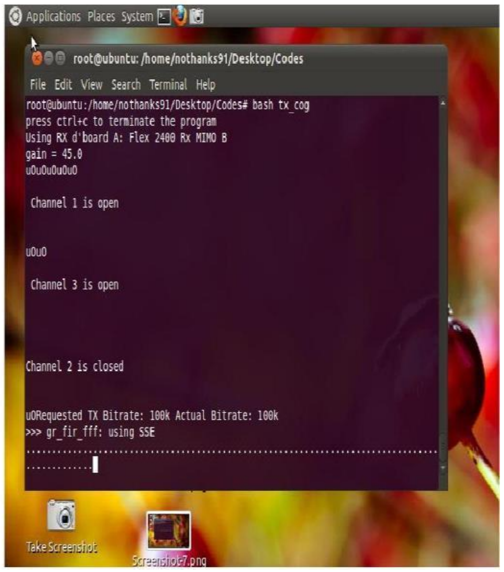
   In this case you can see that when a user starts transmitting over his frequency on channel 2, that is on 2.401 GHz, the cognitive transmitter scans the channels and shows the us that channel 2 is closed and rest of the channels are open and ready to use.

TX-1 Console transmitting at 2.4G

TX-2 Console doing spectrum sensing.
Ch2 is closed because somebody(TX-1) is already transmitting at that frequency


Applications Places System
root@ubuntu: /home/nothanks91/Desktop/Codes
File Edit View Search Terminal Help

```
root@ubuntu:/home/nothanks91/Desktop/Codes# ./benchmark_tx.py -f 2.4G
Requested TX Bitrate: 100k Actual Bitrate: 100k
>>> gr_fir_fff: using SSE
..........................................uU.............................
..............................
```


Applications Places System
root@ubuntu: /home/nothanks91/Desktop/Codes
File Edit View Search Terminal Help
Take Screenshot
Screenshot-7.png

```
root@ubuntu:/home/nothanks91/Desktop/Codes# bash tx_cog
press ctrl+c to terminate the program
Using RX d'board A: Flex 2400 Rx MIMO B
gain = 45.0
uOuOuOuOuO

 Channel 1 is open


uOuO

 Channel 3 is open




Channel 2 is closed


uORequested TX Bitrate: 100k Actual Bitrate: 100k
>>> gr_fir_fff: using SSE
.................................................................................
...........
```

We had chosen three channels of 2.399 GHz, 2.4 GHz and 2.401 GHz frequencies. The cognitive transmitter starts the transmission first by scanning the channels in order to detect the presence of the primary user. In the first case, you can see that all the channels are open, that is, there are no transmissions going on over these channels. The cognitive transmitter chooses any one of the channels and starts transmitting over it.

The cognitive receiver synchronizes itself with the cognitive transmitter and starts receiving the random data.

Once the free channels are found the secondary user (cognitive transmitter) just chooses one of the free channels and starts transmitting over it. The cognitive receiver synchronizes itself with the cognitive transmitter and starts receiving the packets.

CHAPTER 3

RESULTS AND ANALYSIS

3.1 IMPLEMENTATION ON MATLAB

3.1.1 SISO

Using the Monte Carlo simulation we simulated the response curves of SISO under the AWGN channel and the Rayleigh Fading Channel

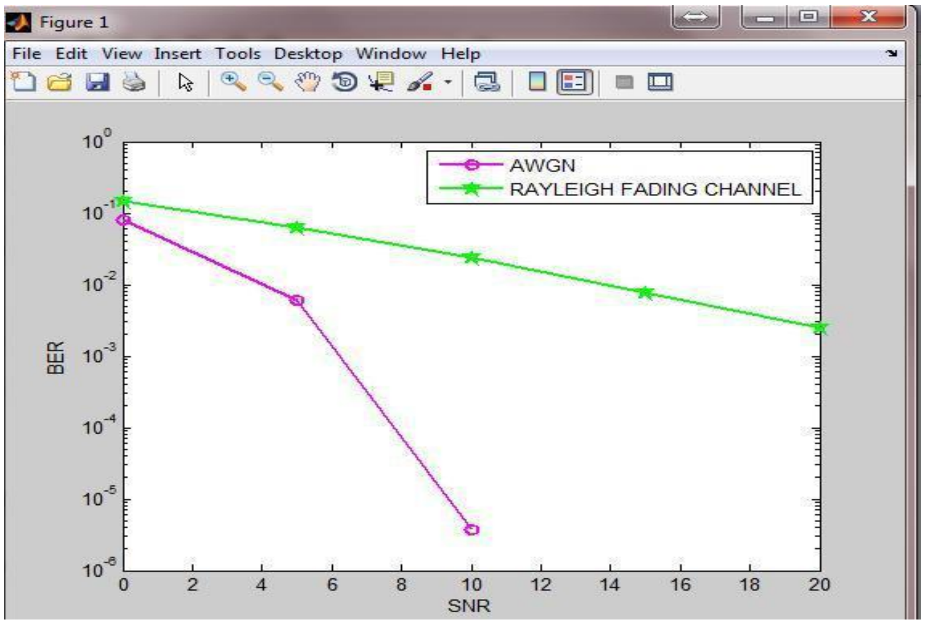


Figure 3-1 SISO AWGN and Rayleigh Comparison

3.1.2 SIMO

MRRC for Rayleigh Fading Channel (Receiver Diversity)

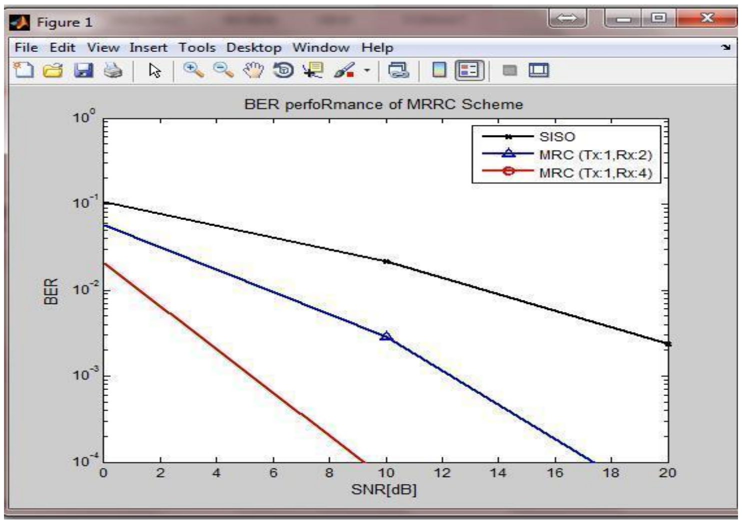


Figure 3-2 SIMO

3.1.3 MISO

Space Time Block Codes/ALAMOUTI SCHEME (Transmit Diversity)

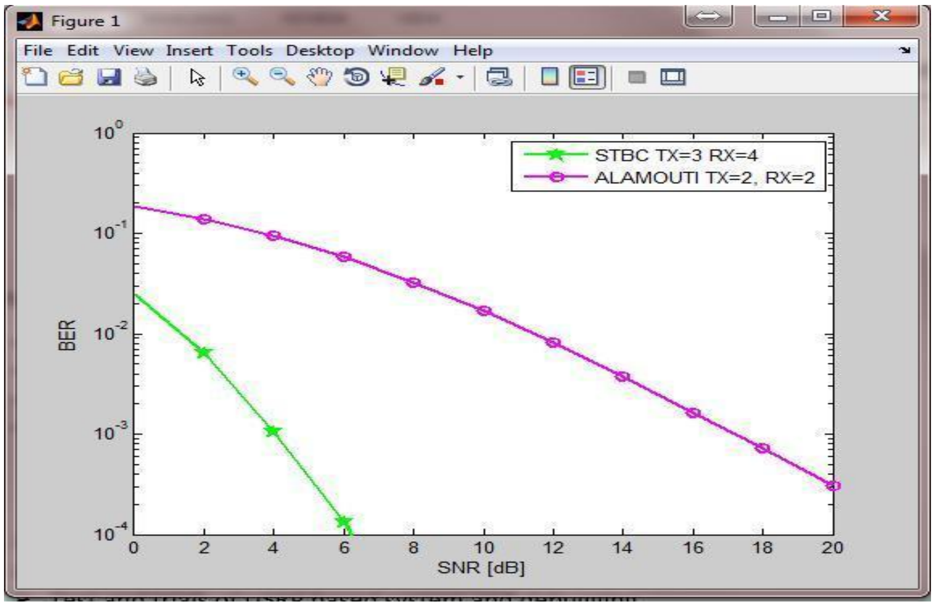


Figure 3-3 MISO

3.1.4 MMSE, ZF Comparison

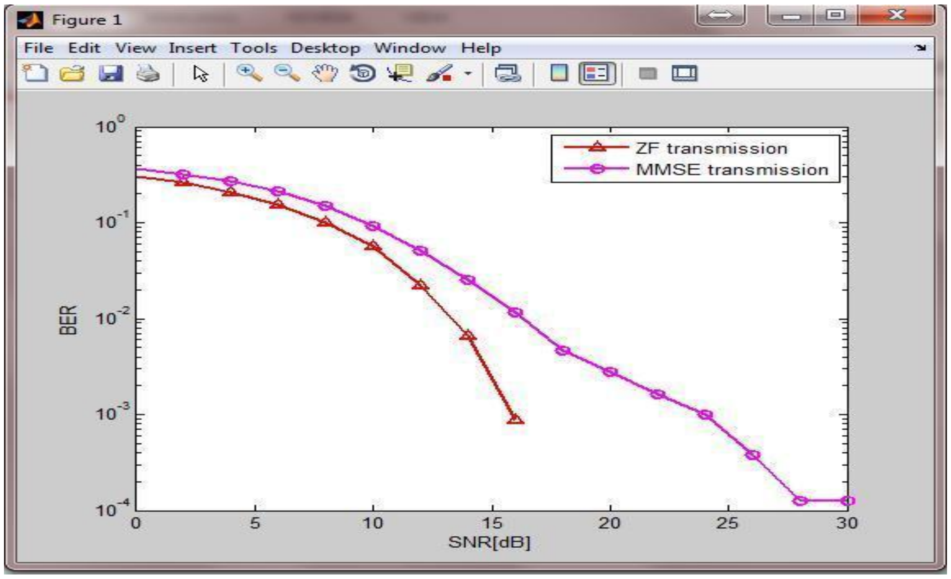


Figure 3-4 ZF and MMSE Comparison

3.1.5 VBLAST DETECTION

Comparison of MMSE VBLAST-OSIC detection with ZF

VBLAST-OSIC detection

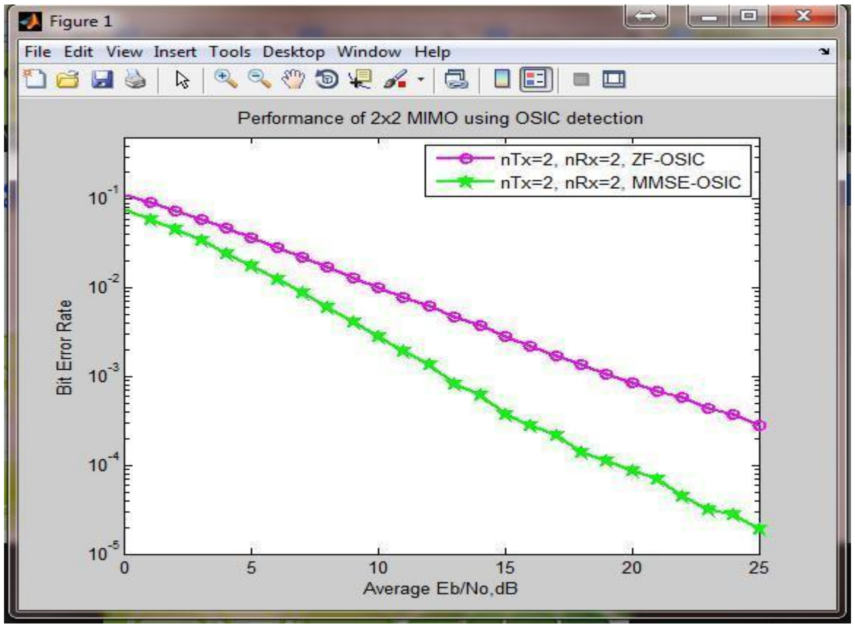


Figure 3-5 VBLAST-OSIC Detection

3.1.6 VBLAST –ZF DETECTION OSIC AND ZF DETECTION

BER Comparison

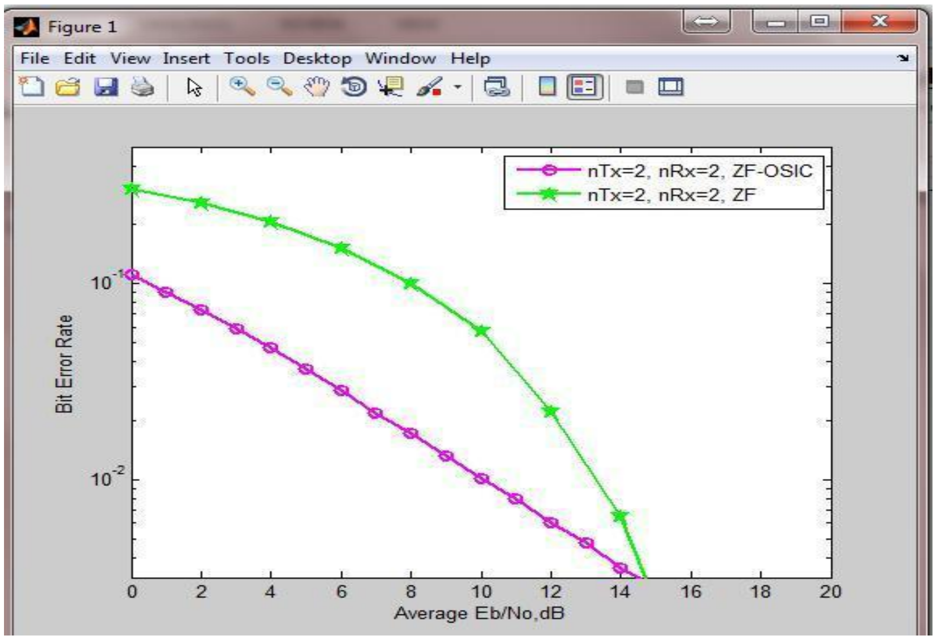
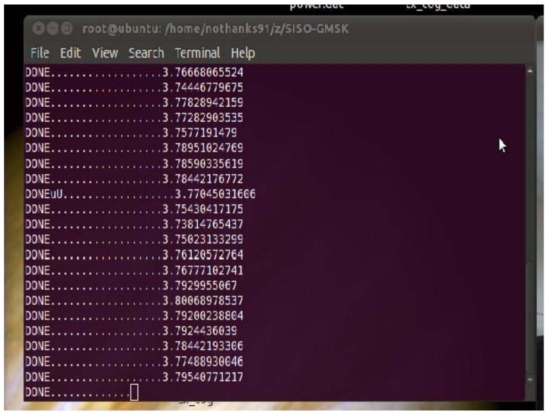


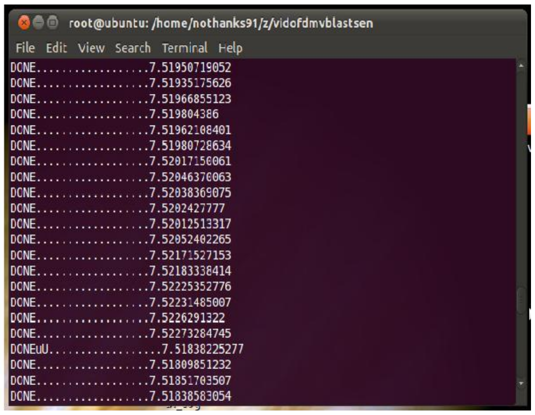
Figure 3-6 VBLAST ZF-OSIC and ZF Detection Comparison

3.1.7 COMPARISON OF FPS OF SISO AND MIMO LINK

The processors are slow for such enhanced computation and processes. The frames per second tend to decrease as the time passes by. A comparison of the frames per second (FPS) of SISO and MIMO link is shown in the figures below


root@ubuntu: /home/nothanks91/z/SISO-GMSK
File Edit View Search Terminal Help

```
DONE..................3.76668065524
DONE..................3.74446779675
DONE..................3.77828942159
DONE..................3.77282903535
DONE..................3.7577191479
DONE..................3.78951024769
DONE..................3.78590335619
DONE..................3.78442176772
DONEuU..................3.77045031606
DONE..................3.75430417175
DONE..................3.73814765437
DONE..................3.75023133299
DONE..................3.76120572764
DONE..................3.76777102741
DONE..................3.7929955067
DONE..................3.80068978537
DONE..................3.79200238804
DONE..................3.7924436039
DONE..................3.78442193306
DONE..................3.77488930046
DONE..................3.79540771217
DONE.............
```


root@ubuntu: /home/nothanks91/z/vidofdmvblastsen
File Edit View Search Terminal Help

```
DONE..................7.51950719052
DONE..................7.51935175626
DONE..................7.51966855123
DONE..................7.519804386
DONE..................7.51962108401
DONE..................7.51980728634
DONE..................7.52017150061
DONE..................7.52046370063
DONE..................7.52038369075
DONE..................7.5202427777
DONE..................7.52012513317
DONE..................7.52052402265
DONE..................7.52171527153
DONE..................7.52183338414
DONE..................7.52225352776
DONE..................7.52231485007
DONE..................7.5226291322
DONE..................7.52273284745
DONEuU..................7.51838225277
DONE..................7.51809851232
DONE..................7.51851703507
DONE..................7.51838583054
```

SISO FPS **MIMO FPS**

The comparison shows that the FPS of MIMO is approximately twice of the FPS of SISO showing a better FPS.

3.2 Receiver Operating Characteristics (ROC)

The Receiver Operating Characteristics (ROC) is the major parameter implied while using the cognitive technique. The ROCs show us the condition of the detector in the absence and presence of the primary user.

False alarms are those faults in the cognitive transmitter (detector) when:

1. There is no transmission but the cognitive transmitter shows a detection
2. There is transmission going on but the cognitive transmitter does not change the frequency band.

Looking at these conditions, a threshold value is set so that the detector detects the transmission efficiently and correctly.

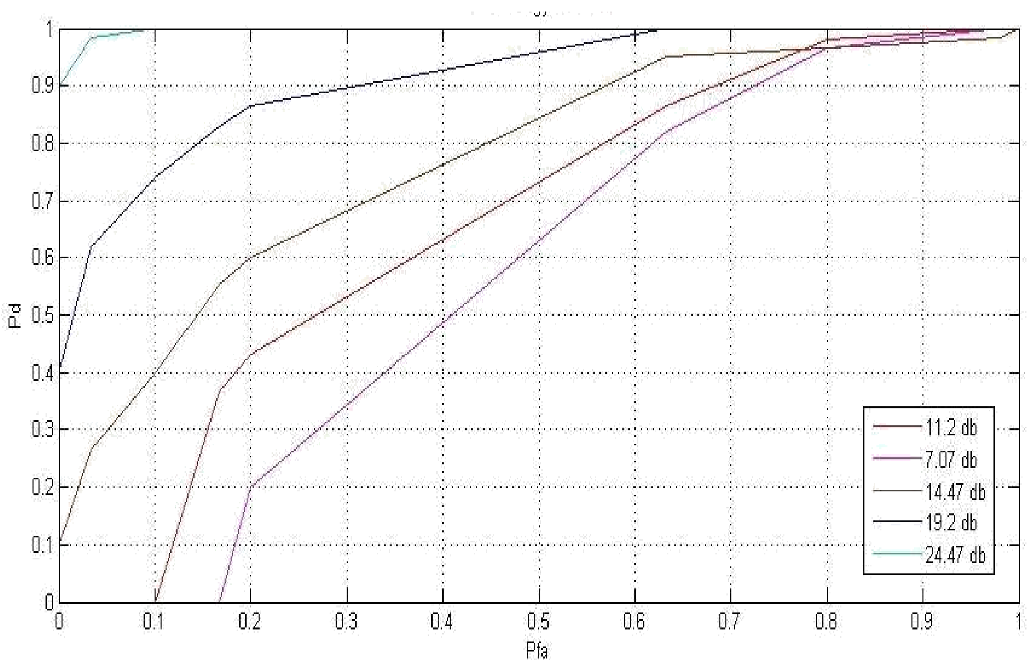


Figure 3-7 Receiver Operating Characteristic (ROC)

The graphs show that when the SNR is increased above 24 dB, the cognitive transmitter's correct probability detection also increases as well of as the probability of false alarms (faulty detections) decreases.

CHAPTER 4

SUMMARY, FUTURE WORK AND CONCLUSION

4.1 SUMMARY

This project was a prototype involving the basics of different upcoming techniques of radio communication. This platform provides a proof of concepts of theory. This project involves the making of a 2x2 MIMO link having two transmit antennas and two receive antennas. The data rate of such a system is increased almost two times than that of a SISO system. MIMO system is also used to remove the multipath fading effects.

OFDM (Orthogonal Frequency Division Multiplexing) is used as a modulation technique. MIMO together with OFDM are used in 4G technologies for enhanced data rates as well as ensured bandwidth efficiency. The closely spaced orthogonal sub carriers in this technique allows the bandwidth to be used efficiently and reduces the Inter Symbol Interference (ISI).

ZF VBLAST Detection is used. Its algorithm involves successive cancellation process which removes the effects of channel and other interferers.

Cognitive technique is used for spectrum sensing in which the secondary user uses partial or complete frequency band of the primary user. The secondary makes sure that there isn’t any sort of interference allows the use of bandwidths effectively and reduces its wastage.

4.2 FUTURE WORK

1. The processing speed has become the major problem in our project. Due to the less computational efficiency and more heat the computers suffer from less efficiency as time passes by. The communication can be made better using better processors.

2. The project can be integrated on high speed processing boards as no one can carry laptops for fast communication. The project can be made portable in a user friendly way.

3. Instead of BPSK as mapping scheme, QPSK can be used which sends more bits over the same bandwidth hence increasing the data rate. In our project, QPSK was used but its data rate was so high that the receiving side was not able to reconstruct the image.

4. Using Channel feedback techniques, the channel estimates can be known at transmitter side and thus the knowledge of the channel conditions at the transmitter side is known. The channel with more favorable conditions for transmission can be selected.

4.3 CONCLUSION

The proposed platform allows a better technique of transmission and reception of data over the link. 2x2 MIMO not only reduces the multipath effects but also increases the data rate.

Using ZF VBLAST Detection the effect of channel is removed accurately along with other interferers allowing better detection than ZF detection. The cognitive technique implemented, ensures the bandwidth not to be wasted. This technique also disallows interference between the secondary and primary transmissions.

This project is a proof of concepts of different mechanisms when carried out in real time communication. The results from the project highlight that using such or better techniques, communication can be enhanced. Those points are helpful for better transmissions in near future. This prototype, containing the software defined radio involving different techniques, is capable of a good real time terrestrial, mobile and satellite communication which can be made better.

BIBLIOGRAPHY

1) Digital communications Fundamentals2nd an edition”.
2) Modern Digital and Analogue communication
3) Discrete Time Signal Processing By “Alan R.Buck”
4) MIMO-OFDM Wireless Communications with MATLAB by Yong Soo Cho, Jaekwon Kim, Won Young Yang, Chung Gu Kang
5) Advanced Sensing Techniques of Energy Detection in Cognitive Radios by Hano Wang, Gosan Noh, Dongkyu Kim, Sungtae Kim, and Daesik Hong
6) Space Time code for MIMO Systems by Fernando Gregorio
7) “V-BLAST: An Architecture for Realizing Very High Data Rates Over the Rich-Scattering Wireless Channel P. W. Wolniansky, G. J. Foschini, G. D. Golden, R. A.
8) Valenzuela Bell Laboratories, Lucent Tec
9) 8) V-BLAST: A SPACE-DIVISION MULTIPLEXING TECHNIQUE PROVIDING A SPECTRAL EFFICIENCY NECESSARY FOR HIGH DATA RATE WIRELESS NETWORKS by Nasir D.
10) Gohar, Zimran Rafique, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology
11) http://gnuradio.org/redmine/wiki/gnuradio
12) http://gnuradio.org/redmine/wiki/gnuradio/BuildGuide
13) http://gnuradio.org/redmine/wiki/gnuradio/UbuntuInstall

APPENDIX –A

“PYTHON CODE SEGMENTS”

```
from gnuradio import gr, gru, eng_notation, optfir, window
from gnuradio import audio
from gnuradio import usrp
import struct
import numpy as np
import Gnuplot,Gnuplot.PlotItems, Gnuplot.funcutils
from easygui import msgbox
class tune(gr.feval_dd):
""""
This class allows C++ code (bin_statistics_f.cc) to callback into python to change USRP RF
center Frequency.
""""
def eval(self, ignore):
""""
This method is called from gr.bin_statistics_f when it wants to change
the center frequency. This method tunes the front end to the new center
frequency, and returns the new frequency as its result.
""""
try:
new_freq = self.fb.set_next_freq()
return new_freq
except Exception, e:
print "tune: Exception: ", e
""""
Class to parse the incomming messages sent by bin_statistics.
2. It extracts the center frequncy and the incomming data vector length
3. Convert received data string to a floating point format.
4. Store the output in the "data" array 79

def set_next_freq(self): target_freq =
self.next_freq
self.next_freq = self.next_freq + self.freq_step if
self.next_freq >= self.max_center_freq: self.next_freq =
self.min_center_freq
if not self.set_freq(target_freq):
print "Failed to set frequency to", target_freq return
target_freq
def set_freq(self, target_freq):
```

```
"""

Set the center frequency we're interested in. @param
target_freq: frequency in Hz @rypte: bool
Tuning is a two step process. First we ask the front-end to tune as close
to the desired frequency as it can. Then we use the result of that
operation and our target_frequency to determine the value for the
digital down converter.
"""
return self.u.tune(0, self.subdev, target_freq) def
set_gain(self, gain): self.subdev.set_gain(gain)
global loop,ch1,ch2,ch3 loop = 0
ch1 = 0 ch2 = 0 80

ch3 = 0
def main_loop(fb): loop = 1
ch1 = 0 ch2 = 0
ch3 = 0 while 1:
g = Gnuplot.Gnuplot(debug=0)
5.Get the next message sent from the C++ code (bin_statistics)
6.It contains the center frequency and the mag squared of the fft m =
parse_msg(fb.msgq.delete_head()) # This is a blocking call.
7.Print center freq so we know that something is happening...
#print "##################################################################"
#fft_correction = np.dot((0.00976562,m.data)
#psd = np.sum(fft_correction,0) #print
m.data
#print "center frequency %4d PSD = " % (m.center_freq)
#print "##################################################################" if
m.center_freq == min_center_freq: # If we get the minimum frequency, it'll
m.data = m.data[1:26]
temp = sum(m.data)/len(m.data) if loop
in range(9,13):
if temp > 2e7:
ch1=ch1+1 #print
ch1
elif loop == 12 and ch1 < 3: status1
= "Open "
print '\n' 81

print " Channel 1 is open " print
'\n'
if loop in range(19,23): if
temp > 2e7: ch2=ch2+1
#print ch2
elif loop == 22 and ch2 < 3:
status2 = "Open "
print '\n'
print " Channel 2 is open " print
'\n'
if loop in range(29,33): if
temp > 2e7: ch3=ch3+1
#temp = temp/(200000000) media
= str(temp)
todo= p + " " + media + '\n'
modo= " Channel " + str(loop) + " " + media + '\n'
channel.write(modo)
power.write(todo)
#print "################################################################" if
m.center_freq == (max_center_freq-freq_step): # If it gets the final frecuency
p=str(m.center_freq) # It'll write the last frecuency with its Power in the power.dat file
# Load the plot with the data obtained from URSP
#g.plot(Gnuplot.File(filename1))
g.plot(Gnuplot.File(filename1,with_= 'linespoints'))
#g.plot(Gnuplot.File(filename1, with_='lines')) 82
g.title('The Sensed Spectrum Plot')
g.xlabel('Frequency')
g.ylabel('magnitude(normalized)')
g.replot() g.hardcopy('spectrum.jpg')
if ch1 > 3 : status1 =
"Closed" print '\n'
print "Channel 1 is closed "
print '\n'
if ch2 > 3 : status2 =
"Closed" print '\n'
print "Channel 2 is closed "
print '\n'
if ch3 > 3 :
raise SystemExit
1.FIXME do something useful with the data...
2.m.data are the mag_squared of the fft output (they are in the
3.standard order. I.e., bin 0 == DC.)
4.m.raw_data is a string that contains the binary floats.
5.You could write this as binary to a file.
loop = loop+1
if __name__ == '__main__':
fb = my_top_block_1()
try:
fb.start() # start executing flow graph in another thread, and return the control to the python
program
main_loop(fb)
except KeyboardInterrupt: 83
pass #!/bin/sh #while
true
echo "press ctrl+c to terminate the program" while true
do
sleep 0.2
```

python /home/user/Desktop/TP/vidofdmvblastsen/spectrum_sensing.py convert spectrum.ps foo.jpg
sleep 0.2
python /home/user/Desktop/TP/vidofdmvblastsen/ofdm_rx.py control_c()
# run if user hits control-c
{
echo -en "\n*** Terminating Program ***\n"
exit $?
}
trap control_c SIGINT echo
$control_c
done # copy the final answers back into options for use by modulator
#options.bitrate = self._bitrate
#options.bitrate = 1500e3 self.txpath =
transmit_path(options) self.connect(self.txpath,
self.u)
def _setup_usrp_sink(self):
""" 84

Creates a USRP sink, determines the settings for best bitrate, and attaches to the transmitter's subdevice.
"""
self.u = usrp.sink_c(fusb_block_size=self._fusb_block_size,
fusb_nblocks=self._fusb_nblocks) self.u.set_interp_rate(self._interp)
6.Set the USRP for maximum transmit gain
7.(Note that on the RFX cards this is a nop.)
self.set_gain(self.subdev.gain_range()[1])
8.enable Auto Transmit/Receive switching
self.set_auto_tr(True)
def _print_verbage(self):
"""
Prints information about the transmit path
"""
print "Using TX d'board %s" % (self.subdev.side_and_name(),) print
"modulation: %s" % (self._modulator_class.__name__) print "interp: %3d"
% (self._interp)
print "Tx Frequency: %s" % (eng_notation.num_to_str(self._tx_freq)) def
add_freq_option(parser):
"""
Hackery that has the -f / --freq option set both tx_freq and rx_freq
"""
def freq_callback(option, opt_str, value, parser):
parser.values.rx_freq = value parser.values.tx_freq = value
85

if not parser.has_option('--freq'): parser.add_option('-f', '--
freq', type="eng_float", action="callback",
callback=freq_callback,
help="set Tx and/or Rx frequency to FREQ [default=%default]",

```
metavar="FREQ")
# ////////////////////////////////////////////////////////////////////////////
# main
# ////////////////////////////////////////////////////////////////////////////
global loop,lnp lnp = 0
loop = 0 def main():
M", "--megabytes", type="eng_float", default=1.0, help="set
megabytes to transmit [default=%default]")
parser.add_option("","--discontinuous",          action="store_true",          default=False,
help="enable discontinuous mode")
#parser.add_option("","--from-file", default="indirect1.jpg",
parser.add_option("","--from-file", default="output.jpg", help="use file
for packet contents")
loop = 0 lnp = 0
my_top_block.add_options(parser, expert_grp)
transmit_path.add_options(parser, expert_grp)
blks2.ofdm_mod.add_options(parser, expert_grp)
blks2.ofdm_demod.add_options(parser, expert_grp)
fusb_options.add_options(expert_grp)
(options, args) = parser.parse_args ()
options.from_file = "output.jpg"
options.verbose = False options.fusb_nblocks =
0 options.modulation = "bpsk" 86
options.rx_freq = 2.4e9

options.rx_freq = 2.4e9 options.interp = 256
options.fusb_block_size = 0 decision =
open("result.dat",'r') decision =
decision.read() options.tx_freq = None
if decision[0:6] == "Open ":
options.tx_freq = 2.399e9
elif decision[6:12] == "Open ":
options.tx_freq = 2.4e9
elif decision[12:18] == "Open ":
options.tx_freq = 2.401e9
print options.tx_freq options.discontinuous
= False options.log = False
options.megabytes = 1.0
options.occupied_tones = 52
options.cp_length = 2 options.freq = None
options.tx_subdev_spec = None
options.fft_length = 64
options.tx_amplitude = 200 options.size =
4000.0
# build the graph
tb = my_top_block(options)
r = gr.enable_realtime_scheduling() if r !=
gr.RT_OK:
```

```
print "Warning: failed to enable realtime scheduling" 87

tb.start() # start flow graph
# generate   and   send   packets   c   =
cvWaitKey(5)
im = Image.open("indirect1.jpg")
im.save("output.jpg",quality=30) source_file =
open(options.from_file, 'r') #dest=
open('yoyo.jpg','w')
if c=='\x1b': break;
tfps=time.time() while n <
nbytes:
data = source_file.read(pkt_size - 2) if data ==
'':
break; if loop:
data   =   struct.pack('!H',   pktno   &   0xffff)   +   data
#dest.write(data)
c = cvWaitKey(5) if
c=='\x1b': break;
payload   =   struct.pack('!H',   pktno   &   0xffff)   +   data
send_pkt(payload)
n          +=          len(payload)
sys.stderr.write('.')
if                options.discontinuous:
time.sleep(1)
pktno += 1 #frametime=time.time()-tfps
#if frametime<0.067:
time.sleep(0.067-frametime) ttfps=ttfps+time.time()-tfps 88
fno+=1
fps=fno/ttfps print fps
sys.stderr.write('DONE')
source_file.close() #time.sleep(10)
lnp = lnp+1 if lnp ==
100:
raise SystemExit
send_pkt(eof=True)
tb.wait() # wait for it to finish if
__name__ == '__main__': try:
main()
except KeyboardInterrupt: pass
#!/usr/bin/env python from opencv.cv
import * from opencv.highgui import *
from opencv import cv import vblast
from opencv import highgui from
gnuradio import gr, blks2 from gnuradio
import usrp
class dual_usrp_source(gr.top_block): def
```

```
__init__(self): gr.top_block.__init__(self) 89

freq0 = 2.4e9 freq1 =
2.4e9
# ----------------------------------------------------------------
# Set up USRP to transmit on both daughterboards
self.u = usrp.sink_c(nchan=2) # say we want two channels
self.dac_rate = self.u.dac_rate() # 128 MS/s self.usrp_interp = 400
self.u.set_interp_rate(self.usrp_interp)
self.usrp_rate = self.dac_rate / self.usrp_interp # 320 kS/s
# we're using both daughterboard slots, thus subdev is a 2-tuple
self.subdev = (self.u.db(0, 0), self.u.db(1, 0))
print "Using TX d'board %s" % (self.subdev[0].side_and_name(),) print
"Using TX d'board %s" % (self.subdev[1].side_and_name(),)
# set up the Tx mux so that
# channel 0 goes to Slot A I&Q and channel 1 to Slot B I&Q self.u.set_mux(0xba98)
self.subdev[0].set_gain(self.subdev[0].gain_range()[1]) # set max Tx gain
self.subdev[1].set_gain(self.subdev[1].gain_range()[1]) # set max Tx gain
self.set_freq(0, freq0)
self.set_freq(1, freq1) self.subdev[0].set_enable(True) # enable
transmitter self.subdev[1].set_enable(True) # enable transmitter
# apply some gain
if_gain = 10000
ifamp = gr.multiply_const_cc(if_gain) 90
def set_freq(self, side, target_freq):
"""
Set the center frequency we're interested in.
"""
print "Tuning side %s to %sHz" % (("A", "B")[side], num_to_str(target_freq)) r =
self.u.tune(self.subdev[side].which(), self.subdev[side], target_freq)
if r:
print " r.baseband_freq =", num_to_str(r.baseband_freq) print "
r.dxc_freq =", num_to_str(r.dxc_freq)
print " r.residual_freq =", num_to_str(r.residual_freq)
print " r.inverted =", r.inverted
if r.inverted == True:
print "......................................................................"
print ""
print "Error : Please make sure that both daughter cards are proerly inserted "
print "......................................................................"
raise SystemExit
print " OK"
return True
else:
print " Failed!"
return False
# Make a static method to call before instantiation
add_options = staticmethod(add_options)
```

```
def add_freq_option(parser):
"""
Hackery that has the -f / --freq option set both tx_freq and rx_freq 91

"""
def freq_callback(option, opt_str, value, parser):
parser.values.rx_freq = value parser.values.tx_freq =
value
if not parser.has_option('--freq'): parser.add_option('-f', '--
freq', type="eng_float", action="callback",
callback=freq_callback,
help="set Tx and/or Rx frequency to FREQ [default=%default]",
metavar="FREQ")
# /////////////////////////////////////////////////////////////////////////////
# main
# /////////////////////////////////////////////////////////////////////////////
global n_rcvd, n_right, loop,lnp loop =
0
lnp = 0
print "\n:::::::::::: LIVE MIMO-OFDM Cognitive Reciever :::::::::::\n"
loop = 0 n_rcvd = 0
n_right = 0 rcv_flag =
True

def carrier_sensed():
""" Return True if the receive path thinks there's carrier """
return   tb.rxpath.carrier_sensed()   #print
carrier_sensed()
def rx_callback(ok, payload): global
n_rcvd, n_right, rcv_flag
(pktno,)   =   struct.unpack('!H',   payload[0:2])   if
pktno == -1: 92

ok == 0
if pktno == 0:
frame = cvLoadImage( 'received.jpg' )
cvShowImage( 'Video Rx', frame )
cvWaitKey(25)
f = open('received.jpg','w') else:
f = open('received.jpg',"a") f.flush()
f.write(payload[2:]) f.close()
n_rcvd += 1 rcv_flag =
False
print "ok = %5s pktno = %4d Received Pkts = %4d Correct Pkts = %4d" % (
(options, args) = parser.parse_args ()
options.verbose = False
options.fusb_nblocks = 0
options.modulation = "bpsk"
options.decim = 128
options.fusb_block_size = 0
options.tx_freq = 2.399e9
options.fft_length = 64
options.discontinuous = False print
decision[0:12]
if decision[0:12] == "0000000000.0":
options.rx_freq = 2399000000.0
elif decision[0:12] == "2399000000.0":
options.rx_freq = 2399000000.0+1000000 print
options.rx_freq 93

#time.sleep(2) options.show_rx_gain_range
= False options.rx_subdev_spec = None
options.rx_gain = None
# build the graph
tb = my_top_block(rx_callback, options) print
carrier_sensed()
r = gr.enable_realtime_scheduling() if r !=
gr.RT_OK:
print "Warning: failed to enable realtime scheduling"
tb.start() # start flow graph
loop = 0 lnp = 0
while 1:
def rx_callback(ok, payload): global
n_rcvd, n_right, rcv_flag lnp = 0
print lnp
(pktno,) = struct.unpack('!H', payload[0:2]) if
pktno == -1:
ok == 0
if pktno == 0:
frame = cvLoadImage( 'received.jpg' )
cvShowImage( 'Video Rx', frame )
cvWaitKey(25)
f = open('received.jpg','w')
print "ok = %5s pktno = %4d Received Pkts = %4d Correct Pkts = %4d" % ( #print
"Carrier not found on current trasmission frequrncy"
loop = loop+1 94

#print loop
if loop == 500000 or lnp==500000:
#print "Carrier not present on current transmission frequency...Rx Changing frequency"
crf = open("crf.dat", "w")
crf = open("crf.dat", "a")
crf.write(str(options.rx_freq))
raise SystemExit
#alpha = 0.001
#thresh = 30 # in dB, will have to adjust
#probe = gr.probe_avg_mag_sqrd_c(thresh,alpha)
#print probe
#print carrier_sensed()
#while carrier_sensed() == True:
# print " Carrier Present on current transmission frequency "
#while carrier_sensed() == False:
```

```
# print " No carrier on current transmission frequency "
# loop = loop+1
# if loop == 10:
# raise SystemExit
tb.wait() # wait for it to finish if
__name__ == '__main__': try:
main()
except KeyboardInterrupt:
pass 95
```