Master Thesis on

Novel Blind Spectrum Sensing Schemes by Utilizing Multiple Antennas in Cognitive Radio

Faculty of Electro-Communications

Department of Electronic Engineering

0832089

AbdulRahman AL-ABBASI

Supervisor: Associate Professor FUJII Takeo

The University of Electro-Communications

Advanced Wireless Communication research Center (AWCC)

FUJII Laboratory

1-5-1, Choufugaoka, Choufu-shi, Tokyo 182-858 Japan

Dedication

This thesis is dedicated to my parents

Yousef and Ayda

who encouraged and supported me during the period of my study.

和文概要

現在の固定的な周波数割り当てに起因する周波数資源不足問題の解決法として、コグニティブ無線の活用が提案されている。コグニティブ無線性能改善のため、ダイナミックスペクトルアクセスおよびソフトウェア無線が大きな役割を果たすと期待されている。ダイナミックスペクトルアクセスを実現するための鍵となる技術の一つとして、スペクトルセンシングがある。スペクトルセンシングは、目的の周波数帯域において、プライマリユーザの存在をスキャンして検出することで実現することができる。

本論文は二つのパートに分けることができる。一つ目のパートは、スペクトルセンシングの理論的検討であり、二つ目のパートは、SDR技術によるスペクトルセンシングの実装に関するものである。理論的研究のパートでは、検出性能に対するチャネルの影響を軽減することを目標とする。無線通信チャネルではプライマリ信号が変動し、その結果としてコグニティブユーザがプライマリユーザを検出することを難しくしている。本研究ではこの問題を解決するために、コグニティブユーザの検出器に、複数アンテナを用いたダイバーシティ手法を用いることを注目している。コグニティブ無線環境では既存システムが分からないため、チャネル推定ができないことから、コグニティブデバイスは低いSNRに耐性が必要となる可能性があり、このことがプライマリユーザ検出をさらに難しくしている。従来のダイバーシティ手法、および上記に記載されている条件も考慮し、本研究ではプライマリユーザをブラインド検出するための二つのダイバーシティ合成手法を提案している。最初の合成手法は、EGCの考え方に基づいた同一利得のウェイトを用いた合成手法である。ここでは、ウェイトを決定する際にチャネル情報を使用できないことを注意する必要がある。本手法では、誤警報確率を設定するために、理論的に閾値を求める手法についても提案している。もう一つの手法はMRCの考え方に基づいた合成手法である。この手法では、受信信号共分散に対する固有ベクトルに対応した振幅及び位相が変化するウェイトを用いる。双方の手法共に、理論的および数値的に解析を行う。シミュレーション結果により、提案手法ではチャネル情報を使用しなくてもプライマリシステムの検出性能が改善できることが確認できる。

本研究で検討しているコグニティブ無線は、GNU Radioと呼ばれるソフトウェア無線技術の活用が実用化の重要な技術と考えられている。そこで、本論文ではこの実装技術についても検討を行っている。最初に、GNU Radioデバイスを用いた通信システムを構築した。これを活用し、本論文の前半部分の実装として、スペクトル検出センサーを作成した。このセンサーを使い、GNU Radioを用いた二つの平均化されたスペクトルセンシングの感度性能について導出した。一つはサンプル数を変化させた結果であり、もう一方は受信信号のSNRを変化させた場合の性能である。

Abstract

Due to the scarcity of the frequency spectrum and to overcome the static spectrum access problems cognitive radio has been proposed. Dynamic spectrum access and software defined radios are expected to play a main role in improving the performance of a cognitive radio system. One of the primary keys for realizing the dynamic spectrum access is spectrum sensing. Spectrum sensing can be achieved by scanning and detecting the primary users' existence in the targeted frequency bands. We divide this thesis into two parts; one is the theoretical research of spectrum sensing, while the other is an implementation part of spectrum sensing using a proposed SDR technology.

As for the theoretical part we aimed for mitigating the channel effect on the detection performance. Wireless channel fluctuates the primary signal which makes it even more difficult for the cognitive users to detect the primary users. We focused in solving this problem by utilizing the diversity schemes using multiple antennas at the cognitive user detector.

Furthermore, we consider the fact that in cognitive radio environment cognitive devices might suffer a low SNR problem which makes the task even harder, because we cannot estimate the channel. Studying the conventional diversity schemes and considering the mentioned constrains we proposed two schemes for blindly detecting the primary users' signals with diversity combining. The first combining scheme is based on the EGC idea of using an equal gain weights for combining, while keeping in mind, not to use the channel information when deciding the weights. A theoretically derived threshold has been proposed for fixing the false alarm probability of this scheme. The second proposed scheme is based on the MRC weights, where it changes the amplitude and phase of the weights corresponding to the eigenvectors of the received signal's covariance matrix. Both schemes have been analyzed theoretically and numerically. Simulation results show the improvement added to the system's detection performance using the proposed schemes without knowing about the channel state information.

In the practical part of this research we utilize an important candidate for realizing software defined radio technology called GNU Radio system. At the beginning we build a communication system using GNU Radio devices. Then we create a spectrum sensor device as an implementation for the first part of this thesis. We derived the sensitivity performance of this GNU Radio device by two means. Once by changing the number of samples used in detection, then, by changing the SNR level of the received signal.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently Wireless communication applications have been developing rapidly. Most of these applications, such as WLAN, WiMAX, LTE and so on, have a fixed assigned spectrum bands. This create two main problems, one is the scarcity of the frequency spectrum, and the other is the variation in temporal and geographical insufficient utilization of the fixed spectrum bands. To overcome this problem Cognitive Radio have been proposed as a solution by realizing dynamic spectrum access (DSA).
Spectrum sensing is crucial to achieve dynamic spectrum access in cognitive radio. Since it is important to sense a very low signal power (around -100 dBm), we had to find a robust method to achieve this sensitivity using the available sensing approaches.
Our main concern was to utilize multiple antennas signal processing schemes for improving the sensing performance. Diversity schemes are one of the best solutions to overcome the fading channel using multi-antenna. However, to utilize one of the multi-antenna processing schemes (as diversity) in cognitive radio environment we face the problem of estimating the channel state information (CSI) of the primary user's signal. To prevail over this problem we use blind diversity combining schemes.
Based on the two primary diversity combining schemes, EGC and MRC, we proposed two combining schemes to be utilized in blind detection without needed knowledge about CSI. The first one is based on EGC, and is called the quantization weights (QUAL) combining scheme. QUAL scheme results in a non fixed P_{FA}, so we have to derive a new threshold to fix the real P_{FA}. The other one is the EVD scheme, which is based on MRC. As the evaluation results will show, these two schemes have almost the same performance in low SNR region.
After we finish the blind detection part, we will demonstrate a new technology on wireless communication systems called GNU Radio. GNU Radio is a software defined radio (SDR) candidate for implementing cognitive radio systems. It

uses C++ and Python with the interpreter compiler SWIG to interface between the software world and the hardware radio. Using GNU Radio we built two systems. One is a transceiver system with a lost packets recovery functionality. The other is a cognitive radio sensor using energy blind detection. We evaluated both systems and demonstrate how they work in the result section. As for the sensor, we managed to evaluate the sensitivity and probability of detection of the GNU Radio hardware device.
The rest of the paper is organized as follows. At first, in chapter 2 we give a fast overview about cognitive radio. Then chapter 3 shows important Probability Density Function (PDF) statistics. After that, in chapter 4 we explain some basics for spectrum sensing and detection theory. Chapter 5 is about wireless channels. At the first part of this chapter we explain the wireless channel's problems, while in the second part some solutions for these problems are shown. Finally, we get to chapter 6 at which we explain and analyze the proposed schemes. In chapter 7 the simulation results of the proposed schemes are shown and discussed. The practical part of the research about GNU Radio is explained in chapter 8. In this chapter we give an overview about GNU Radio hardware and software, then we explain both of our systems, followed by the results. Finally appendix A shows how to install GNU Radio in your PC.

Chapter 2

Cognitive Radio

Due to the scarcity of the frequency spectrum and the variation of temporal and geographical utilization of the spectrum, cognitive radio have been proposed as a key technology to overcome those impediments. A cognitive radio system can be defined as an intelligent wireless communication system that is aware of its environment and uses some learning algorithms to learn from the surrounding radio environment and adapts its parameters to adequate with it. Cognitive radio (CR) systems still do not have a clear definition, or an exact functions. The mentioned definition is one among many definitions of CR. For a simple CR there are some functions which can be extracted from the famous cognitive cycle, fig.2.1.

- We will simply discuss about this figure, starting from the spectrum sensing step. This is our research subject, Detecting unused frequency bands and spectrum holes which can be shared without harmful interference with other users (spectrum holes are a primary user's (PU) frequency bands, however, at a particular time and specific geographic location, these bands are not being utilized). This function (Spectrum Sensing) can be implemented through many approaches as :

    1. Transmitter Detection: is based on detecting the weak signals of the primary transmitters. In this paper we consider this kind of detection to reach our goal.

    2. Cooperative Detection: incorporates the information from multiple cognitive users (CU) to improve the PU detection performance. This kind of detection can be implemented using one of two technologies. Centralized and distributed cooperative detection. The advantage of this type of detection is it's ability to overcome the hidden terminal

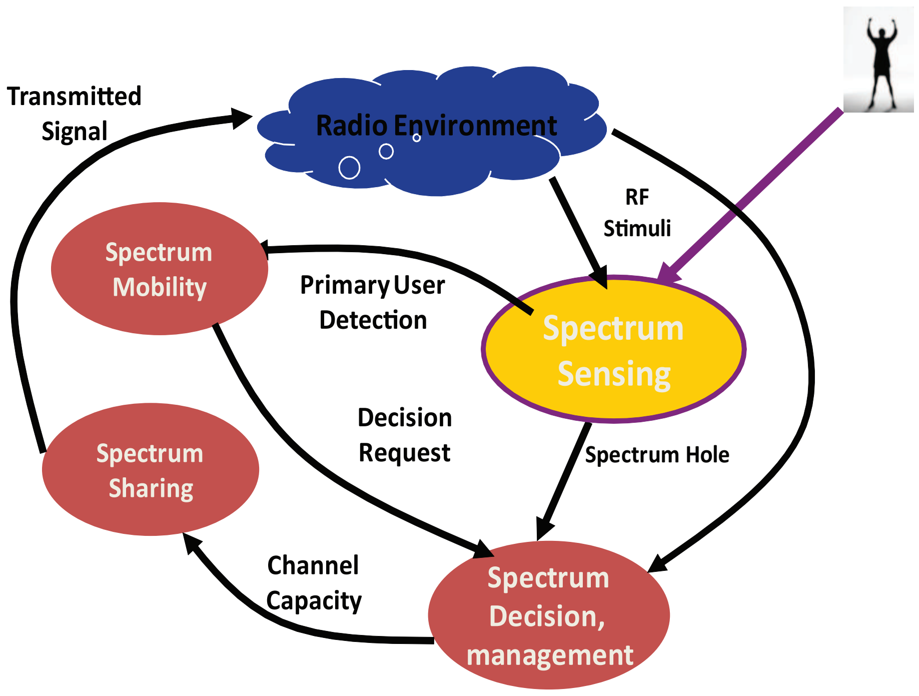


Figure 2.1: Basic Cognitive Radio Cycle

and shadowing problems.

3. Interference Temperature Detection: accounts for the cumulative RF energy and sets a maximum level called Interference temperature. Depending on this level we decide whether to use the band or not.

4. Receiver Detection: detects the PU receiver instead of the PU transmitter. This type of detection needs infrastructure detectors to be set.

- After explaining about spectrum sensing function we move to the spectrum management functionality of CR. By utilizing this function a CU should be able to capture the best available spectrum to meet the user's communication requirements.

- A CU should have a spectrum mobility functionality to be able to maintain a seamless communication requirements during the transition to better spectrum hole.

- Finally, the spectrum sharing step, which is necessary for a CU systems to provide the fair spectrum scheduling method among coexisting CU users.
- Moreover, a CU system should have the ability for orientation and prioritizing by binding the observation to a previously known set of radio stimuli.
- It should also have methodologies for learning and planing from the previous experience to the future improvement of the system.
- Finally, using all the previous parameters, CU system should be able to decide, among the available candidates, and act, by initiating its parameters based on its decision.

- One of the main advantages of CR system is its reconfigurability. This advantage is based on using the SDR (Software Defined Radio) platform in CR system. In SDR system the radio"s physical layer behavior is primarily defined using a software. Furthermore, it accepts fully programmable traffic and control information, while at the same time it can change its initial configuration to satisfy the user requirements. Not lastly, SDR have the ability to support a broad range of frequencies, air interfaces, and application software. One of the hardware implementation for SDR system is GNU Radio (USRP, Universal Software Radio Peripheral). In the last chapter of this thesis we will discuss our two implemented projects using GNU Radio and USRP.

Chapter 3

Important statistics Probability Density Functions (PDFs)

3.1 Introduction

As we have mentioned before, spectrum sensing is one of the key points to realize dynamic spectrum access for cognitive radio. To achieve an optimized spectrum sensing process, we have to intensely analyze the sensed data and then adjust the detector to achieve optimal detection. Therefore, in this chapter we are going to analyze, discuss a set of data distributions, which some of are well known, while the others might be not widely known distributions.

3.2 Gaussian distribution

Gaussian Random Distribution (also referred to as Normal distribution) is one of the most important distributions in the probability theory and detection theory. Under some conditions, any random variable X consist of general distributed components tends to have Gaussian distribution if the number of its variables becomes large enough (also referred to as Central Limit Theorem). The normal probability density function can be expressed as

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp[-\frac{1}{2\sigma^2}(x-\mu)^2], \tag{3.2.1}$$

where, μ is the mean and σ^2 is the variance of x. As in Fig.3.2.1, the normal distribution can be characterized by its variance and mean as, $X \sim \mathcal{N}(\mu, \sigma^2)$. Therefore, if we knew μ and σ^2 of the normal distribution, we can analyze the signal behavior. A specific case of a normal distribution in a wireless communication system is AWGN (Additive White Gaussian Noise) which have zero mean and unity variance. When the mean is zero the moment of normal distribution is given by

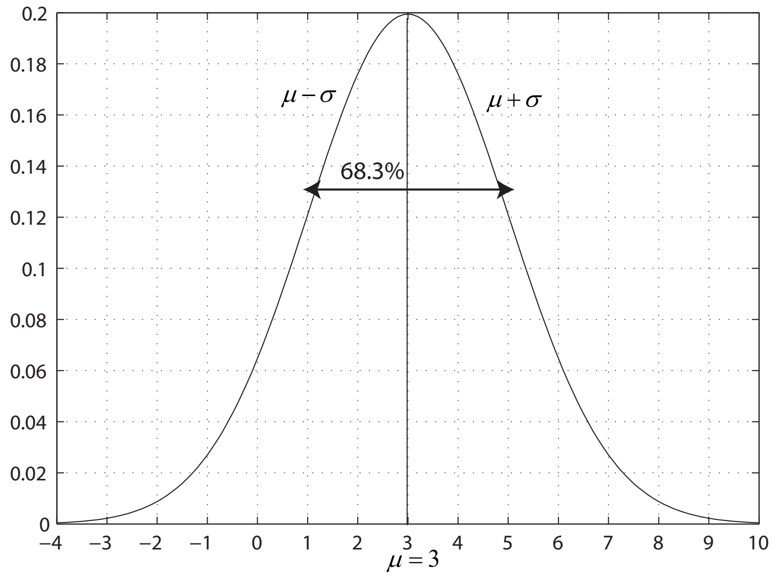


Figure 3.2.1: Basic Gaussian distribution

$$E(x^n) = \begin{cases} 1.3.4\ldots(n-1)\sigma^n & \text{n even} \\ 0 & \text{n odd} \end{cases}. \tag{3.2.2}$$

The cumulative distribution function (CDF) for $\sim \mathcal{N}(0,1)$ distribution is expressed as

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}t^2)\text{dt}. \tag{3.2.3}$$

Since our concern is about detection theory (deciding the probability of false alarm and mis detection), it is unavoidable to mention about the right tail distribution of a normal random variables. The Gaussian right tail distribution is defined as (error probability) $Q(x) = 1 - \Phi(x)$, and can be expressed as

$$Q(x) = \int_{x}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}t^2)\text{dt}. \tag{3.2.4}$$

It is known that Q function can be expressed in term of the error function as

$$Q(x) = \frac{1}{2}\text{erfc}(\frac{x}{\sqrt{2}}). \tag{3.2.5}$$

Figure 3.2.2 shows the complementary of right tail probability for a Gaussian distribution.

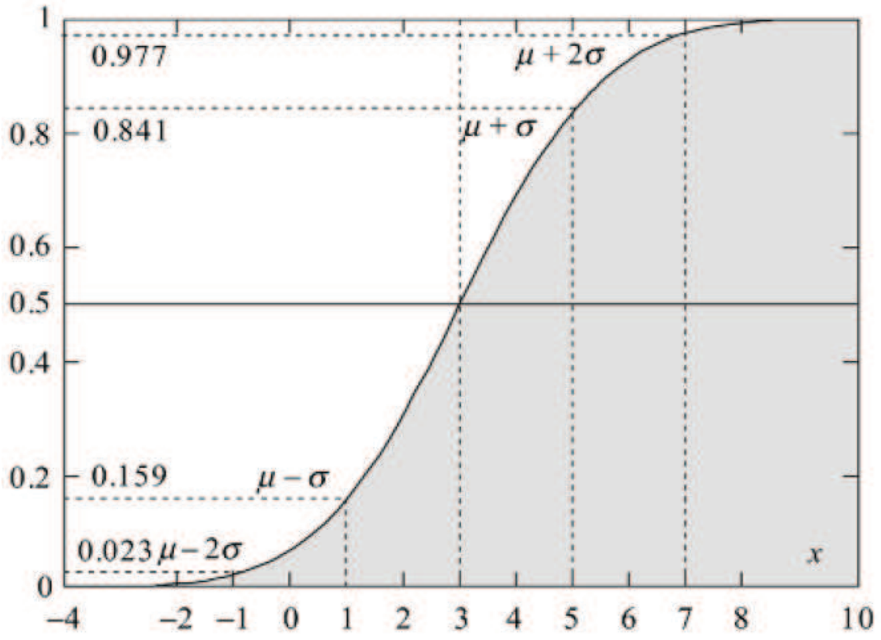


Figure 3.2.2: Complementary of right-tail probability for normal distribution.

3.3 Chi-Square(Central)

The chi-square distribution straddles to both exponential and Gaussian distributions. Chi-square PDF has v degrees of freedom, and is given by

$$p(x) = \begin{cases} \frac{1}{2^{\frac{v}{2}}\Gamma(\frac{v}{2})} x^{\frac{v}{2}-1} \exp(-\frac{1}{2}x) & x > 0 \\ 0 & x < 0 \end{cases}, \tag{3.3.1}$$

where, $\Gamma(.)$ is the Gamma function. The chi-square degrees of freedom considered to be $v > 0$. If we have $X = \sum_{i=1}^{v} x_i^2$, where $x_i \sim \mathcal{N}(0,1)$ and the x_is are IID among each other, then X follows the chi-square distribution, denoted as χ_v^2. For χ_v^2 we have

$$E(x) = v, \tag{3.3.2}$$

$$var(x) = 2v. \tag{3.3.3}$$

As for the probability of the right tail (for signal detection concern) it can be found in [7]. It is important to notice that the distribution of the output of energy detector follows chi-square distribution.

3.4 Chi-Square (Non Central)

In the previous sub section we considered the case where the x_is components of the chi-square are $\sim \mathcal{N}(0,1)$. If the random variables x_is do not have zero mean, the output will be chi-square (Non Central), with non centrality parameter $\lambda = \sum_{i=1}^{v} \mu_i^2$. From Fig.3.4.1 we can notice that for a chi-square distribution, the larger the number of the summed x_i the closer to the Gaussian distribution we are. Non central chi-square denoted as $\chi_v'^2(\lambda)$, and its PDF is given by

$$p(x) = \begin{cases} \frac{1}{2}(\frac{x}{\lambda})^{\frac{v-2}{4}} \exp[-\frac{1}{2}(x+\lambda)] I_{\frac{v}{2}-1}(\sqrt{x\lambda}) & x > 0 \\ 0 & x < 0 \end{cases}, \tag{3.4.1}$$

where $I_r(u)$ is the modified Bessel function of the first kind. The mean and variance of chi-square non central distribution is expressed as

$$E(x) = v + \lambda, \tag{3.4.2}$$

$$var(x) = 2v + 4\lambda. \tag{3.4.3}$$

As for the the right-tail probability of the non central chi-square it can be calculated using generalized Marcum Q-function as in [7].

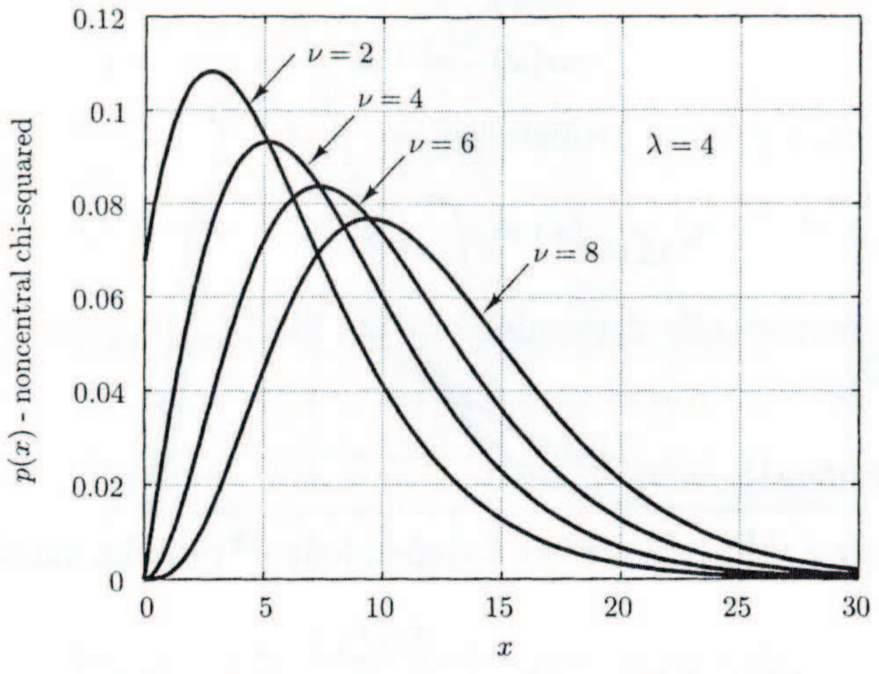


Figure 3.4.1: Chi-Square Non Central distribution with

3.5 Rayleigh Distribution

In wireless communication systems there are many channel models to help us emulating the wireless channels. Depending on the channel's time and frequency characteristics the model changes. For example, if we consider large scale fading or small scale fading the channel model will be different. For small scale fading channels we can use one of four channels models depending on both time and frequency characteristics (coherence time, coherence frequency). Based on the channel's coherence frequency we have flat fading and Frequency Selective fading. On the other hand, based on channel's coherence time we classify the channel into fast and slow fading channels. It is common in wireless communication to use Rayleigh distribution to model the envelope of a fully scattered multi-path, without direct ray received signal, which endure a flat fading channel. The Rayleigh distribution PDF is given by

$$p(r) = \begin{cases} \frac{r}{\sigma^2}\exp(-\frac{1}{2\sigma^2}(r^2)) & r > 0 \\ 0 & r < 0 \end{cases}, \tag{3.5.1}$$

where, r is considered as a random variable given by $r = \sqrt{x_1^2 + x_2^2}$, σ^2 is the variance for the random variables x_1 and x_2. Rayleigh distribution has mean and variance expressed by

$$E(r) = \sigma\sqrt{\frac{\pi}{2}}, \tag{3.5.2}$$

$$var(r) = \sigma^2(2 - \pi/2). \tag{3.5.3}$$

3.6 Generalized Extreme Value Distribution

In chapter 6 we are going to present our proposed scheme for blind detection. One of these scheme based on choosing the maximum of multiple branches of the receiving antennas after multiplying them by phase weights (refer to chapter 6 for more details). The process of choosing the maximum branch results in a distribution called extreme value distribution.

Definition:if a large number of independent random variables are generated from the same probability distribution, then we take their maximum values, the resulting distribution of the maximized values is approximated as a generalized extreme value (GEV) distribution.

The GEV was first introduced by Jenkinson (1955) and it has three types of distributions depending on the shape factor (which will be introduce later on this section). It is important to know cumulative distribution function (CDF) of GEV because we are going to use it in our calculation of the threshold for the false alarm probability (one of the error probabilities in detection theory, see chapter 4). The CDF of three types of GEV is given as [10]:

- Type 1, (Gumbel-type distribution), for $\xi = 0$

$$F(x;\mu,\sigma) = \exp(-e^{-(x-\mu)/\sigma}), \tag{3.6.1}$$

- Type 2, (Frechet-type distribution), for $\xi > 0$

$$F_x(x) = \exp -(\frac{x-\mu}{\sigma})^{-\xi} \qquad \mu < x < \infty, \tag{3.6.2}$$

- Type 3, (Weibull-type distribution), for $\xi < 0$

$$F_x(x) = \exp -(\frac{\mu-x}{\sigma})^{\xi} \qquad \mu > x > -\infty, \tag{3.6.3}$$

where σ and μ are the standard deviation and the mean of the original random variables, and ξ is a new parameter called the shape parameter and it can be used to model a wide range of tail behavior. The above three distributions corresponds to different tails distribution. $\xi > 0$ is for polynomially decreasing tail function. $\xi = 0$ is for exponentially decreasing tail function (Gaussian tails). Finally, $\xi < 0$ is for short limited tails. Our concern will be the case when $\xi = 0$ (Gumbel distribution), which follows the following PDF

$$p_X(x) = e^{-e^{-(x-\mu)/\sigma}}\frac{1}{\sigma}e^{-(x-\mu)/\sigma} \qquad -\infty < x < \infty. \tag{3.6.4}$$

The mean and variance of the Gumbel distribution are given as

$$E(X) = u + \beta\sigma = \mu + 0.57722\sigma, \tag{3.6.5}$$

$$Var(X) = \frac{\pi^2}{6}\sigma^2, \tag{3.6.6}$$

where β is Euler’s value.

Chapter 4

Spectrum Sensing and Detection Theory

4.1 Background of Detection Theory

In this subsection we introduce a general background about the detection theory. Starting from a general detection principles, followed by the optimal detector principles with full knowledge about the signal parameters (coherent detector). Then, we introduce detection with unknown parameters (non coherent detector). Finally, we show a comparison between the non coherent and coherent detection.

4.1.1 General Detection Principles

Figure 4.1.1 shows the principles of detection. To explain this figure let us consider the following equation

$$x[n] = \begin{cases} w[n] & H_0\text{(signal is absent)}, n = 0, 1, \ldots, N-1 \\ A + w[n] & H_1\text{(signal is exist)}, n = 0, 1, \ldots, N-1 \end{cases} . \quad (4.1.1)$$

For simplicity, In equation (4.1.1) we consider only a constant DC signal (A), so the received signal x[n] has two hypotheses. The first hypothesis is H_0, for $x[n] = w[n]$, where, w[n] is Additive White Gaussian Noise (AWGN). The second hypothesis is H_1, for $x[n] = A + w[n]$, where, A is the signal level. In a wireless system A is varying, and is equivalent to hs[n], where h is the Rayleigh channel and s[n] is the transmitted symbols. In this case we consider both hypotheses have the Gaussian distribution. The evaluation process of a detector is done by using a correct detection probability, an error detection probability or both of them.

As in Fig.4.1.1 there are 3 detection probabilities, one is correct probability, and two are error probabilities. The correct one can be calculated using the

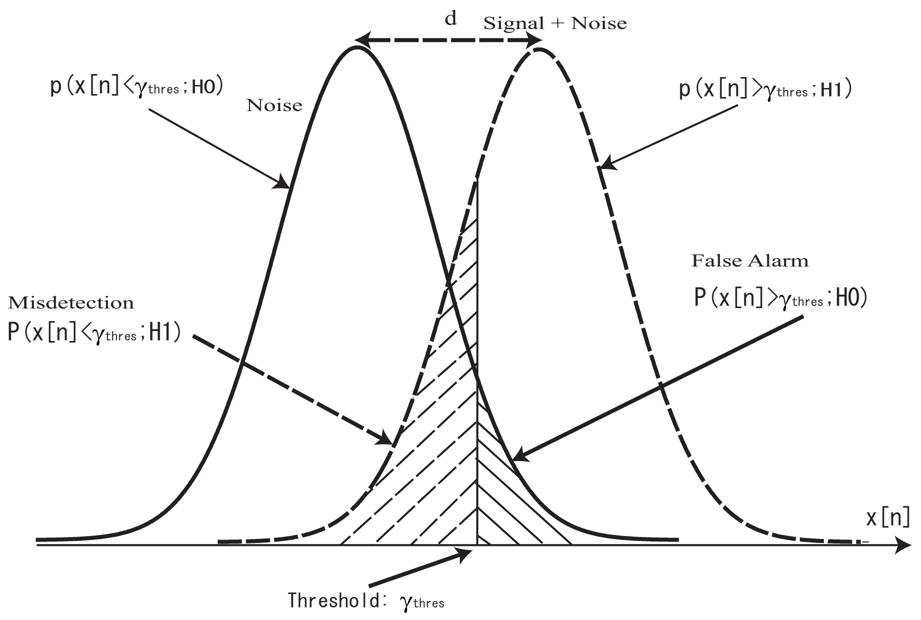


Figure 4.1.1: Principle of Detection theory

conditional PDF $p(u(x) > \gamma_{thres}; H_1)$, where u(x) is the test statistic. Test statistic changes from one detector to another, but in this example it is equivalent to $u(x) = (1/N) \sum_{n=0}^{N-1} x[n]$, and follows Gaussian distribution. The detector decide one of the two hypotheses according to the following equation

$$u[x] \begin{array}{ccc} > \gamma_{thres} & \overset{decide}{\rightarrow} & \text{signal exist:} H_1 \\ < \gamma_{thres} & \overset{decide}{\rightarrow} & \text{signal does not exist:} H_0 \end{array} . \tag{4.1.2}$$

According to Fig.4.1.1, the correct detection probability P_D occurs if the test statistic of the observed sample/s is larger than a certain threshold in the case of H_1 (decide H_1 when H_1 is true). As for Type 1 error probability, it occurs when the detector decide H_1, but H_0 is true. That is, the probability that the observed sample/s is larger than a certain threshold in the case of H_0 is true. Type 1 error probability is also called probability of false alarm P_{FA}. Moreover, Type 2 error probability occurs when the detector decide H_0, but H_1 is true. That is, the probability that the observed sample/s is less than a certain threshold in the case of H_1 is true. Type 2 error probability is also called probability of miss detection P_{MD}. As for Gaussian PDF the P_{FA} for x[0] can be shown as in [7]

$$\begin{aligned}
P_{FA} &= P[\mathcal{H}_1 : \mathcal{H}_0] \\
&= Pr[x[0] > \gamma_{thres}; \mathcal{H}_0] \\
&= \int_{\gamma_{thres}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}t^2)dt\text{'} \\
&= Q(\gamma_{thres})
\end{aligned} \tag{4.1.3}$$

where, Q is the Gaussian right tail probability. The probability of detection can be calculated in the same way.

$$\begin{aligned}
P_D &= P(\mathcal{H}_1 : \mathcal{H}_1) \\
&= Pr[x[0] > \gamma_{thres}; \mathcal{H}_1] \\
&= \int_{\gamma_{thres}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(t-A)^2)dt\text{'} \\
&= Q(\gamma_{thres} - A)
\end{aligned} \tag{4.1.4}$$

There is a trade off between P_{FA} and P_{MD} based on the chosen threshold (γ_{thres}). If we increase the threshold, P_{FA} will decrease and P_{MD} will increase. On the other hand, if we decrease the threshold, P_{MD} will decrease and P_{FA} will increase.
Using the mentioned probabilities we can evaluate the system performance. A common evaluation method is to plot P_{MD}/P_D versus SNR. Another evaluating method is to plot P_{MD}/P_D versus P_{FA} and it is called complementary/Receiver Operating Characteristics (CROC/ROC).

4.1.2 Optimal Detectors (NP)

Neyman-Pearson Theorem: To maximize P_D for a given $P_{FA} = \alpha$ decide $\mathcal{H}_1$ if

$$L(x) = \frac{p(x;\mathcal{H}_1)}{p(x;\mathcal{H}_0)} > \gamma_{thres}, \tag{4.1.5}$$

where, the threshold γ_{thres} can be found from

$$P_{FA} = \int_{\{x:L(x)>\gamma_{thres}\}} p(x;\mathcal{H}_0)dx = \alpha, \tag{4.1.6}$$

and L(x) is the likelihood ratio (liklihood of $\mathcal{H}_1$ and liklihood of $\mathcal{H}_0$). Equation (4.1.5) referred to as likelihood ratio test (LRT). Lets Consider equation (4.1.1), under $\mathcal{H}_1$, $X \sim \mathcal{N}(0, \sigma^2 I)$, while under $\mathcal{H}_0$, $X \sim \mathcal{N}(A1, \sigma^2 I)$. Then NP detector decide $\mathcal{H}_1$ if

$$\frac{\frac{1}{(2\pi\sigma^2)^{N/2}} \exp[\frac{1}{2\sigma^2} \sum_{n=0}^{N-1}(x[n]-A)^2]}{\frac{1}{(2\pi\sigma^2)^{N/2}} \exp[\frac{1}{2\sigma^2} \sum_{n=0}^{N-1}(x[n])^2]} > \gamma_{thres}. \tag{4.1.7}$$

After taking logarithm and simplify the equation we have

$$\frac{1}{N}\sum_{n=0}^{N-1} x[n] > \frac{\sigma^2}{NA} ln\gamma_{thres} + \frac{A}{2} = \gamma'_{thres} \tag{4.1.8}$$

where, as we defined before $u(x) = (1/N)\sum_{n=0}^{N-1} x[n]$ is the test statistic under each hypothesis. Finally, by considering $P_{FA} = Pr[u(x) > \gamma'_{thres}; \mathcal{H}_0]$ and $P_D = Pr[u(x) > \gamma'_{thres}; \mathcal{H}_1]$, we have a relation ship between P_D and P_{FA} as

$$\begin{aligned} P_D &= Q(\frac{\sqrt{\sigma^2/N}Q^{-1}(P_{FA}) - A}{\sqrt{\sigma^2/N}}) \\ &= Q\left(Q^{-1}(P_{FA}) - \sqrt{\frac{NA^2}{\sigma^2}}\right) \end{aligned}, \tag{4.1.9}$$

which can be used to evaluate the detector ROC performance. In wireless communication literature NP detector is called matched filter.

4.1.3 Detection with unknown Signal Parameters

In previous section we considered detection with knowing all signal parameters, however in this section we consider the detection performance with some or all of the signal‘s parameters are unknown. In detection theory literature, there are some known tests for the case of unknown signal parameters. As an example, GLRT (Generalize Likelihood Ratio Test) and Bayesian test, both as a general test. Furthermore, Wald test and Rao test can be used for a specific unknown parameters. Some of the common unknown parameters are arrival time, frequency band, amplitude, PDF and so on. The more the knowledge we have about these parameters the better the detection accuracy will be. For simplicity, we consider GLRT with the same signal as in equation (4.1.1), but with no knowledge about the amplitude (A). The GLRT detector decides $\mathcal{H}_1$ if the $L_G(x) > \gamma_{thres}$, where $L_G(x)$ is the GLRT likelihood function of the two hypotheses $\mathcal{H}_1$ and $\mathcal{H}_0$. After doing some simplification for the $L_G(x)$ function we have

$$\begin{aligned} lnL_G(x) &= -\frac{1}{2\sigma^2}\left(\sum_{n=0}^{N-1} x^2[n] - 2\bar{x}\sum_{n=0}^{N-1} x[n] + N\bar{x}^2 - \sum_{n=0}^{N-1} x^2[n]\right) \\ &= -\frac{1}{2\sigma^2}(-2N\bar{x}^2 + N\bar{x}^2) \\ &= \frac{N\bar{x}^2}{2\sigma^2} \end{aligned} \quad . \tag{4.1.10}$$

That is, we decide $\mathcal{H}_1$ if $|\bar{x}| > \gamma'_{thres}$, where $\bar{x} = \sum_{n=0}^{N-1} x[n]$. From equation (4.1.10) we can confirm that the knowledge of the signal amplitude is not necessary for deciding the new threshold γ'_{thres}. This kind of detectors called non coherent blind detectors. In section 4.1.4 we show a comparison between coherent and non coherent detectors. From This comparison we confirm the advantage of coherent detectors, because it utilizes the knowledge of the received signal’s parameters.

4.1.4 Comparison between Coherent and Non Coherent Detectors

The lack of knowledge about the signal's parameters results in a degradation of detection performance when compared with the optimal matched detector. In this section we compare a GLRT detector (unknown signal's parameter detector) performance with clairvoyant (NP, matched detector which have full knowledge about the signal's parameters). The upper bound of any suboptimal detector is decided using the performance of the optimal detector (matched filter NP), which expressed by

$$P_D = Q(Q^{-1}(P_{FA}) - \sqrt{d^2}), \tag{4.1.11}$$

where $d^2 = \xi$ is the deflection coefficient and ξ is the signal energy. The deflection coefficient is a measure to express the difference between the mean of the signal + noise PDF and the mean of noise PDF. As for the GLRT detector, consider

$$x[n] = \begin{cases} w[n] \qquad H_0\text{(signal is absent)}, n = 0, 1, \ldots, N-1 \\ s[n] + w[n] H_1\text{(signal is exist)}, n = 0, 1, \ldots, N-1 \end{cases}, \tag{4.1.12}$$

where $s[n]$ is deterministic and completely unknown signal. To obtain $\hat{s}[n]$ using the MLE (Maximum Likelihood Estimation) under $\mathcal{H}_1$, we maximize the likelihood function over the signal samples. Then, we obtain the MLE values as $\hat{s}[n] = x[n]$. Subtitute the MLE values in the GLRT's $\mathcal{H}_1$ equation, we get

$$\frac{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}}}{\frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left(\frac{1}{2\sigma^2} \sum_{n=0}^{N-1} x^2[n]\right)} > \gamma_{thres}, \tag{4.1.13}$$

by simplification, we deduct that the test statistic $u(x) = \sum_{n=0}^{N-1} x^2[n] > \gamma'_{thres}$, which equivalent to energy detection. Assuming large N, we can find the energy detector's deflection coefficient as

$$d_{ED}^2 = \frac{(\frac{\xi}{\sigma^2})^2}{2N}. \tag{4.1.14}$$

Where, the Matched filter deflection coefficient is given by

$$d_{ED}^2 = \frac{\xi}{\sigma^2}. \tag{4.1.15}$$

Combining both (4.1.14) and (4.1.15) and using a given P_D to calculate the necessary input SNR in dB as [7]

$$\begin{aligned} 10log_{10}\eta_{MF} &= 10log_{10}d^2 - 10log_{10}N \\ 10log_{10}\eta_{ED} &= 5log_{10}d^2 + 1.5 - 5log_{10}N \end{aligned} \tag{4.1.16}$$

Equation (4.1.16) is plotted in Fig.4.1.2. As shown, we notice the degradation of energy detection comparing to matched filter detection, at $N = 1000$, around 11.5 dB.

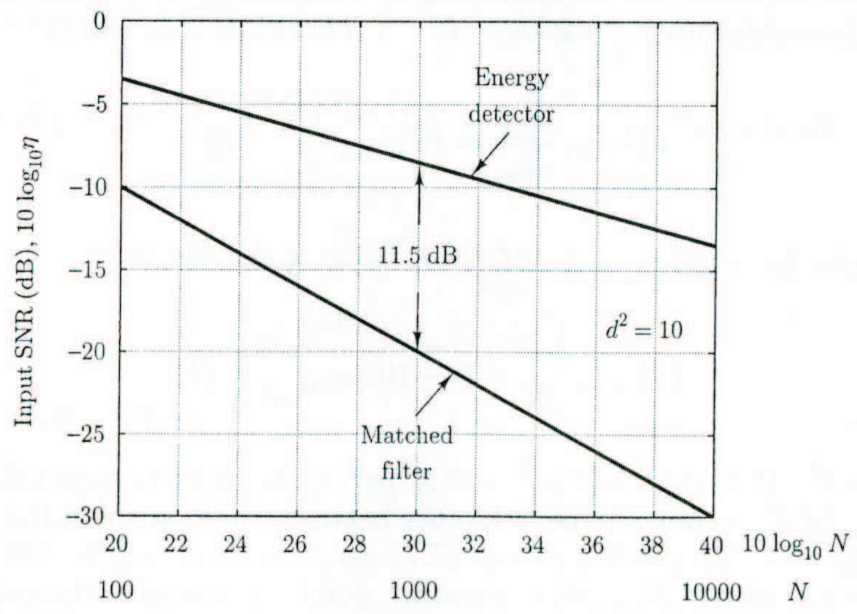


Figure 4.1.2: Required input SNR for given detection performance.

4.2 Overview for Transmitter Detectors in Spectrum Sensing

As we have mentioned before, one of the spectrum sensing approaches is the transmitter detection. For detecting the PU transmitter signal, depending on our target we select one of the following detectors, energy detector, cyclostationary detector and matched filter detector. In this subsection we will discuss both matched filter and cyclostationary detector. Since we are using it in the proposed methods, energy detection is going to be discussed in details in next sections, for AWGN and Rayleigh fading channels.

4.2.1 Matched filter

In Sects. 4.1.2 and 4.1.4, the principle of optimal detection have been explained. Matched filter is an optimal detector, it has the following characteristics:

1. Less sensing time due to coherency of the receiver.
2. Achieving maximum SNR due to the knowledge of the PU signal (modulation type, time synchronization, frequency band and etc...).
3. Channel equalization and carrier synchronization is possible.

On the other hand, the main draw back of this method is that its required a dedicated receiver for each signal type. Moreover, it requires the knowledge of the primary signal and channel which is difficult to obtain in a cognitive radio environment (because it considered in low SNR).

4.2.2 Cyclostationary Detection

Cyclostationary detection utilizes the periodicity of the signal due to the modulation type, coding or by intentionally produced periodicity to help in estimation the channel. Cyclostationary detection is recognized by its robust signal detection under low SNR region. This is because AWGN is random, therefore by knowing the periodicity of the signal, the detector can differentiate between the signal and AWGN. The Cyclostationary detection uses the Cyclic Autocorrelation Function (CAF) for detecting the signal as follows [15]

$$R^{\alpha}(\tau) = \begin{cases} \sigma_a^2 e^{j2\pi\alpha t_0} X \int G^*(f+\alpha)G(f)e^{j2\pi f\tau}df, & \alpha = \frac{k}{T_0} \\ 0, \quad otherwise \end{cases} . \tag{4.2.1}$$

The function $R^{\alpha}(\tau)$ is the CAF and α_0 is the cyclic frequency. As in equation (4.2.1), given α_0 the CAF can determine the correlation between spectral components of the signal separated, in frequency, by α_0. As for detection, the CAF has a nonzero value only for multiple integers of α_0.
Even though Cyclostationary detection is more powerful than ED in detecting the signals under low SNR, but it needs exhaustive computational complexity to achieve it's performance.

4.2.3 Energy Detection

We are going to discuss about the Energy Detector in details in the next section (4.3).

4.3 Spectrum Sensing Using Energy Detector

In Sect.4.1.3 we have already started to introduce the energy detection (ED), that it does not require any information about the signal. Then, in Sect.4.1.4 we compared the Non-coherent detection (ED) with coherent detection performance. In this section we are going to explain the energy detector for detecting a deterministic unknown PU signal in details. For now, lets assume AWGN channel, latter on we assume wireless channel (Rayleigh, multi-antenna Rayleigh). [6],[7],[8] Assuming the received signal as

$$Y(t) = \begin{cases} n(t) & H_0(\text{signal is absent}) \\ x(t) + n(t) & H_1(\text{signal is exist}) \end{cases}, \tag{4.3.1}$$

where *n(t)* is the AWGN with zero mean and variance σ^2, *x(t)* is the PU signal.

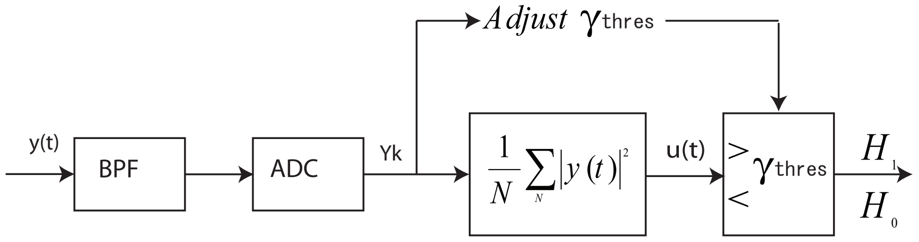

Adjust γ_{thres}
y(t)
BPF
ADC
Yk
$\frac{1}{N}\sum_N |y(t)|^2$
u(t)
$\gtrless \gamma_{\text{thres}}$
H_1
H_0

Figure 4.3.1: Basic Energy Detector.

As in Fig.4.3.1, we receive the signal at a certain center frequency, and then we do the squaring and averaging over the summation of *N* samples. By choosing a large *N*, the sensing time will increase, but on the other hand, the performance and accuracy of detection will also improve. The result of this process called test statistic (which mentioned in previous sections with the same or different definitions), noted as *U(t)*. The test statistic *U(t)* is described as:

$$U(t) = \sum_{n=1}^{N} |Y_n(t)|^2. \tag{4.3.2}$$

Then, *U(t)* is compared with the threshold and the detector decides the existence of the primary signal (similar to equation(4.1.2)) as follows:

$$U(t) \begin{array}{c} > \gamma_{thres} \overset{decide}{\longrightarrow} \qquad \text{signal exist} \\ < \gamma_{thres} \overset{decide}{\longrightarrow} \text{signal does not exist} \end{array}. \tag{4.3.3}$$

In the case of PU absence, the test statistic consists of summation of *N* squared independent identical (IID) variables with zero means, which results in central chi-square variables with *N* degree of freedom (DOF). If we consider large *N*, we can treat the test statistic as a normal distribution, as stated in the Central Limit Theorem (CLT). While in the case of PU presence the test statistic consists of summation of *N* squared IID variables with nonzero means, which results in Non-central Chi-square distribution with *N* DOF. By assuming large *N*, we can apply the CLT for signal distribution. Then, we consider the test statistic as normal distribution with mean and variance as follows:[6],[8]

$$U \sim \begin{array}{ll} \text{Normal}(\sigma_n^2, 2\sigma_n^4/N) & \text{under } H_0 \\ \text{Normal}(\sigma_n^2 + \sigma_x^2), 2(\sigma_n^2 + \sigma_x^2)^2/N) & \text{under } H_1 \end{array} . \tag{4.3.4}$$

As introduced in previous sections, we evaluate a detector performance by calculating P_{FA} and P_D/P_{MD}. We can calculate the ED performance's probabilities by two means. One is approximating the test statistics as a Gaussian distribution to find P_{FA}. The other is exact calculation of both P_{FA} and P_D/P_{MD}. At first we describe the approximating approach then we mention the exact approach.
Approximation approach: As shown in Fig. 4.3.2, since the false alarm probability is occupying the tail of a Gaussian distribution, we can find P_{FA} using the Q function for a given threshold as follows [8]

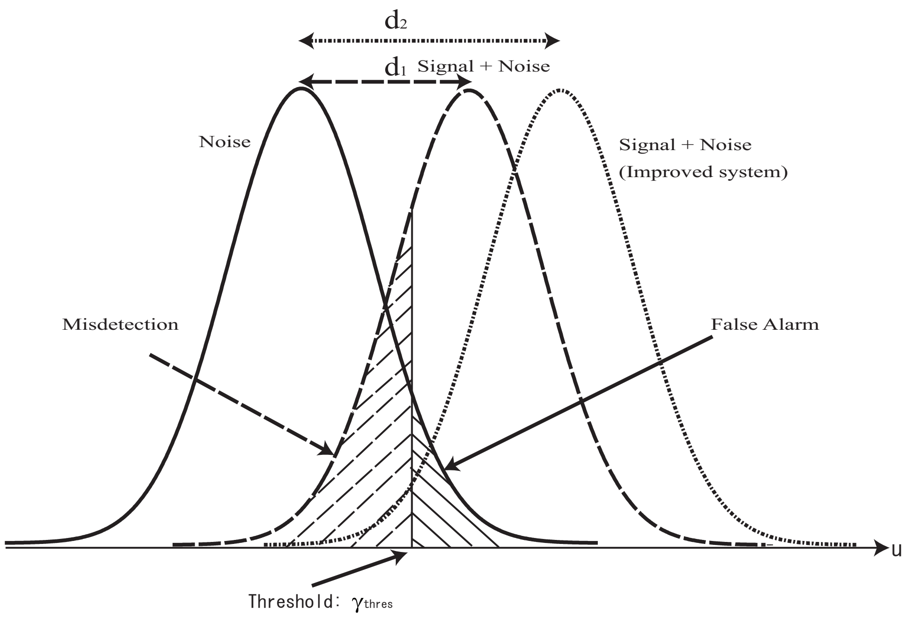


Figure 4.3.2: Principles of Gaussian distribution detection.

$$P_{FA} = Q(\frac{\gamma_{thres} - \sigma_n^2}{\sigma_n^2/\sqrt{N/2}}). \tag{4.3.5}$$

Furthermore, if we want to find the threshold for a certain false alarm we can solve equation (4.3.5) to find:

$$\gamma_{thres,FA} = \sigma_n^2 (1 + \frac{Q^{-1}(P_{FA})}{\sqrt{N/2}}). \tag{4.3.6}$$

Figure 4.3.2 shows the way of improving the detecting performance of an energy detector. Consider the curve for improved S+N (Signal + Noise) has d_2 mean (referenced from the Noise mean) is larger than the d_1 of normal (not improved) S+N curve. This can be utilized by either fixing the threshold to improve P_D or by adjust the threshold to decrease the P_{FA} and fix P_D. A point should be mentioned here, is that equation (4.3.5) should be considered when we have a fixed mean of the received signal. Otherwise we have to adjust the threshold according to the signal and noise mean.

Exact approach: From equation (4.3.2) we can see that the test statistic is the sum of the squares of 2z Gaussian variables, z is the time bandwidth product. Thus, from equation (4.3.1), under H_0, $U(t)$ follows a central chi-square distribution with 2z degrees of freedom (we've mentioned this before, but using N as another definition for 2z). On the other hand, under H_1 hypothesis, U(t) is a non-central chi-square distribution with 2z degrees of freedom and 2λ as a non-centrality parameter. As a result we have U(t) as

$$U(t) \sim \left\{ \begin{array}{ll} \mathcal{X}_{2z}^2 & H_0 \\ \mathcal{X}_{2z}^2(2\lambda) & H_1 \end{array} \right. . \tag{4.3.7}$$

Then we find the PDF of U(t) as

$$f_U(u) = \left\{ \begin{array}{ll} \frac{1}{2^z\Gamma(z)} u^{z-1} e^{u/2}, & H_0 \\ \frac{1}{2}(\frac{u}{2\lambda})^{\frac{z-1}{2}} e^{-\frac{2\lambda+u}{2}} I_{z-1}(\sqrt{2\lambda u}), & H_1 \end{array} \right. , \tag{4.3.8}$$

where $\Gamma(.)$ is the gamma function, and $I_v(.)$ is the v^{th}-order modified Bessel function. Using the definition of P_{FA} and P_D

$$\begin{array}{c} P_D = Pr(U > \gamma_{thres}|H_1) \\ P_{FA} = Pr(U > \gamma_{thres}|H_0) \end{array} . \tag{4.3.9}$$

Integrating the tail of the H_0 part of equation (4.3.8), we can find the false alarm probability as

$$P_{FA} = \frac{\Gamma(z, \lambda/2)}{\Gamma(z)}. \tag{4.3.10}$$

The CDF of equation (4.3.8) can be shown as

$$F_U(u) = 1 - Q_z(\sqrt{2\lambda}, \sqrt{u}), \tag{4.3.11}$$

where, Q_z is generalized Marcum Q-function. Hence,

$$P_D = Q_z(\sqrt{2\lambda}, \sqrt{u}). \tag{4.3.12}$$

Chapter 5

Wireless channels and Diversity Schemes

Since our research concerns about signal detection in wireless channel, it is necessary to discuss the different types of wireless channels. Each of these wireless channels has its specific causes and effects on the transmitted signal. Then, we move to discuss the means of overcoming these problems. Some of these solutions are already utilized in our research, while the others are either on the process of being utilized or cannot be utilized due to our system characteristics.

5.1 Wireless Channels

The wireless communication channels has three main types. One is a free path loss, which attenuates the signal depending on the traveled distance, it is also called large term (scale) path loss. Another is shadowing effect, which fluctuates the transmitted signal over smaller time period (compared to the free path loss). Finally, small scale (multi-path) fading, which cause a signal fluctuation even faster (and within smaller distance) compared to shadowing effect. The previously mentioned channel effects are due to distance traveled by the signal, reflection, diffraction, and signal scattering (in addition to the terminal mobility). In this section we will discuss the last channel effect, small scale fading. As for the small scale fading there are many factors should be taken into account. As, multipath propagation, speed of the mobile terminal, speed of surrounding objects, and the transmission bandwidth. We are going to start by explaining the parameters, upon which we classify the channels.

5.1.1 Time dispersive parameters: delay spread and coherent bandwidth

Excess delay (delay spread) is the relative delay of the i^{th} multi-path component as compared to the first arriving component and is given by τ_i. The maximum excess delay of the channel is given by $N\Delta\tau$, where N is the number of equally time separated multi-paths, and $\Delta\tau$ is an equal time separation between every path and the next one, e.g $\Delta\tau = \tau_1 - \tau_0$. The channel time dispersiveness properties can be quantified by using the rms excess delay. The rms excess delay is the square root of the second central moment of the power delay profile, given by [1]

$$\sigma_\tau = \sqrt{\bar{\tau^2} - (\bar{\tau})^2}, \tag{5.1.1}$$

where, $\bar{\tau}$ is the mean excess delay. Therefore the channel time dispersive parameters can be obtained from the power delay profile. It is important to mention that rms excess delay and coherence bandwidth (which is going to be explained later) are inversely proportional. Analogous to the delay spread parameters in the time domain, coherence bandwidth is used to characterize the channel in the frequency domain.
Coherence bandwidth is a statistical measure of the range of frequencies over which the channel can be considered flat, i.e. it passes all the frequency components with approximately equal gain and linear phase. Coherence bandwidth is approximated as

$$B_c = \frac{1}{50\sigma_\tau}. \tag{5.1.2}$$

5.1.2 Time varying parameters (frequency dispersiveness): Doppler Shift and coherence time

Doppler spread and coherence time describes the time varying nature of the channel in a small-scale region. **Doppler Shift** B_D is a measure of the received spectral broadening caused by the mobile radio channel. Encountering a Doppler shift caused to see the spectral of the received signal varying in the range of $f_c + f_d$ to $f_c + f_d$, where, f_d is the Doppler shift and f_c is the carrier frequency. Doppler shift depends on both the velocity of the mobile radio terminal and the angle between the direction of movement and the direction of arrival.

Coherence Time is the time domain dual of Doppler shift, and it is used to quantify the channel frequency dispersiveness property in time domain. Coherence time is defined as a statistical measure of the time duration over which the channel impulse response is invariant. We can relate both Doppler shift and coherence time as

$$T_c = \frac{1}{f_m}, \tag{5.1.3}$$

where, f_m is the maximum Doppler shift given by $f_m = v/\lambda$.
Finally, after discussing the channel parameters, we can classify the wireless channel due to small scale fading to 4 types:

- Flat Fading: Bandwidth of signal $<$ Bandwidth of channel and Delay spread $<$ Symbol period
- Frequency Selective Fading: Bandwidth of signal $>$ Bandwidth of channel and Delay spread $>$ Symbol period
- Fast Fading: High Doppler spread and Coherence time $<$ Symbol period
- Slow Fading: Low Doppler spread and Coherence time $>$ Symbol period

At the earlier stages of our research we focused on slow flat fading channel. Then, we started to consider frequency selective fading channels.

5.2 Diversity Schemes

As we introduced in the previous section, in this research we consider a flat fading channel. As far as our knowledge, the main counteract to flat fading channels is to use diversity combining schemes. The diversity principle is based on the fact that when a signal passes through an independent fading channels, the probability of experiencing a deep fading is low. To mitigate the fading channel we can utilize several types of diversity combining.

- In a frequency selective channel it is possible to utilize the frequency diversity. Because the spectral components of the frequency selective channel are varying by varying the band, then by considering receiving on a bandwidth larger than the coherent bandwidth we can achieve frequency diversity. This type of diversity is usually used in wide band signals.

- The other type is time diversity, which can be achieved by using the proper coding and interleaving schemes. To achieve this type of diversity it is necessary to transmit (using interleaving) over a time periods larger than the channel coherence time.

- In this paper, to mitigate the fading effect we are going to utilize space (antenna) diversity. There are two factors to be considered for achieving this type of diversity (lets consider the receiving space diversity not transmitter diversity). One factor is to use multiple antennas with a separation larger than 0.38 λ (wave length). This is in case we are experiencing uniform scattering environment and omni directional transmit and receive antennas. If we are using directional antennas, the antenna separation should be increased. The other factor is the scattering environment. It is more likely that the independent channels environment can be found under rich scattering environments. After receiving different faded signals we should use an effective combiner (the optimal one is the coherent combiner) to combine between the faded signals constructively to achieve additional array gain (array gain results from coherent combining, diversity gain results from experiencing independent fading channels). A common diversity linear combiner is shown in Fig.5.2.1

In this section, we analyze several space diversity schemes because we are going to use it in our evaluation of our proposed schemes as upper or lower bounds. The analysis method is by deriving the overall (combined) SNR versus both, Bit

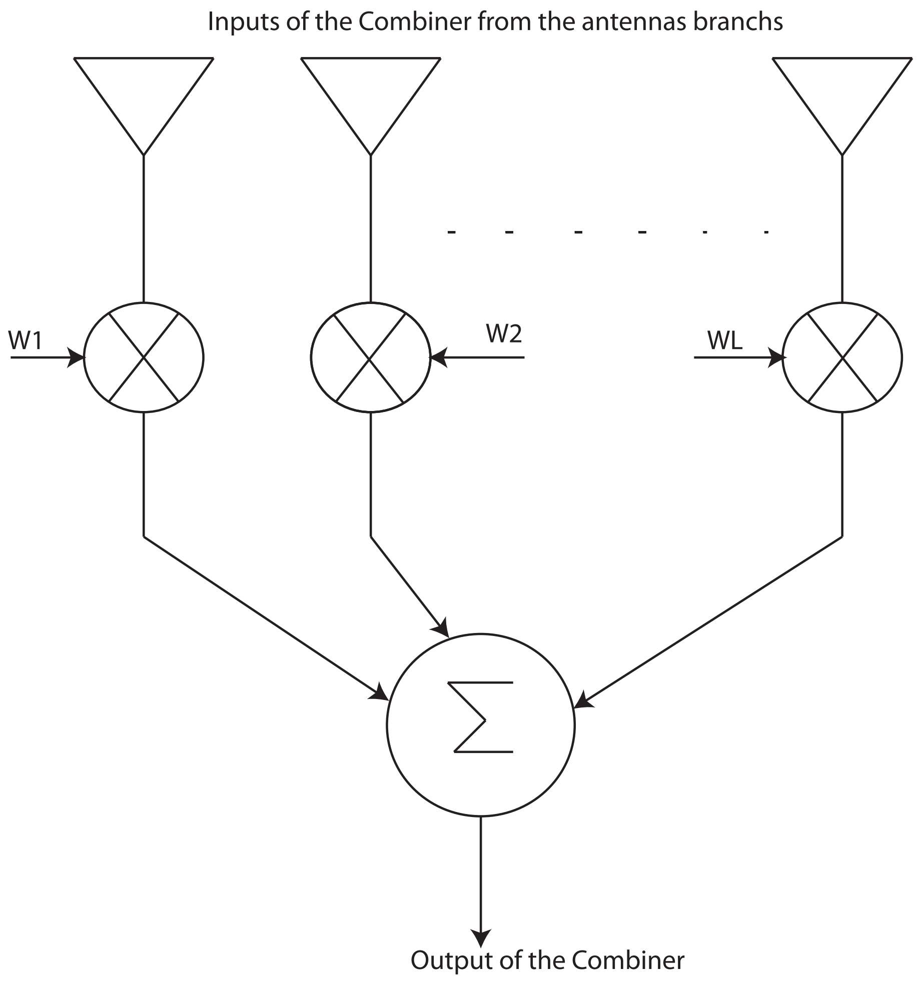


Figure 5.2.1: Space diversity linear combiner.

Error Rate (BER for BPSK modulation case), and the probability of detection [9],[12]. To avoid notation confusing we used γ_{thres} to note the detection threshold. Hereafter, we are going to use γ, γ_t or $\overline{\gamma}$ to denote the SNR.

5.2.1 EGC Diversity

1. SNR and P_b Analysis:

   We describe the EGC scheme assuming perfect knowledge of Channel State Information (CSI) because we are going to use it as the lower bound (optimal performance) for our evaluation of the proposed QUAL scheme (proposed schemes will be explained later on Ch. 6). Furthermore, the proposed QUAL shares the same principle with EGC, where it selects equal gain, variable phase weights. After that, theoretically we derive the detection performance which has been already discussed in other papers [4]. Considering a flat slow fading channel, the channel response is given by

   $$\mathbf{h}_i = \alpha_i e^{j\phi_i}, \tag{5.2.1}$$

   and the received signal at receiver side is expressed as

   $$\mathbf{y}_i = \mathbf{h}_i \mathbf{x} + \mathbf{n}_i, \tag{5.2.2}$$

   where $\mathbf{y}_i$ is the received signal vector at branch i, $\mathbf{n}_i$ is the AWGN of branch i. And $\mathbf{h}_i$ is the i^{th} antennas's channel response, follows Rayleigh distribution, where α is its amplitude, ϕ is the phase. In EGC scheme we combine the signals from the different antenna branches after multiplying them by weights equal to conjugate of the channel phase expressed as

   $$\mathbf{w}_i = \frac{|\mathbf{h}_i|}{\mathbf{h}_i} = e^{-j\phi}. \tag{5.2.3}$$

   EGC scheme achieves full phase combining coherency, which results in an output SNR γ_t equal to the sum of each branch's SNR γ_i, as expressed by

   $$\gamma_t = \sum_{i=1}^{L} \gamma_i = \frac{1}{N_0 L} \left(\sum_{i=1}^{L} \sqrt{E_s} \right)^2 . \tag{5.2.4}$$

   where, E_s is the average energy per branch. Later on we will see that the proposed QUAL scheme has partial phase coherence but not a full one.

   Then the averaged probability of bit error for 2-antennas case can be expressed as

   $$\bar{P}_{e,EGC} = \frac{1}{2} \left[1 - \sqrt{1 - \left(\frac{1}{1+\bar{\gamma}} \right)} \right] . \tag{5.2.5}$$

   where,$\bar{\gamma}$ is the averaged SNR, which is equal for all branches.

2. Detection Analysis:

   After combining the signals we apply the energy detector to the output combined signal. Then we can express the probability of detection for EGC output of AWGN channel as [4]

$$P_{dEGC} = Q_{Lu}\left(\sqrt{2\gamma_t}, \sqrt{\gamma_{thres}}\right), \tag{5.2.6}$$

   where u is the TW (time bandwidth) product, γ_{thres} is the threshold and $Q_u(a,b)$ is the generalized Marcum Q-function. To find the probability of detection under Rayleigh fading channel, we average equation (5.2.6) over the *PDF* of IID Rayleigh fading branches, as expressed in the following

$$f(\gamma_t) = \frac{1}{(L-1)!\overline{\gamma}^L}\gamma_t^{L-1}\exp\left(-\gamma_t/\overline{\gamma}\right), \tag{5.2.7}$$

   where $\bar{\gamma}$ is the average SNR. After averaging equation (5.2.6) over equation (5.2.7) we have the averaged detection probability of EGC over fading channel given by

$$\bar{P}_{d,EGC} = \alpha\left[G_1 + \beta\sum_{n=1}^{u-1}\frac{(\gamma_{thres}/2)^n}{2(n!)}F_1(L; n+1; \frac{\gamma_{thres}}{2}\frac{L\bar{\gamma}}{2L+L\bar{\gamma}})\right], \tag{5.2.8}$$

   where, G_1, F_1, β, and α are as defined in [4], and $\bar{\gamma}$ is the averaged SNR, which is equal for all branches.

5.2.2 MRC Diversity

1. SNR and P_b Analysis:

   Consider the channel as in equation (5.2.1). Using MRC scheme, we combine the signals from the different antenna branches after multiplying them by weights equal to the channel conjugate values. This means that we have combining coherency in both amplitude and phase. The output of the MRC combiner is expressed as:

   $$\mathbf{y} = \frac{\sum_{i=1}^{L} \mathbf{h}_i \mathbf{y}_i}{\sum_{i=1}^{L} \mathbf{h}_i \mathbf{h}_i^*}, \tag{5.2.9}$$

   where, L is the number of used antennas.

   Comparing to the EGC's total output SNR, we can see that the MRC's total SNR is larger.

   $$\gamma_{t,MRC} = \sum_{i=1}^{L} \frac{E_s}{N_0} = \sum_{i=1}^{L} \gamma_i = L\overline{\gamma}, \tag{5.2.10}$$

   where γ_i is the SNR of branch i, L is the number of used antenna. Thus, the total output SNR $\gamma_{t,MRC}$ equal to averaged SNR per branch times the number of antennas. Then, we find the averaged error probability as given by

   $$\bar{P}_{e,MRC} = \left(\frac{1-\Gamma}{2}\right)^{L} \sum_{m=0}^{L-1} \begin{pmatrix} L-1+m \\ m \end{pmatrix} \left(\frac{1+\Gamma}{2}\right). \tag{5.2.11}$$

   where $\Gamma = \sqrt{\overline{\gamma}/(1+\overline{\gamma})}$.
   It is important to mention that for amplitude coherence combining schemes (as MRC) we have to use a correct normalization factor to get a correct detection performance.

2. Detection Analysis
   As for the detection performance of MRC, the same way of deriving the detection performance for EGC can be applied to MRC. We average the P_D in AWGN channel over Rayleigh channels PDF to get the final equation, which can be calculated iteratively as in equation (9) in [3].

   $$\bar{P}_{d,MRC}(\bar{\gamma}, L) = \bar{P}_{d,MRC}(\bar{\gamma}, L-1) + \sqrt{\frac{\bar{\gamma}}{\pi}} \frac{e^{\frac{\gamma_{thres}^2}{1+\bar{\gamma}}}}{(L-1)!(1+\bar{\gamma})^{L-0.5}} I_1(2L-2) \tag{5.2.12}$$

5.2.3 Selection Diversity

1. SNR and P_b Analysis:

   If we consider the noise power at all antenna branches is the same, we deduct that SNR is equivalent to S+N (Signal plus Noise is equivalent to signal over noise ratio), this is the reason why in this paper we name SC (Selection Combining) as MSPN (Maximum Signal Plus Noise). Then, the output SNR is equal to the maximum branch's SNR. However, in this kind of diversity scheme, it is required to have detector at all branches to keep track of the branch SNR, then compare all of them to decide the maximum SNR. The average total SNR (the average combiner output SNR) is given by

   $$\gamma_{t,MSPN} = \bar{\gamma} \sum_{i=1}^{L} \frac{1}{i}. \tag{5.2.13}$$

   Then, we find the averaged probability of error as

   $$\bar{P}_{e,MSPN} = \frac{L}{2} \sum_{l=0}^{L-1} (-1)^l \frac{\begin{pmatrix} L-1 \\ l \end{pmatrix}}{1+l+\bar{\gamma}}. \tag{5.2.14}$$

2. Detection Analysis:
   The PDF of the maximum SNR branch out of IID Rayleigh branches is given by: [4]

   $$f_{\gamma max}(\gamma) = \frac{L}{\bar{\gamma}} \left(1 - e^{-\gamma/\bar{\gamma}}\right)^{L-1} e^{-\gamma/\bar{\gamma}}. \tag{5.2.15}$$

   Hence, the averaged P_d of the MSPN can be evaluated as

   $$\bar{P}_{d,MSPN} = L \sum_{i=0}^{L-1} \frac{(-1)^i}{i+1} \begin{pmatrix} L-1 \\ i \end{pmatrix} \bar{P}_{dRay} \left(\frac{\bar{\gamma}}{i+1}\right), \tag{5.2.16}$$

   where $\overline{P}_{dRay}\left(\frac{\bar{\gamma}}{i+1}\right)$ is the averaged detection probability over one branch Rayleigh channel, and given by:

   $$\begin{aligned} \overline{P}_{dRay} = e^{-\frac{\gamma_{thres}}{2}} \sum_{n=0}^{u-2} \frac{1}{n!} \left(\frac{\gamma_{thres}}{1}\right)^2 + \left(\frac{1+\overline{\gamma}/(i+1)}{\overline{\gamma}/(i+1)}\right)^{u-1} \\ \left[e^{\frac{\gamma_{thres}}{2(1+\overline{\gamma}/(i+1))}} - e^{-\gamma_{thres}/2} \sum_{n=0}^{u-2} \frac{1}{n!} \frac{\gamma_{thres}\overline{\gamma}/(i+1)}{2(1+\overline{\gamma}/(i+1))} \right]. \end{aligned} \tag{5.2.17}$$

Chapter 6

Proposed Schemes

6.1 Introduction

At the beginning of this chapter we will discuss and analyze the proposed QUAL scheme. This scheme is based on EGC, since it selects an equal gain variable phase antenna's weights. EGC selects the phase corresponding to the channel phase conjugate, however, proposed QUAL scheme selects the phase randomly. Therefore, QUAL does not require knowledge about CSI. In later sections, we analyze the proposed QUAL scheme. Then we derive both simulation and theoretical threshold to fix P_{FA}, since it varies due to the selection of the maximum branch and the increment of the noise mean.
After that, we discuss the second proposed scheme, Eigen Value Decomposition (EVD). EVD scheme is also called blind MRC, from its name we conclude that it is based on MRC scheme. MRC selects the weights according to the amplitude and phase of the channel, while EVD selects the weights based on the received signal amplitude and phase. Therefore, EVD does not require knowledge about CSI.

6.2 Proposed QUAL System Analysis

In our proposed QUAL spectrum sensing scheme, we consider unknown CSI between PU transmitter and the CU receiver. The channel is multi-path Rayleigh channels of multiple antennas are independent, and are considered as flat Rayleigh fading channels. The AWGN has zero mean and variance σ_n^2 is independent in each antenna branch. The PU signal is assumed to be IID random process with variance of σ_s^2. For modeling simplicity, the proposed scheme is explained by using the example of three antennas with four quantization weights per antenna. Figure 6.2.1 shows the CU detector system model. In this model we use the first antenna's signal as a reference without multiplying it by any weight. As for the other antennas (the second and the third antennas in this case) we multiply each branch by k_i (where i is the antenna number) number of pre-decided weights, in our model $k = k_2 = k_3 = 4$. The weights consist of uniformly random distributed random variables, phase random variable (equal gain), distributed over the angles from 0 to 2π. In our simulation, we select the weights in two different ways. The first way is selecting weights uniformly, as shown in Fig.6.2.2(a). The second way is selecting the weights with a randomly selected phase for both antennas as shown in Fig.6.2.2(b). This comparison is done for comparing the difference by changing the antenna's pattern. However, the results showed that both of them have the same impact.

After weighting the antenna branches, we add the first signal with one of the four weighted signals from the second antenna branch, plus one of the weighted signals from the third antenna branch. Repeat this process for all the weights in both the second and the third antennas. As a result, this operation produces $k_2 * k_3$ output signals. The output signals derived by this model can be represented by the following equation

$$\begin{aligned} \mathbf{y'}_m &= \mathbf{y}_1 + \mathbf{y}_2.\mathbf{w}_{1i} + \mathbf{y}_3.\mathbf{w}_{2j} \\ &\quad \ldots i, j = 1, \ldots, 4 \qquad , \\ &\quad m = i + (j-1).k \end{aligned} \tag{6.2.1}$$

here, $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$ are the received signals from antenna $1, 2, 3$ respectively. $\mathbf{w_{1i}}, \mathbf{w_{2j}}$ are the weights for antenna $2, 3$ respectively. And m is the sequence number of the signals results from this process. Then, we apply equation (6.2.1) to the energy detector to get 16 test statistics. After that, we compare the output test statistics $\sum_{n=1}^{N} |y'_{n,m}|^2$ and choose the maximum as in equation (6.2.3). In other word, compare the Signal-plus-Noise value and choose the largest one among them as shown in the following equations:

$$\sigma^2_{\mathbf{y'_m}} = \sigma^2_{s_1+s'_2+s'_3} + \sigma^2_{n_1+n_2+n_3} \tag{6.2.2}$$

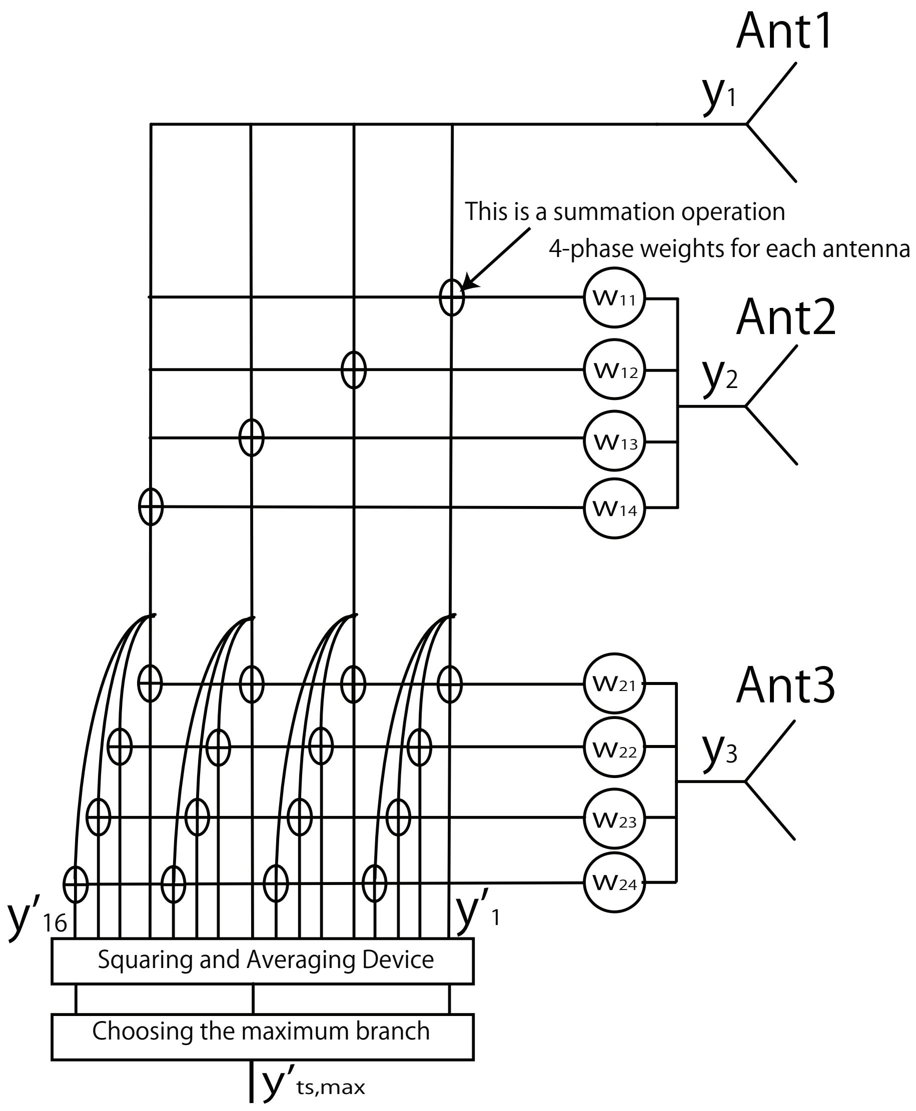

Ant1
y1
This is a summation operation
4-phase weights for each antenna
W11
W12
W13
W14
Ant2
y2
W21
W22
W23
W24
Ant3
y3
y'16
y'1
Squaring and Averaging Device
Choosing the maximum branch
y'ts,max

Figure 6.2.1: Receiver system model for Proposed QUAL scheme.

s′ is the weighted signal. After the selection process, the output signal $y'_{ts,max}$ is be given by:

$$y'_{ts,max} = max(\sum_{n=1}^{N} |y'_{n,m}|^2) \tag{6.2.3}$$

where $y'_{n,m}$ is the complex received signal after weighting $\mathbf{y'}_m$. Then we apply the maximum test statistic to the decision making step and compare it with the threshold to decide the existence of the PU.

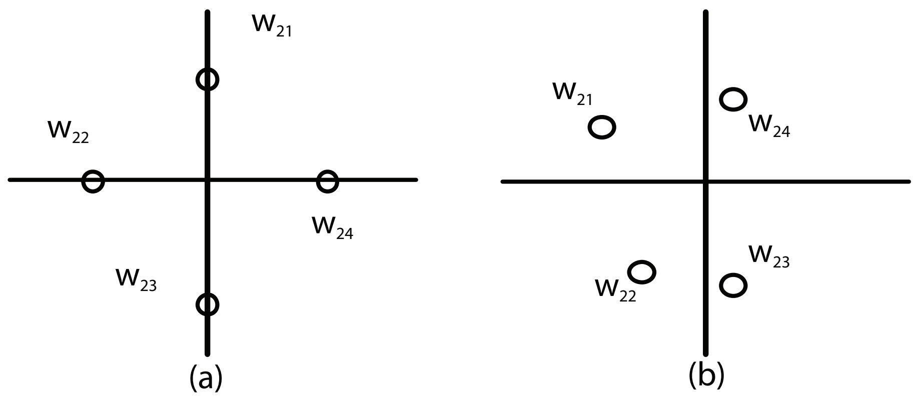


Figure 6.2.2: Distribution of QUAL scheme's Weights.

6.3 Threshold derivation of Proposed QUAL System

As we previously explained, our proposed system chooses the maximum output among many test statistics. This is different from other combining schemes, which results in a variation in the variance and mean of the output. Consequently, if we use the same threshold as we used in the conventional combining schemes we cannot obtain a fixed P_{FA}. This means that we cannot apply our system in a real situation, because we will have large range of error fluctuation in the false alarm probability. Furthermore, deriving a wrong threshold might fix the P_{FA} but results in a degraded P_D performance, which we took care of in our both threshold derivation methods.
To solve the previously mentioned problems, we derived a new threshold by two ways:

6.3.1 Simulation threshold

In which we adjust the old mean and variance of the original threshold by the mean and variance of the new maximum selected test statistics. Adjusting equation (4.3.6) as shown in the following equation,

$$\gamma_{thres,FA} = \mu_{max} + \frac{\sigma^2_{max}\text{erfc}^{-1}(2P_{FA})}{\sqrt{N}}, \tag{6.3.1}$$

where μ_{max} is the mean of the maximum branch, and σ^2_{max} is the variance of the maximum branch.

6.3.2 Theoretical threshold

As we have explained in Sect.3.6, if a large number of independent random variables are generated from the same probability distribution, then we take their maximum values, the resulting distribution of the maximized values is approximated as a Generalized Extreme Value (GEV) distribution. We have the same case in QUAL proposed scheme. Before choosing the maximum value among the 16 test statistics, the distribution is independent normal distribution. Therefore the maximum output follows the distribution of GEV Type 1. The CDF of GUMBEL distribution is given by equation(3.6.1), from equation (4.1.3) we get the probability of False alarm as

$$P_{FA} = 1 - CDF_{GEV} = 1 - \exp(-e^{(-(x-\mu)/\sigma)}). \tag{6.3.2}$$

Solving equation (6.3.2), we get the following threshold:

$$\gamma_{thres,gumb} = -\sigma \log(-\log(1 - P_{FA})) + u \tag{6.3.3}$$

After substituting μ and σ by the original mean and variance values we have the final equation as:

$$\gamma_{thres,gumb} = \frac{-\sigma_{noise}^2}{\sqrt{N}} \log(-\log(1 - P_{FA})) + \sigma_{noise}^2 \tag{6.3.4}$$

We will see in Sect.7 that this threshold fixes the false alarm probability, comparing to the original threshold, while having the improved detection probability.

6.4 Proposed EVD to Achieve Blind MRC Detection

As discussed in some literatures [11], the wireless environment consists of directional sources and white Gaussian noise, the Eigen values of the received signal correlation matrix $\mathbf{R_y}$ can be divided into two sets. Since the eigenvectors of $\mathbf{R_y}$ are orthogonal to each others, it can be considered as L-dimension space. This L-dimensional space can be linked to the directional signal sources and the noise. It can be divided into two subspaces as below.
The largest M eigenvalues are associated with the M directional sources, and the corresponding eigenvectors called signal eigenvectors; these signal eigenvectors are corresponding to first subspace called signal subspace.
The $L - M$ smallest eigenvalues are associated with noise, and the corresponding eigenvectors called noise eigenvectors; these noise eigenvectors are corresponding to the second subspace called noise subspace.

The Eigen structure-based method searches for signal directions, in which the steering vectors associated with these directions are orthogonal to the noise subspace and contained the signal subspace. The source directions correspond to the local minimal (maximal) of the function $|w^H s_{th}|$. In this function s_{th} denotes a steering vector and w^H are the weights. When these steering vectors are not guaranteed to be corresponding to the signal subspace, due to some large interference or large noise variance, there may be more minimum (maximum) values than the number of sources. This method can be also called Eigen-Beamspace beam former [16]. It can be used for anticipating the number of incident plane waves.

After this discussion, we present a way for using the eigenvalue decomposition theory to achieve the MRC optimal performance without the necessity to estimate the channel. If we use the eigenvector, corresponding to the largest eigenvalue of the received signal's covariance matrix, as a weight vector for the array antenna, the same performance of MRC in high SNR region can be achieved. Therefore we consider this performance as the best blind detection scheme which does not need knowledge of CSI. Starting from equation (5.2.2), we can calculate the covariance matrix of the received signal as:

$$\mathbf{R_y} = \mathbf{YY^H}, \tag{6.4.1}$$

where $\mathbf{R_y}$ is the covariance matrix of the received signal Y. The received matrix Y is expressed as:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y_1} & \mathbf{y_2} \dots & \mathbf{y_L} \end{bmatrix}^{\mathbf{T}}. \tag{6.4.2}$$

In $\mathbf{R_y}$, each row contains the covariance values of the corresponding branch with other branches according to the column number,

$$R_y \mathbf{V} = \mathbf{V\Lambda}, \tag{6.4.3}$$

where

$$\Lambda = \begin{bmatrix} v_1 & \cdot & \cdot & \cdot \\ \cdot & v_2 & \cdot & \cdot \\ \cdot & \cdot & \ddots & \cdot \\ \cdot & \cdot & \cdot & v_L \end{bmatrix}, \mathbf{V} = \begin{bmatrix} \mathbf{v_1} & \dots & \mathbf{v_L} \end{bmatrix}. \tag{6.4.4}$$

In equation (6.4.4), $v_1 \geq v_2 \geq \ldots \geq v_L$ are the covariance matrix's eigenvalues, starting from the largest v_1 (which is corresponding to the dominant channel. At which the SNR is maximum and we can achieve better communication capacity and interference avoidance). $\mathbf{v_1}, \ldots, \mathbf{v_L}$ are the corresponding eigenvectors. To realize the maximum SNR and to achieve the performance of MRC, we use the eigenvector (corresponding to the largest eigenvalue) as a weight $\mathbf{w}$ for the array antenna.

$$\mathbf{w} = [\mathbf{v_1}]^{\mathbf{T}}, \tag{6.4.5}$$

where the output of the combiner will be expressed as:

$$\mathbf{y_{out}} = \mathbf{w^H y} = \mathbf{w^H H x} + \mathbf{w^H n}. \tag{6.4.6}$$

When evaluating the detection performance for EVD, it is compulsory to normalize the test statistics values with the proper factor, because it will effect the detection performance.

6.5 System Characteristics

In this section, we explain the main characteristics of the proposed schemes. As we mentioned in Sect. 4.3, to improve the detection performance, we have to decrease the probability of false alarm or probability of miss-detection (which mean to increase probability of detection). As in Fig4.3.2 the detection performance can be improved by increasing the mean of the *S+N* distribution. In other word, by increasing the distance d_2 more than the distance d_1, the probability of error detection decrease. We can make use this increment by two ways, either by fixing the threshold which automatically causes the increasing of P_D, or, by changing the threshold to fix the P_D and decrease the P_{FA}. The results which shows the improvement of the test statistics's PDF means for the proposed QUAL and EGC schemes are shown in both Fig.7.0.1 and table 7.0.2 in Sect.7

6.5.1 Discussion of the Conventional Schemes

The improvement of the conventional schemes are due to two factors. One is diversity gain, and the other is the coherency in combining, both phase and amplitude coherency. In Sect.7 we will see that the order of system improvement is (from best performance) MRC, EGC, EVD/QUAL, MSPN. One common improvement gain for all schemes is the diversity gain which achieved due to using multiple antennas experiencing IID fading channels. The MRC's improvement is due to the amplitude, phase coherence combining. On the other hand, EGC has only phase coherence, that is why it has lower performance than MRC. As for the MSPN (select the maximum signal plus noise variance branch) the improvement is due to only the diversity principle (no combining scheme), therefore it achieve the worse performance. Later on in this section we will see the difference between only selecting the maximum signal plus noise (MSPN) branch directly, comparing to, selecting the maximum branch after applying the proposed QUAL scheme (which has combining scheme).

6.5.2 Difference between Power Gain and Diversity Gain in Detection Performance

An important concept to be explained is the difference between the space diversity gain and the power gain (PG) concepts. By having multiple antennas at the receiver side we have two main sources of gain. One is diversity gain and the other is power gain. The diversity gain results from the diversity principle. The power gain results from increasing the observation time for one antenna to have an overall samples as many as the multi-antennas case. This cause the total SNR to be increased linearly by a factor of L, where L is the number of antennas or the factor by which we increased the observation time. Considering the inner term of the error probability for BPSK, we can divide it to show these two gains, as follows:[12]

$$\|h\|^2 SNR = \underbrace{LSNR}_{P.G} \cdot \underbrace{\frac{1}{L}\|h\|^2}_{D.G}, \tag{6.5.1}$$

where the first term corresponds to the power gain $(P.G)$ and the second term shows the diversity gain $(D.G)$. In the high SNR region the diversity gain can be shown as an exponent of $(1/SNR)^L$, while the power gain expressed as the previous linear term,

$$\underbrace{\begin{pmatrix} 2L-1 \\ L \end{pmatrix}}_{P.G} \underbrace{\frac{1}{(4SNR)^L}}_{D.G}. \tag{6.5.2}$$

In Sect.7, Fig. 7.0.8 shows a comparison between achieving the power gain and achieving the diversity gain for all schemes which have been explained in this paper.

6.5.3 Discussion of the Proposed QUAL System and the Correlation Coefficient

For the proposed QUAL system, the mean of *S+N* depends on the number of weights. Increasing the number of candidate weights will increase the mean of the output signals. However, in this case d_2 (as in Fig.4.3.2) does not depends only on *S+N* mean, because the noise mean is also changing (increasing the number of weights will also increase the noise mean). Therefore the system improvement depends on the difference between both means of *S+N* and noise of the proposed QUAL system. We can discuss the origin of increasing the *S+N* mean in the proposed QUAL scheme by two ways.

- ***The first way*** is by using the vector chart (graph) as in Fig.6.5.1, in which we consider only two antennas for simplicity. The first antenna branch is marked by $"Ant.\#1"$, while the second antenna's four weighted branches marked by $"'Ant.\#2, w_i"'$, where $'i'$ is the weight's number. When we combine the two branches the best choice is to select the closest $"Ant.2"$ vector to $"Ant.1"$ vector, which, in our case is $"Ant.\#2, w3"$. In this case the output of the combiner will be the largest comparing to the others vector;

this is what we called phase coherence combining. And the worst choice will be to combine with "$Ant.\#2, w1$". If "$Ant.\#2, w3$" is exactly aligned to "$Ant.\#1$" vector, it will be the EGC weight case. The proposed scheme drawback, is that this concept does not apply only for the signal's vectors but also for the noise vector. Therefore, sometime we might improve the noise mean by choosing the maximum output of the combiner.

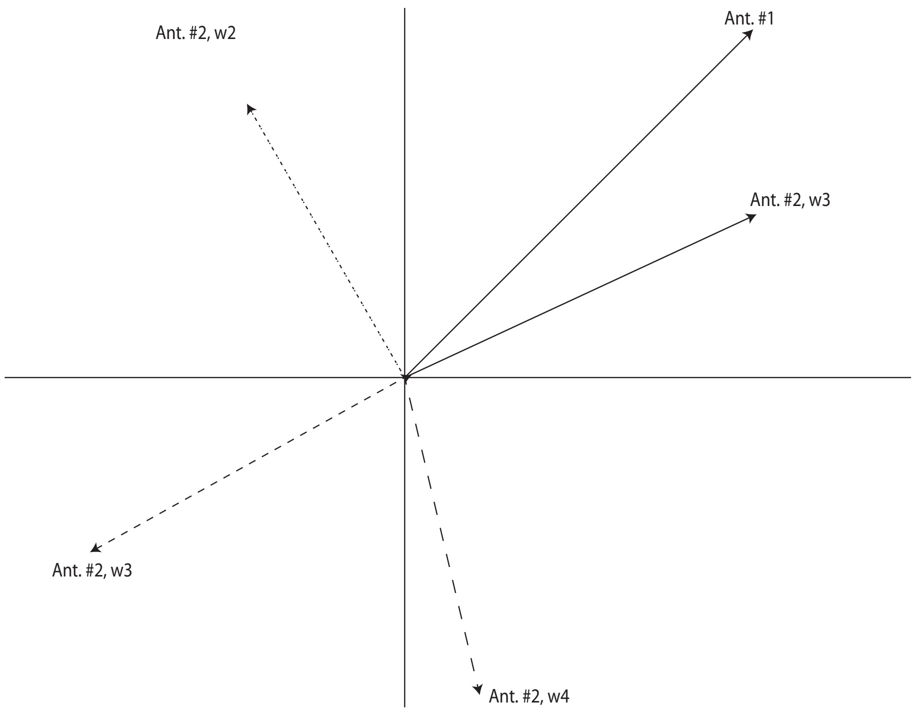


Figure 6.5.1: Reference and Quantized Weighted vectors for the QUAL proposed scheme.

- ***The second way*** is by checking the complex correlation coefficients. As we know, the output after combining two IID signals will have variance given by

$$Var(Z) = \sigma_x^2 + \sigma_y^2. \tag{6.5.3}$$

where $Z = X + Y$. However, if there is a correlation between X and Y, the value of the output variance will change (improves for positive correlation and degrades for negative correlation) as can be derived easily in the following:

$$Var(Z) = \sigma_x^2 + \sigma_y^2 + 2\rho_{x,y}.\sqrt{\sigma_x^2\sigma_y^2}, \tag{6.5.4}$$

where, $\rho_{x,y}$ is the complex correlation coefficient between X and Y. Improving of the combiner output's variance leads to improvement in the mean of the test statistic *u(t)*. Checking from Table 6.1 the value "$\rho_{ref,max} = 0.361$" is the correlation coefficient between the reference branch and the maximum branch, which is positive and is not equivalent to zero as the others. The notation is considered as ($\rho_{ref,1}$ correspond to w_1 branch, $\rho_{ref,2}$ correspond to w_2 branch, $\rho_{ref,3}$ correspond to w_3 branch, $\rho_{ref,4}$ correspond to w_4 branch).

Table 6.1: Correlation Coefficient matrix of proposed system branches.

Ref. branch	**Max.** branch $(\rho_{ref,max})$	**W1** branch $(\rho_{ref,1})$	**W2** branch $(\rho_{ref,2})$	**W3** branch $(\rho_{ref,3})$	**W4** branch $(\rho_{ref,4})$
1	0.361	-0.010	-0.000	-0.009	-0.006

This partial phase coherency in combining of multiple branches is the source of improving the detection performance of the proposed QUAL system comparing to the MSPN system. Where, on the later one we achieve only diversity, however, on the formal one, we achieve diversity gain plus partial combining coherency. But, its not full coherency, therefore, the QUAL system detection improvement is less than the EGC scheme.

Another crucial point to be cleared is the different between multiplying the antenna branches by a QUAL weights and multiplying the antenna branches by an equal gain weights equivalent to the Signal + Noise phase. The later method is equivalent to taking absolute of the received signal, which means that we add the noise's phase every time we multiply by weights or take absolute. This method results in a destructive combining among multiple antennas. Therefore, the later method does not achieve neither diversity nor combining gains, it achieves only power gain. On the other hand, the formal method (proposed QUAL) achieves diversity and (constructive addition) combining gains as in Sect.7.

Chapter 7

Numerical Results

In this section we quantify the receiver performance by three evaluation methods using the parameters described in Table 7.0.1. One is using the complementary receiver operating characteristic (ROC) curves P_{MD} versus P_{FA}. Another evaluation method is P_{FA} V.S SNR, to show the robustness of the new derived threshold. The last one is by depicting P_{MD} V.S SNR, to show the improvement of the detection performance by increasing the SNR. It is important to mention that one crucial applications for blind sensing in a very low SNR (around -100 dBm) is the DTV band using 802.22 WRAN [17].

Table 7.0.1: SIMULATION PARAMETERS.

Parameter Name	**Value**
# used antennas	1,2,3
Fixed weights for $2^{nd}-3^{rd}$	4-4
Averaging bits	128, 256
Fixed Primary SNR	-10 dB,-5 dB
# sent packets/one probability value	5000, 50000

Table 7.0.2 and Fig.7.0.1 shows the normalized values of the output test statistics distributions means, after/before applying both the EGC scheme and the proposed QUAL scheme. The normalization factor follows the Zero-Mean normalization,

$$u' = (u - \mu_{noise})\,/\sigma_{noise}. \tag{7.0.1}$$

Considering the noise only (H_0 case), before and after applying it to the EGC scheme, it's mean does not have large changes, close to zero. However, checking the shifted value of the S+N mean before and after applying it to the EGC combiner, it is shifted from *"1.1216"* to *"2.0273"*. As for the proposed system, the

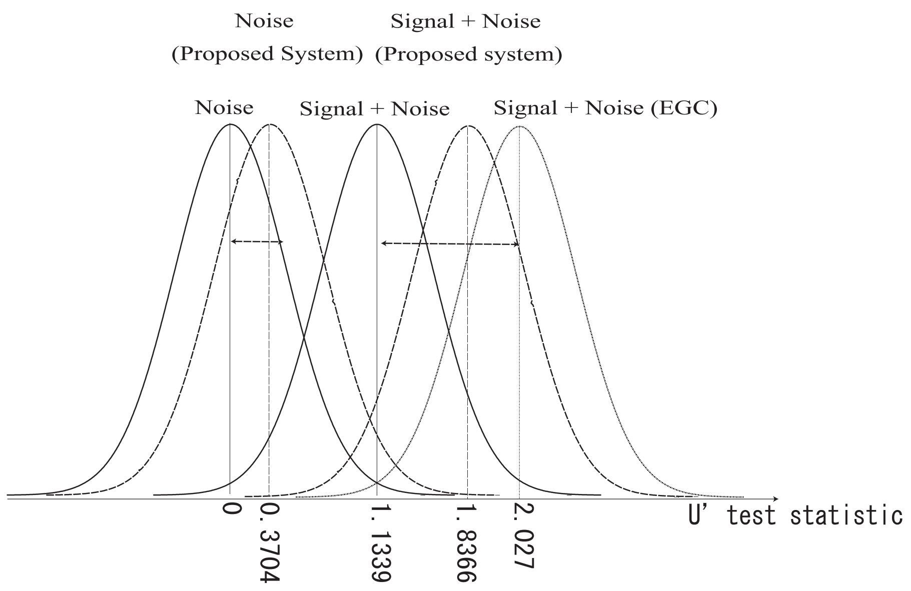


Figure 7.0.1: Gaussian Shifts according to the proposed QUAL scheme.

mean of H_0 case, after applying the proposed scheme, has increased to *"0.3704"*. However the increment when applying the proposed scheme to S+N is larger than that for H_0, which reaches *"1.8366"*, this clarify the reason of the system improvement. Even though the values of the means for S+N of the EGC and the proposed scheme are almost the same but the shift in the noise mean for both cases is not the same, which give the advantage to EGC comparing to the proposed scheme.
After this discussion, we can say that if we combined the multiple branches without any weight, we can be sure that there will be no improvement, because the combining is totally destructive (no phase coherence). Then, we confirm that using the QUAL weights results in a partially coherence combining, which cause the shifting the mean of S+N.

Table 7.0.2: Shifted means of the proposed QUAL scheme & EGC.

Detection case / before, after applying the system	EGC	Proposed System
Noise / Before	0	0
Noise / After	0.0395	0.3704
Signal + Noise / Before	1.1216	1.1339
Signal + Noise / After	2.0273	1.8366

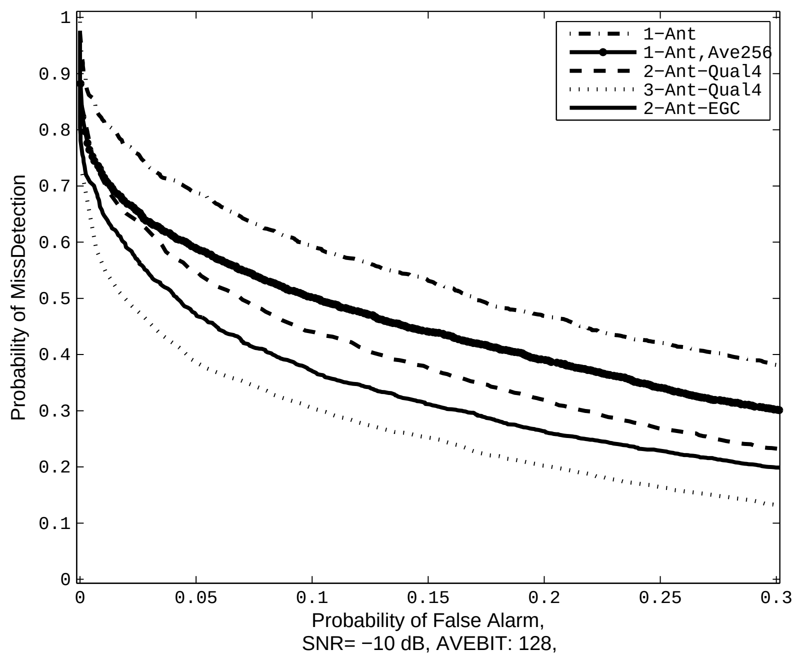


Figure 7.0.2: Detection performance of combining signals using the proposed QUAL scheme.

Figure 7.0.2 shows the sensing performance of the proposed QUAL scheme. Here, we use 1, 2, 3-multiple antennas with 4-quantization weights. From this figure, by using the proposed QUAL scheme, the performance of the primary signal detection is improved compared to that of the single antenna reception case. In the same figure we plot the performance of EGC scheme with perfect channel estimation, which can be considered as a lower bound (optimal performance) for the QUAL scheme. We can see that for $P_{FA} = 0.1$ the improvement in the P_{MD} for power gain is about *(0.5024-0.5918)* 0.09. As for the 2-antenna and 3-antenna case the improvement is about 0.1520 and 0.2892 respectively. On the other hand, for EGC 2-antennas case, the improvement in P_{MD} performance is about 0.22118, which is larger than the similar QUAL case and lower than QUAL's 3-Antennas case.

Figure 7.0.3 evaluates the sensing detection performance by changing the SNR. From this figure, we confirm that the performance of P_{MD} is improved by increasing SNR values from -20 to 0 dB when the false alarm probability is fixed as $P_{FA} = 0.1$, for different numbers of antennas. It is possible to say that the improvement in the detection performance caused by enhancing the SNR using QUAL scheme. For SNR more than -13 dB we confirm the noticeable improve-

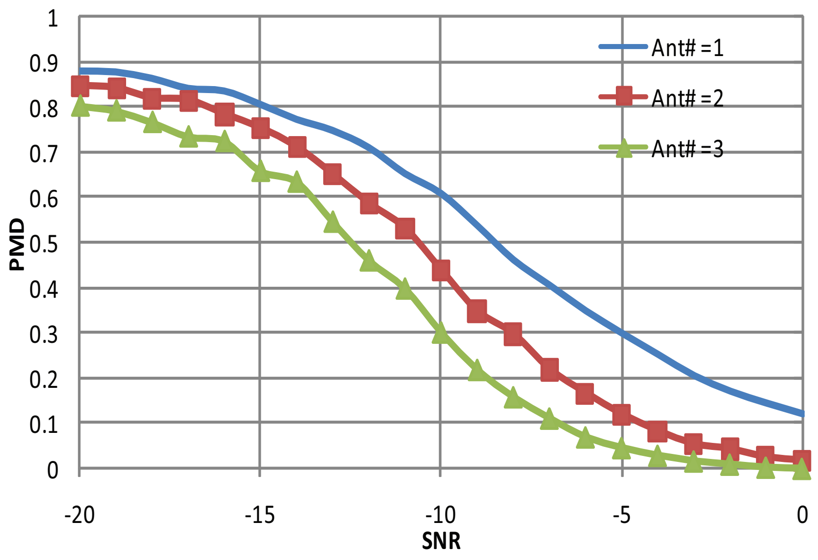


Figure 7.0.3: Performance of P_{MD} of QUAL scheme by changing SNR.

ment of the proposed method (about 2 dB for 2-antennas case and 4 dB for 3-antennas case). However for very low SNR ($SNR = -20$ dB) the performance of the QUAL system does not have a noticeable improvement.

In Fig. 7.0.4 we compare the probability of false alarm between two methods, using the adaptively derived GUMBEL threshold and using normal ED threshold both with 4-QUAL weights . Using GUMBEL threshold we obtain almost fixed P_{FA} around 0.1. We notice that the P_{FA} achieved by the normal threshold is shifted to around 0.16. Furthermore, the varying range of P_{FA} is from *0.12 to 0.19*. On the other hand, using GUMBEL threshold the P_{FA} is fixed around *0.1*, while the varying range is from *0.08 to 0.12*. This shows the accuracy of our derived threshold.

Figure 7.0.5 shows the Bit Error Rate performance for different diversity schemes. These schemes are Maximum Ratio Combining (MRC) and Eigen Value Decomposition combining (EVD), and they are considered for 1, 2, 3, 4-antennas respectively, where the first line is for 1-antenna case. The main point here is not to discuss the MRC BER performance, because it has been discussed deeply in many literatures. The point is to compare the EVD performance to the MRC performance. As we know, because the MRC has phase and amplitude coherence, it achieves the optimal performance among all diversity combining schemes. However, MRC needs CSI to detect the signal. In Fig.7.0.5, EVD achieves exactly the same performance as MRC in moderate and high SNR region, without any necessity to know about CSI. From this result, we confirm that the EVD combining is

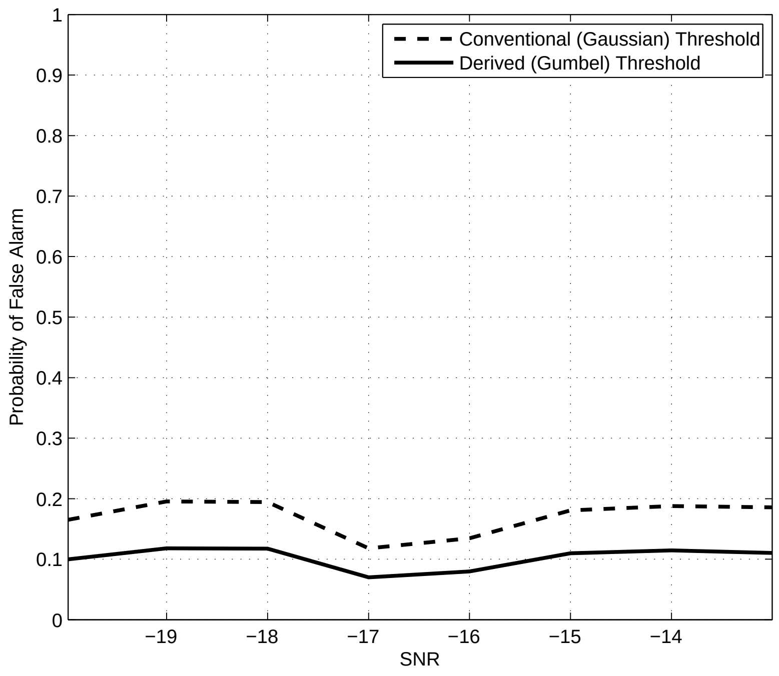


Figure 7.0.4: Performance of False alarm using the Gumbel and the conventional threshold versus SNR.

the best blind combining scheme in moderate SNR. In spite of achieving exactly the same performance of MRC in $SNR > 0$, EVD scheme performance degrades in low SNR region.

Figure 7.0.6 evaluates the performance of P_{MD} versus SNR using 2-3-4 antennas of EVD case. We confirm the improvement achieved by using more antennas from 2 to 4. Furthermore, it is clear that if the SNR increases, the P_{MD} decreases. So, we achieve around 0.1 P_{MD} for -5, -7, -8.5 dB by using 2, 3, 4-antennas respectively. Therefore, we confirm the diversity gain (D.G) improvement 2, 1.5 dB between 2-3, 3-4 antennas cases. Where D.G decreases by increasing the number of antennas (largest case is for 1-2 antenna). It is clear that around -8 dB the improvement of 2-antennas is larger that -15 dB.

Figure 7.0.7 evaluates the probability of miss-detection performance for four antennas using EVD, MRC, EGC and MSPN (selecting Maximum Signal Plus Noise branch) by changing the SNR. From this figure, we can confirm that the performance of P_{MD} is improved by increasing SNR values from -20 to 0 dB when the false alarm probability is fixed as $P_{FA} = 0.1$, for all the schemes. As it is shown, EVD scheme get closer to MRC with increasing the SNR. It agrees with what we

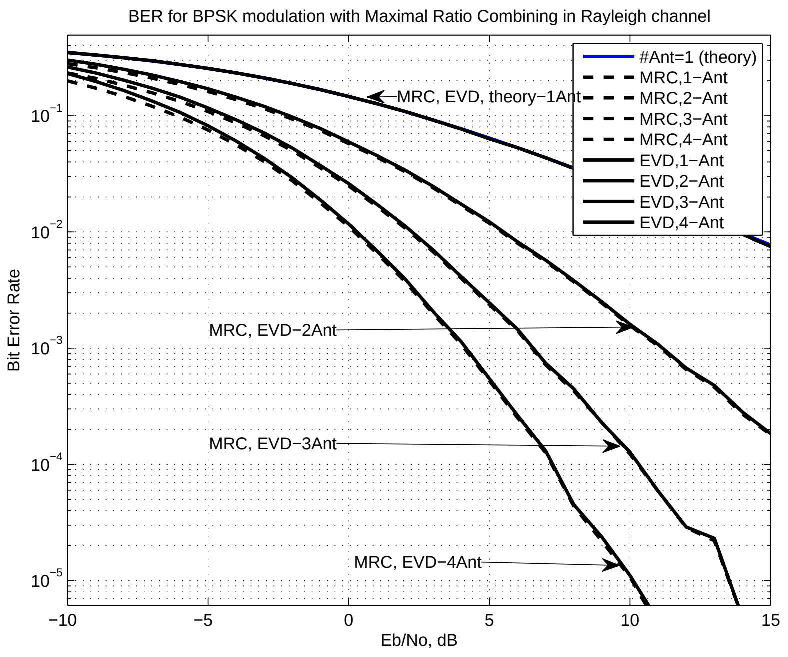


Figure 7.0.5: Bit Error Rate for MRC and EVD versus SNR.

have seen in Fig. 7.0.5, which shows that EVD achieve the same performance as MRC in high SNR region.

Figure 7.0.8, is the last result, and it compares the detection performance using the CROC (Complementary Receiver Operating Characteristics) among all implemented diversity schemes, in low SNR region. In this figure we show the diversity gain improvement of each combining scheme and compare it to the power gain case. As mentioned before, the power gain can be realized by considering averaging time equal to double the one in 2-antennas case (which tagged in the figure by *"1-Ant, Ave256"*). Considering the diversity gain improvement, in ascending order, the schemes can be ordered as follows, MSPN, EVD, QUAL, EGC and MRC. As expected, the optimal scheme is MRC, followed by EGC. Then the proposed QUAL and EVD schemes have almost the same detection performance without CSI. This is important results show that these two schemes have the optimal blind detection performance under low SNR region. They have almost the same diversity gain. Even though the proposed schemes have a degraded performance compared to MRC and EGC, however the proposed schemes do not required any knowledge about CSI. This makes the QUAL and EVD scheme more likely to be implemented when we start sensing the spectrum.

To calculate the weight of EVD, we need to calculate the received signal's covari-

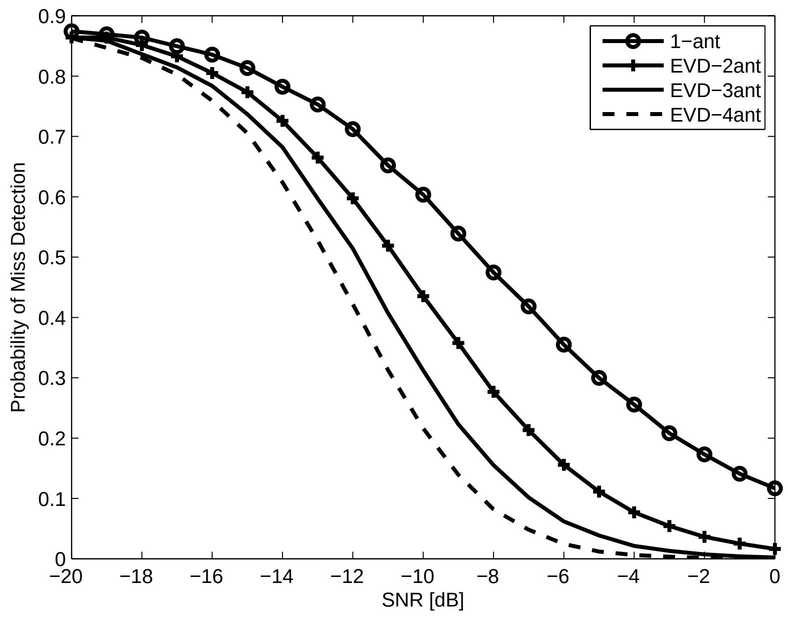


Figure 7.0.6: Performance of P_{MD} of EVD by changing SNR, and changing the number of antennas.

ance matrix, and then extract the eigen vector which is corresponding to the maximum eigen value from that covariance matrix. Then we multiply this Eigen vector by the received signal to realize EVD combining. This required a large amount of computational power and complexity. On the other hand, QUAL scheme is very simple just to multiply the branches by a 4-random phase weights, then combine them and choose the maximum branch will give the same results as EVD.
Finally, the lowest diversity gain achieved by MSPN selection combining. This is because the selection combining schemes does not utilize any coherency in combining signals, because it selects one branch only.

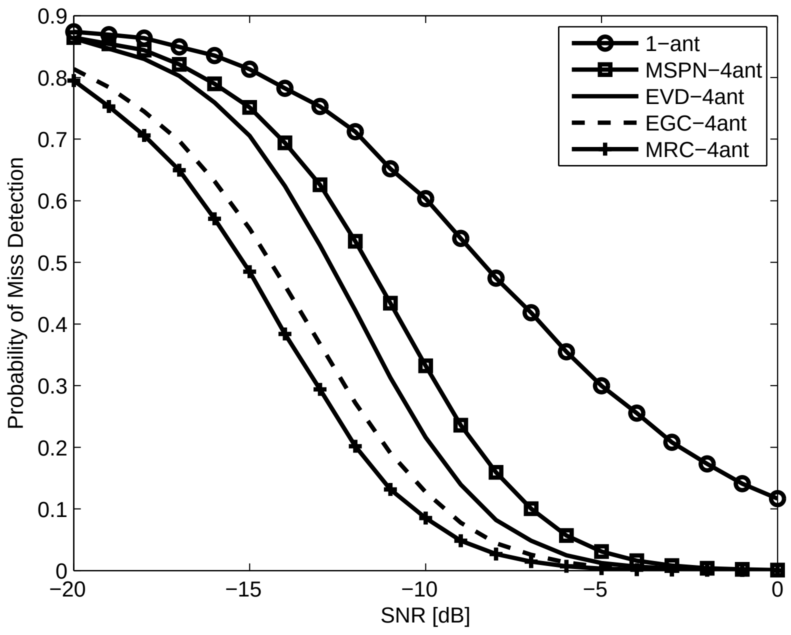


Figure 7.0.7: Performance of P_{MD} of the EVD by changing SNR versus other diversity schemes.

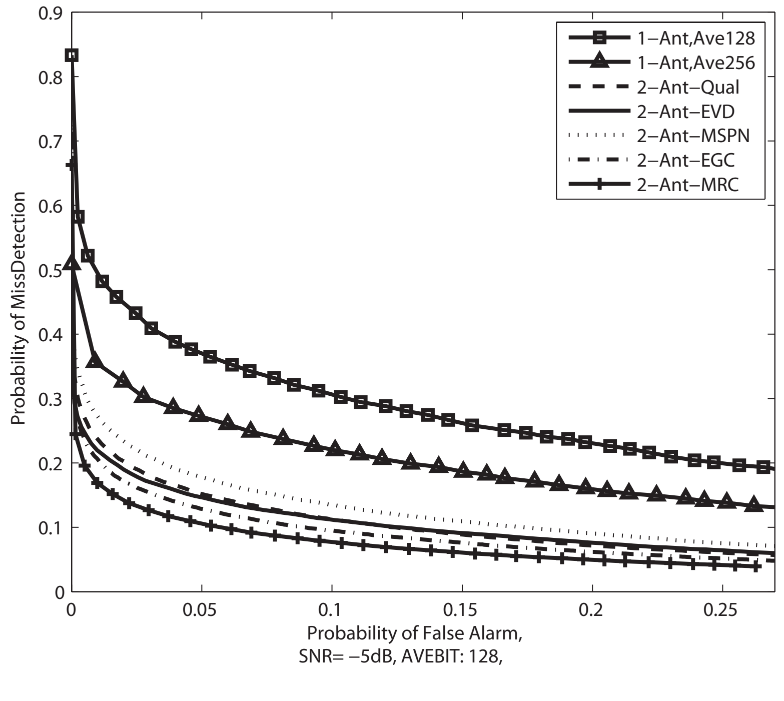


Figure 7.0.8: Performance of P_{MD} of proposed and conventional schemes versus P_{FA} for SNR=-5 dB.

Chapter 8

GNU Radio

8.1 Introduction.

In chapter 2 we talked about SDR systems and how does it play an important role for realizing cognitive radio system. One of the candidates for implementing SDR is GNU Radio implemented using a USRP (Universal Software Radio Peripheral) device. USRP is produced by ETTUS research center [19].
GNU Radio is a free software development toolkit that provides the signal processing runtime and processing blocks to implement software radios using readily-available, low-cost external RF hardware and commodity processors. It is widely used in hobbyist, academic and commercial environments to support wireless communications research as well as to implement real-world radio systems. GNU Radio applications are primarily written using the Python programming language, while the supplied, performance-critical signal processing path is implemented in C++ using processor floating point extensions where available. Thus, the developer is able to implement real-time, high-throughput radio systems in a simple-to-use, rapid-application-development environment. While not primarily a simulation tool, GNU Radio does support development of signal processing algorithms using pre-recorded or generated data, avoiding the need for actual RF hardware.[18]
In this chapter we explain the utilization of GNU Radio as a tool for our spectrum sensing research. We start by a fast overview about the Software part of GNU Radio. Then, we present a Hardware glance, including some necessary parts of USRP1 to keep the reader familiar with it. Finally, in the last sections we present our implemented work using GNU Radio, which is divided into two parts. In appendix A we explain about the GNU Radio's installation procedures.

8.2 GNU Radio - Software.

In this section we give a fast overview about the GNU Radio's software based part. As shown in Fig.8.2.1 GNU Radio software is based on two-tire structure. These tires are Python and C++, we can think of this as two processing layers for different purposes. These two languages are tied together using SWIG interpreter.

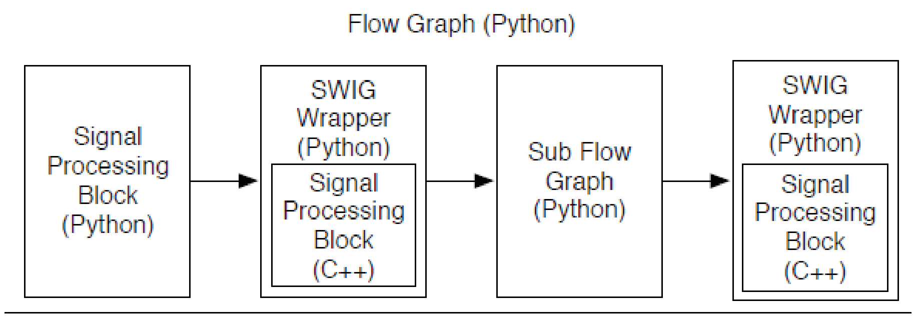

Flow Graph (Python)
Signal Processing Block (Python)
SWIG Wrapper (Python)
Signal Processing Block (C++)
Sub Flow Graph (Python)
SWIG Wrapper (Python)
Signal Processing Block (C++)

Figure 8.2.1: Software Architecture in GNU Radio.

The lower programming layer is to take care of the performance-critical and complex signal processing parts. It is done using C++ because of its high processing capabilities. For example, source, sink which play the role of obtaining and outputting data from and to the real world are in the lower layer. Filtering and coding needs high complexity mathematical operations, therefore it is done in the lower layer.
On the other hand, the higher layer processing part is done using Python, which is an object oriented programming language (OOP). The higher layer is taking services from the lower layer and giving it inputs. At the same time, it organizes, directs and glues the different signal processing block from the lower layer. Debugging is easily done through the higher layer. Interfacing with hardware (USRP) is done through the lower layer. An interesting capability of GNU Radio is using python we can glue different and many C++ signal processing blocks to create the needed system, e.g receiver, transmitter, walkie-talkie, modulator, radar etc.
SWIG is an interface compiler that connects programs written in C and C++ with scripting languages such as Perl, Python, and Ruby. It works by taking the declarations found in C/C++ header files and using them to generate the wrapper code that scripting languages need to access the underlying C/C++ code. In addition, SWIG provides a variety of customization features that let you tailor the wrapping process to suit your application.
In the next section, we explain a hardware background for GNU Radio. With this

background the reader should have enough information to understand our discussion of the implemented systems in later sections.

8.3 GNU Radio - Hardware (USRP).

The Universal Software Radio Peripheral (USRP) is designed to allow general purpose computers to function as high bandwidth software radios. In essence, it serves as a digital baseband and IF section of a radio communication system. In This paper we are going to discuss the important parts of USRP. Figure 8.3.1 shows a photo of a real USRP1, and 8.3.2 shows a simple schematic diagram of the USRP motherboard.

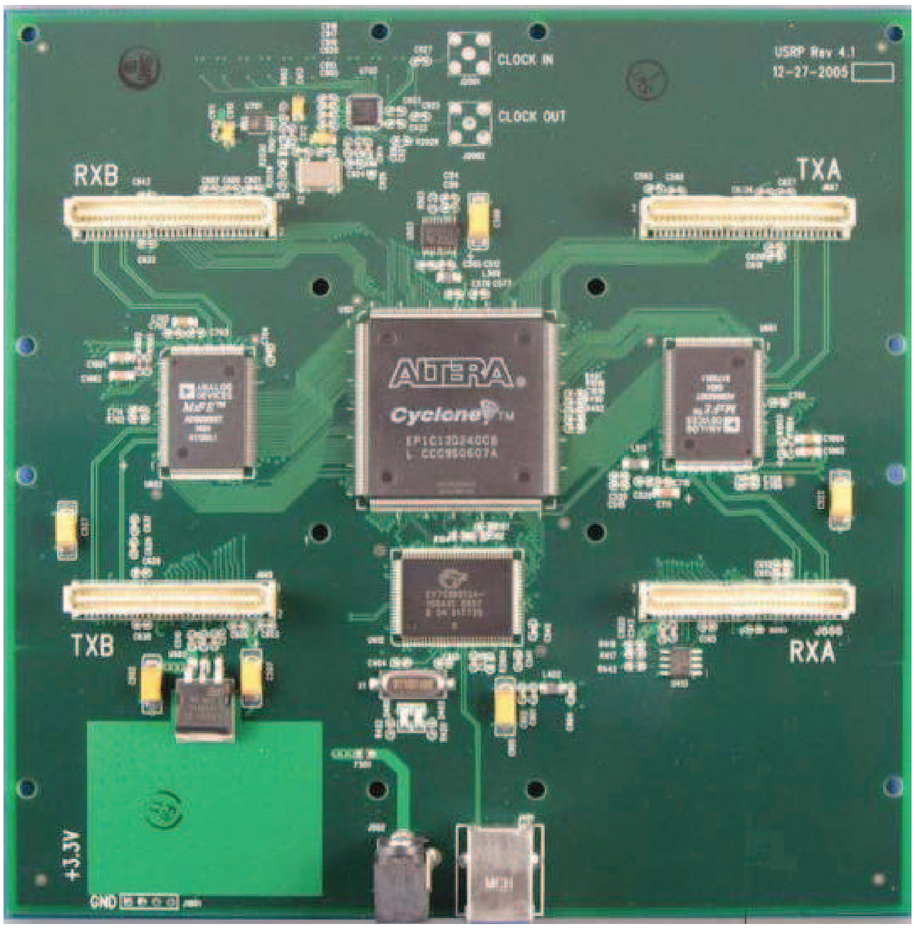


Figure 8.3.1: USRP Real Photo.

As shown in Fig.8.3.2, the USRP Motherboard is connected to a PC through a ’FX2 Micro-controller (USB Interface)’. The high speed math part is done in the FPGA part of the motherboard. The FPGA is connected to a 4-DA/AD high speed converter. ADC/DAC are connected to 2 daughter boards in side A, and two daughter boards in side B. Before discussing about the motherboard’s parts,

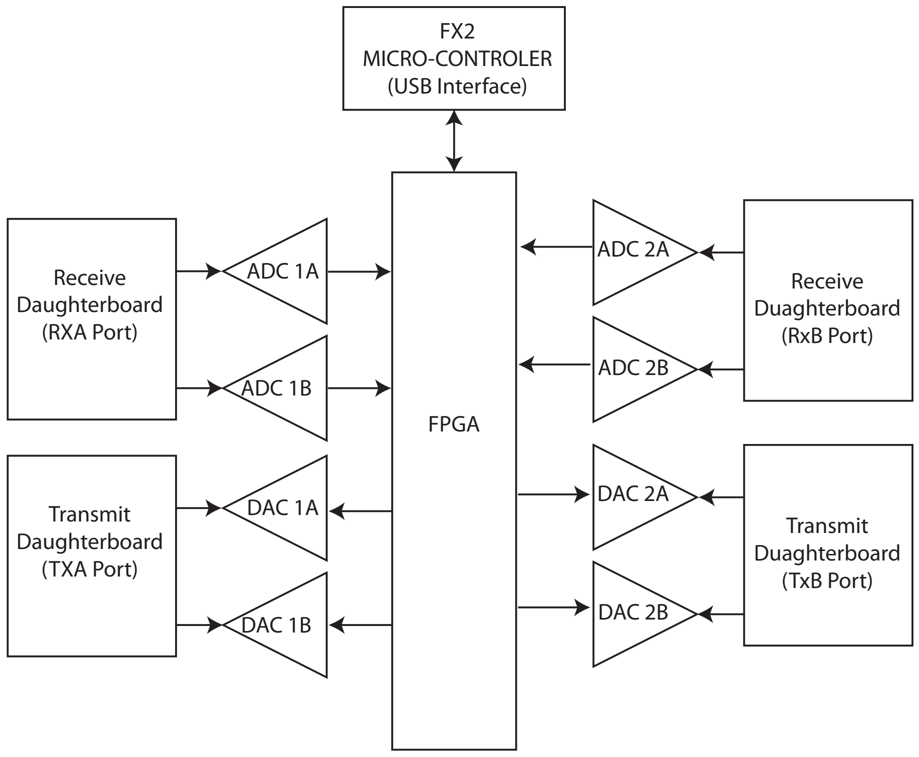

FX2
MICRO-CONTROLER
(USB Interface)
FPGA
Receive Daughterboard (RXA Port)
Transmit Daughterboard (TXA Port)
Receive Duaghterboard (RxB Port)
Transmit Duaghterboard (TxB Port)
ADC 1A
ADC 1B
DAC 1A
DAC 1B
ADC 2A
ADC 2B
DAC 2A
DAC 2B

Figure 8.3.2: A schematic diagram of the USRP motherboard.

we would like to draw your attention to the fact that there are three configuration modes for FPGA, as shown in the Table8.3.1. Now lets start our journey in the motherboard parts.

Table 8.3.1: FPGA Operating Modes.

rbf file name	**FPGA description**
$multi\_2rxhb\_2tx.rbf$	there are two or more USRPs
$std\_2rxhb\_2tx.rbf$	contains 2 Rx paths with halfband filters and 2 tx paths
$std\_4rx\_0tx.rbf$	contains 4 Rx paths without halfbands and 0 tx paths

- Digital Down Converter.
  In the receiving path the standard FPGA configuration includes digital down converters (DDC) implemented with 4 stages cascaded integrator-comb (CIC) filters. CIC filters are very high-performance filters using only adds and delays. For spectral shaping and out of band signals rejection, there is also 31 tap half-band filters cascaded with the CIC filters to form complete DDC stage. The standard FPGA configuration implements 2 complete DDC. Also

there is an image (configuration mode) with 4 DDCs but without half-band filters. This allows 1, 2 or 4 separate RX channels. A schematic diagram of the USRP's standard DDC is shown in Fig.8.3.3.

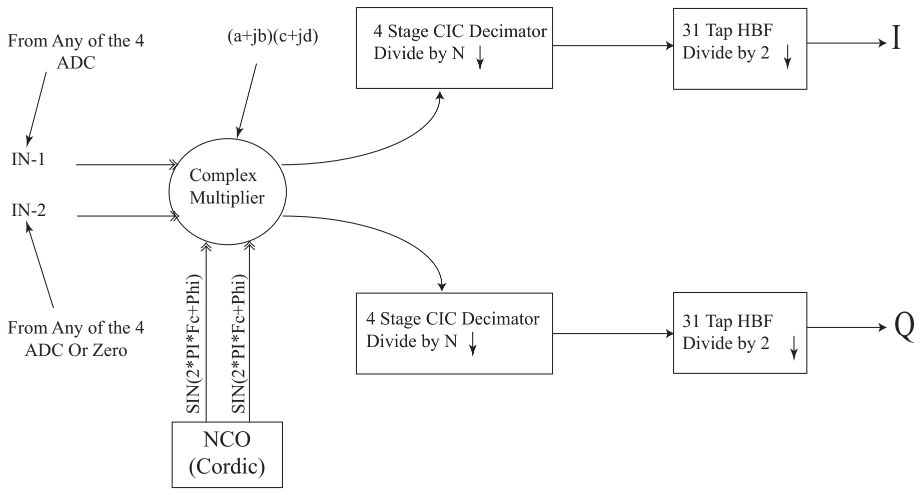


Figure 8.3.3: A schematic diagram of the USRP DDC.

Now let's see what does the DDC do. First, it down converts the signal from the IF band to the base band. Second, it decimates the signal so that the data rate can be adapted by the USB 2.0 and is reasonable for the computers' computing capability. The complex input signal (IF) is multiplied by the constant frequency (usually also the IF) exponential signal. The resulting signal is also complex and centered at 0. Then we decimate the signal with a factor N. This decimation stage can be thought of as a low pass filter followed by a down sampler. The decimation rate must be in between [8, 256]. Finally the complex I/Q signal enters the computer via the USB. Note that when there are multiple channels (up to 4), the channels are interleaved.

- Digital Up Converter.
  Typically, what happened in the Rx path happens in the Tx path, with some reversing changes. We need to send a baseband I/Q complex signal to the USRP board. The digital up converter (DUC) will interpolate the signal, up convert it to the IF band and finally send it through the DAC.
  The digital up converters (DUC) on the transmitter side are actually contained in the AD9862 CODEC chips, not in the FPGA (as shown in the Fig.8.3.4). The only transmit signal processing blocks in the FPGA are the CIC interpolators.
  The interpolator outputs can be routed to any of the 4 CODEC inputs. In

multiple TX channels (1 or 2) all output channels must be the same data rate (i.e. same interpolation ratio).

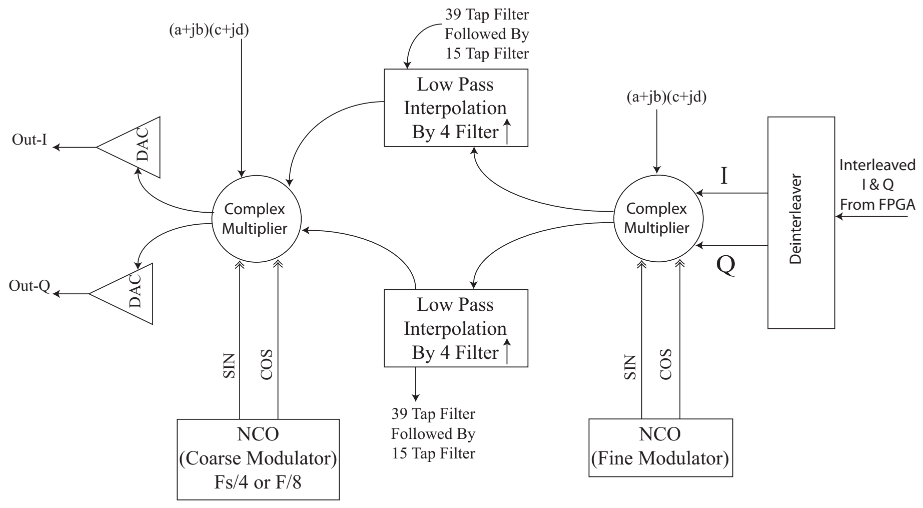


Figure 8.3.4: A schematic diagram of the USRP DUC.

- Analogue Digital Converter.
  There are 4 high-speed 12-bit AD converters. The sampling rate is 64M samples per second. In principle, it could digitize a band as wide as 32MHz. The AD converters can bandpass-sample signals of up to about 200MHz. The higher the frequency of the sampled signal, the more the SNR will be degraded by jitter. 100MHz is the recommended upper limit.
  The full range of the ADCs is 2V peak to peak, and the input is 50 ohms differential. This is 40mW, or 16 dBm. There is a programmable gain amplifier (PGA) before the ADCs to amplify the input signal to utilize the entire input range of the ADCs, in case the signal is weak. The PGA is up to 20 dB.

- Digital Analogue Converter.
  At the transmitting path, there are also 4 high-speed 14-bit DA converters. The DAC clock frequency is 128 MS/s, so Nyquist frequency is 64MHz. However, we will probably want to stay below it to make filtering easier. A useful output frequency range is from DC to about 44MHz. The DACs can supply 1V peak to a 50 ohm differential load, or 10mW (10 dBm). There is also PGA used after the DAC, providing up to 20 dB gain. This PGA is software programmable.

- Daughter boards.
  There are different types of daughter boards, which can be inserted in both sides of the mother board. These daughter boards have different capabilities, to know more about it, refer to the original sources as [18], [19].

  - Basic TX/RX Daughter boards.
    Each has two SMA connectors that can be used to connect external up/down tuners or signal generators. We can treat it as an entrance or an exit for the signal without affecting it. Some form of external RF front end is required.
  - Low Frequency TX/RX Daughter boards.
    The LFTX and LFRX are very similar to the Basic-TX and Basic-RX, respectively, with 2 main differences. Because the LFTX and LFRX use differential amplifiers instead of transformers, their frequency response extends down to DC.
  - TVRX Daughter boards.
    This is a receive-only daughter board.
  - DBSRX Daughter boards.
    This is a receive-only daughter board.
  - RFX Daughter boards.
    The RFX family of daughter boards is a complete RF transceiver system.

At the end of this section, it is a suitable summary to check Fig.8.3.5 . This figure shows the schematic diagram of the mother board, for both receiving and transmitting paths. This figure shows how do we receive a signal from the RF front passing by ADC, then MUX, followed by DDC, till it reach the USB port of the PC (receiver path). At the same time it shows how the signal goes out from the USB port to the sender front end (transmitter path), through DEMUX, then DUC and DAC.

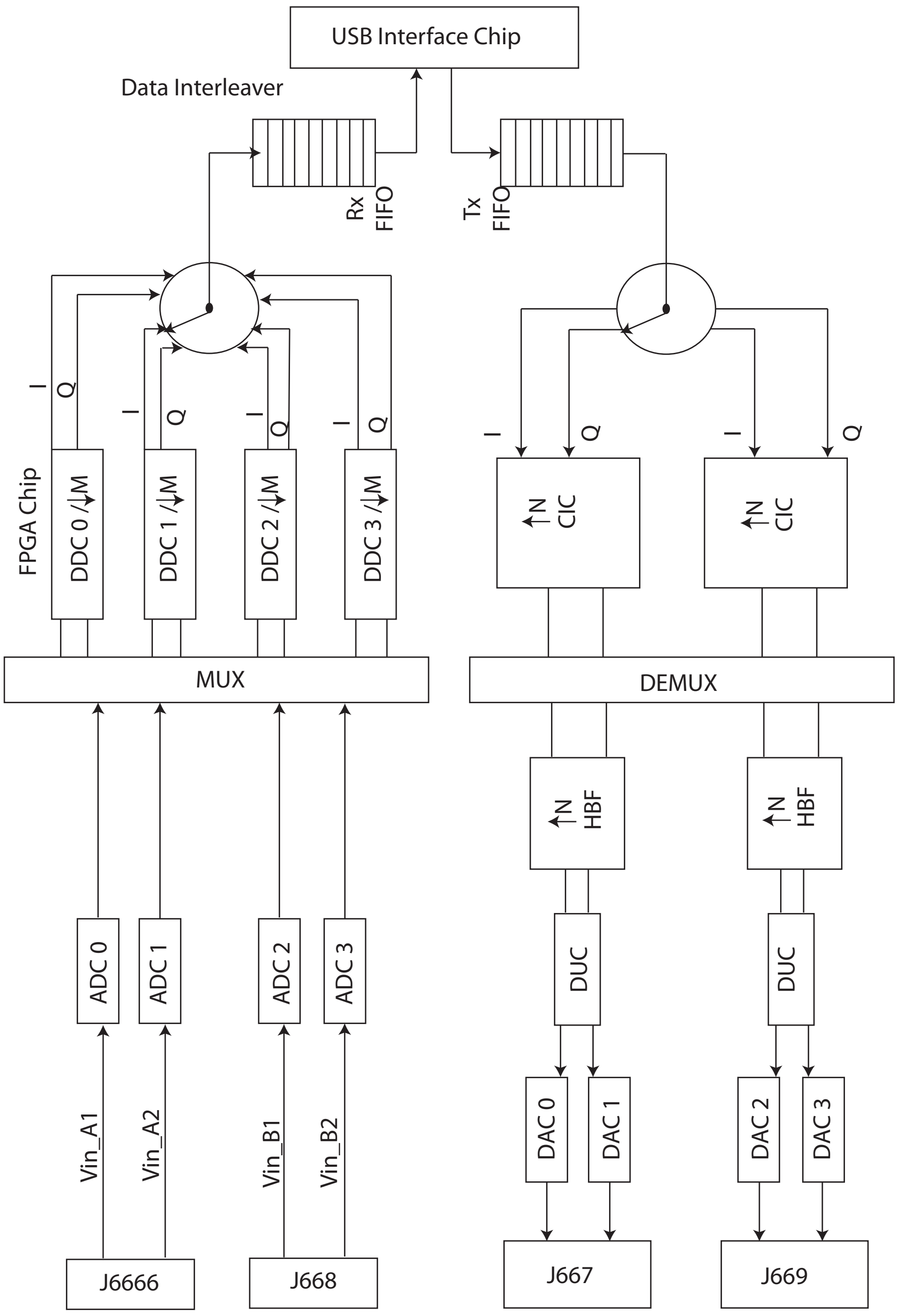


Figure 8.3.5: A schematic diagram of the USRP motherboard, transmit and receive path.

8.4 Packet Recovery Transceiver

In this section we are discussing one of the two parts of the our practical implementation. This part works as a half duplex transceiver device. So, one USRP device can transmit and receiver signals to and from the other USRP. In wireless channels fading effect cause the loss of a number of packets. As a solution for this problem, we use a simple acknowledgment (ACK) exchange policy between the transceivers to assure the complete reception of the packets. As we have pinpointed to in Sect.8.2, using python, GNU Radio programmer can glue between the C++ signal processing blocks. Gluing among the different blocks enables us to create a whole complete system. The gluing of the system's parts creates a system flow graphs. In this section we will have a fast overview of a GNU Radio's flow graph system. After that we explain our transceiver model, followed by snapshot results. In this section, as in the next sections, we might explain some code samples which is crucial for understanding the system.

8.4.1 A GNU Radio System Flow Graph using Python

As explained in Sect.8.2, the idea of the flow graph system (which based on graph theory) is to connect the C++ signal processing blocks together to form a whole system. To give GNU Radio more flexibility, each C++ signal processing block performs one task. Lets see some examples for already created flow graph systems by connecting individual signal processing blocks.

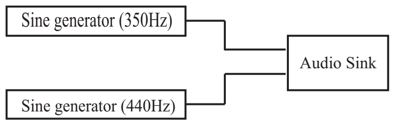


Figure 8.4.1: The flow graph of dial tone generator.

- Dial Tone Generator. As in Fig.8.4.1, we can see how it is possible to create a dial tone generator using three blocks. One sink and two sources. So as we can see all source do not have input, while all sinks can not have output.

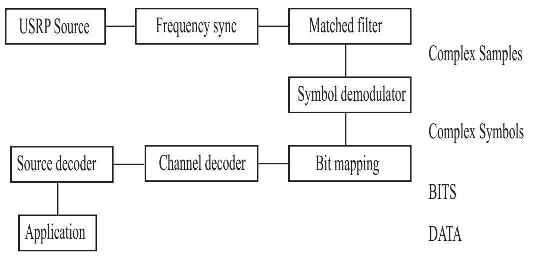


Figure 8.4.2: The flow graph of QPSK demodulator.

- QPSK Demodulator. Figure 8.4.2 shows how to connect different mono task block to form the QPSK demodulator system. As we know from communication basic letirature these are the necessary blocks for creating this kind of demodulator. We notice that there are many middle blocks which has both input and output. Some blocks has the input and output as complex data, some have complex input and bits output.

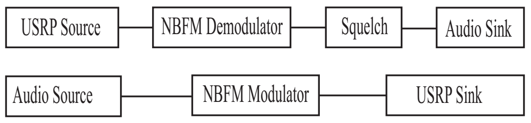


Figure 8.4.3: The flow graph of a Walkie Talkie.

- Walkie Talkie. I never created walkie talkie in my life, but I've tested the idea behind this flow graph and it works well. The idea behind the walkie talkie flow graph, Fig.8.4.3, is to show the capability of GNU Radio flow graph system to handle the parallel independent flow graphs.

From the previous flow graphs we can conclude the following:

- All signal processing in GNU Radio is done through flow graphs.
- A flow graph consists of blocks. A block does one signal processing operation.
- Data passes between blocks in various formats, complex or real integers, etc.

- Every flow graph needs at least one sink and source.

As an example to show the how to connect the flow graph using python consider the following code lines (this code is taken from the module gnuradio-examples/python/audio/dial_tone.py).
$src0 = gr.sig\_source\_f(sample\_rate, gr.GR_SIN\_WAVE, 350, ampl)$
$src1 = gr.sig\_source\_f(sample\_rate, gr.GR\_SIN\_WAVE, 440, ampl)$
$dst = audio.sink(sample\_rate, "")$
$self.connect(src0, (dst, 0))$
$self.connect(src1, (dst, 1))$
The first and the second code lines create signal source with different frequency. The third line creates the audio sink. Finally the last two lines connect the first and the second sources with the audio sink. Because Python is an OOP, we pass the returned value of the module's (class) constructor to a specific named variable, which can be used always as an object of that module. Finally, we are going to mention some important characteristics of the GNU Radio flow graph system.

- It is possible to combine two blocks into one hierarchical block using the module '$gr.hier\_block2$'. This make the work easier and more flexible sometimes. The hierarchical block can be a source, sink or middle block.

- Sometimes it is crucial to use two or more flow graphs working at the same time. GNU Radio's flow graph system enables us to do so. To do multiple flow graph control we have to create a top block and connect all the flow graph (which will be considered as sub flow graphs) to this top block. This is as explained in the Walkie Talkie example.

- Dynamic flow graph control is possible by using some already created methods as stop, run, connect, start, wait, etc. It is also possible to control a flow graph from a different module (or class) by calling its corresponding object.

8.4.2 Implemented Transceiver System.

After thinking about a nice and easy way for explaining the system, we decided to show an overall system flow graph, and then explain it. Each of the graph rectangles correspond to one programming block (made by either C++ or Python). In each sub flow-graph the blocks are connected together from right (up) to left (down). The dotted line indicate that there is a relation between the connected blocks. Figure 8.4.4 shows the whole system flow-graph. We explain each first level sub flow-graph individually, starting from '$usrp\_receive\_path$'.

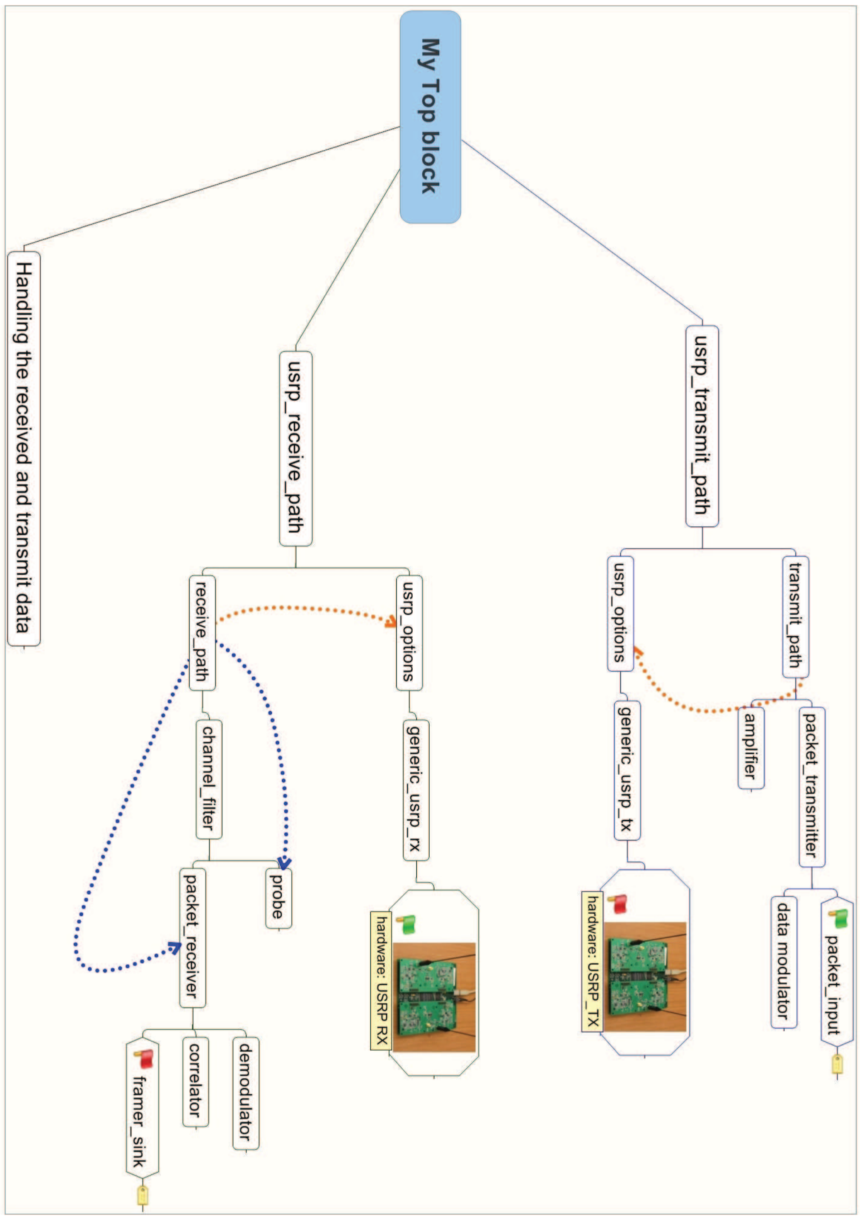


Figure 8.4.4: The flow graph of the implemented transceiver.

- $usrp\_receive\_path$: This block acts as a hierarchical block for the sub flow-graph's blocks of the receiving path. As we mentioned in Sect.8.4.1, it is possible to combine many blocks which have different tasks into one large block. This large block deals with all the sub tasks in a specified organized timing, so it increases the reliability and flexibility of the system.

- $HardwareUSRP\_RX$: This is the receiving Hardware.

- $generic\_usrp\_rx$: This block takes care of setting the hardware (USRP) configurations. It set the PGA before the ADC. It passes the required decimation rate to the corresponding DDC, and also decides the multiplexer's value based on how many channels do we have. Power gain, Center frequency are also controlled in a flexible way using this block. All the previously mentioned parameters can be specified for both daughter boards (DB) A and B. Some of them are independently assigned for DB A and B, while others must have the same values for both DB. All of these parameters are taken from the $'usrp\_options'$ block, which is entrusted to carefully decides these values.

- $usrp\_options$: This block is responsible for recognizing the users' desired options of both transmitting and receiving paths. It also examines the user's input data, and the desired device configuration, whether or not they are available, suitable, and achievable.

- $receive\_path$: This block deals with the received signal just after receiving it from hardware. It is responsible for channel filtering, carrier checking, demodulation, error checking. As for error checking, there are two built-in error checking methods. One way is Cyclic Redundancy Check (CRC). The other one is code access correlation for synchronization.

- $channel\_filter$:Is the closest block of the receiving flow graph to the hardware part. We create filter coefficients corresponding to the desired filter type. Here we use a low pass filter. Furthermore, we can select different windows types such as a Hanning window '$gr.firdes.WIN\_HANN$'. Other windows are available as, Hamming, Blackman, Kaiser and Blackman Harris. The main role for this block is to compensate for the transmission channel.

- $probe$: It Computes a running average of the magnitude squared of the the input. The level and indication as to whether the level exceeds a threshold can be retrieved with the level and unmuted accessors. This block can be thought of as carrier power detector.

- $packet\_receive$: As its named, this block is for managing the received packets.

- $demodulator$: This is a demodulation block should be assigned corresponding to the modulator in the transmitting path. It is possible to select different types of demodulation as PSK (from 2 to 8), DBPSK (2-8) and QAM (8, 16, 64, 256).

- *correlator*: This block examines its input for specified access code, one bit at a time. It is considered for synchronization purposes. The input data of this block is: stream of bits, 1 bit per input byte (data in LSB). The output data is: stream of bits, 2 bits per output byte (data in LSB, flag in next higher bit).

- *framer_sink*: Works as a message queue to hold the packet from the physical layer and then use it in *handling the receive and transmit data flow−graph*.

Then we explain '*usrp_transmit_path*'.

- *usrp_transmit_path*: This block acts as '*usrp_receive_path*', but for the transmitting path.

- *packet_input*: This is the source of data which we want to send. Here we use '*message_source*' module to convert the received (input) messages into a stream.

- *data_modulator*: This is a modulation block should be assigned corresponding to the demodulator in the receiving path. It is possible to select different types of modulation as PSK (from 2 to 8), DBPSK (2-8) and QAM (8, 16, 64, 256).

- *packet_transmitter*: As its named, this block is for managing the transmit packets.

- *amplifier*: This block multiply the signal by a desired amplitude to amplify the signal.

- *transmit_path*: This block deals with the transmit signal before passing it to the hardware. It is responsible for generating, modulating and amplifying the signals.

- *usrp_options*: Performs just as *usrp_options* in *usrp_receive_path*.

- *generic_usrp_tx*: This block takes care of setting the hardware (USRP) configurations. It passes the required interpolation rate to the corresponding DUC, and also decides the de-multiplexer's value based on how many channels do we have. Power gain, Center frequency are also controlled in a flexible way using this block. All the previously mentioned parameters can be specified for both daughter boards (DB) A and B. Some of them are independently assigned for DB A and B, while others must have the same values. All of these parameters are taken from the '$usrp\_options$' block, which is entrusted to carefully decides these values.

- $Hardware : USRP\_TX$: This is the receiving Hardware.

Finally we explain '$handling\ the\ receive\ and\ transmit\ data$'.

- $handling\ the\ received\ and\ transmit\ data$: In this block we handle both the transmitted and received data. This block is created by only python.
  As for the transmitted data we decide which file do we want to transmit. Then we divide the whole file into a suitable number of packets according to USRP specifications and roles. After that, we attached the corresponding header to the data. Sometime we transmitted an ACK, which takes only one packet to be transmitted. This ACK packets are organized through the received data handling part. After we create the data/ACK packets we pass it to '$usrp\_tranmit\_path$' to prepare this data to be transmittable through USRP.
  On the other hand, when we receive the packet from '$usrp\_receive\_path$' block we separate both header and data in the packet. This enables us to recognize whether we received packet is ACK packet or data packet. Starting from this point we begin the explanation of the simple proposed exchange system for recovering the lost packets. According to the header we take one of the next actions, as follow:

  - If the header is for ACK packet; We inform the user that the sent message has been received in the other side correctly.
  - If the header is for NACK packet. This happens when the transmitted packets were lost in the channel; Therefore, we request a data retransmission from the transmitter handler. In this case we only transmit the lost data.
  - If the header is for data packet; We check the packet number if it is the last packet we close and save the file. If it is not the last packet, we add it to the receiving matrix.
  - If the header is for a lost retransmitted data packet; We add it to the correct location in the receiving matrix. Then save the file.

  This recovery algorithm is explained in Fig.8.4.5.

  At last, Fig.8.4.6 shows a summary block diagram for the our communication system.

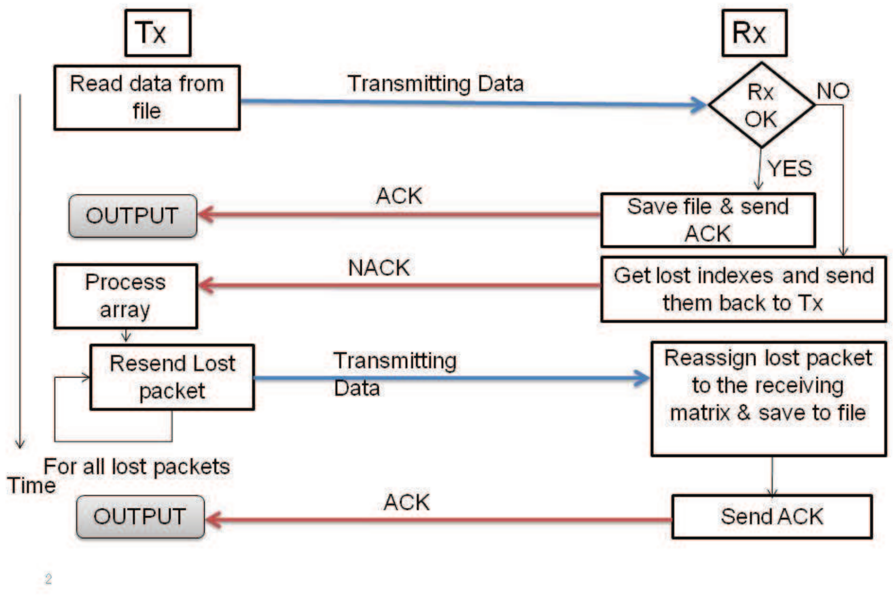


Figure 8.4.5: The Recovery Algorithm for Lost Packets.

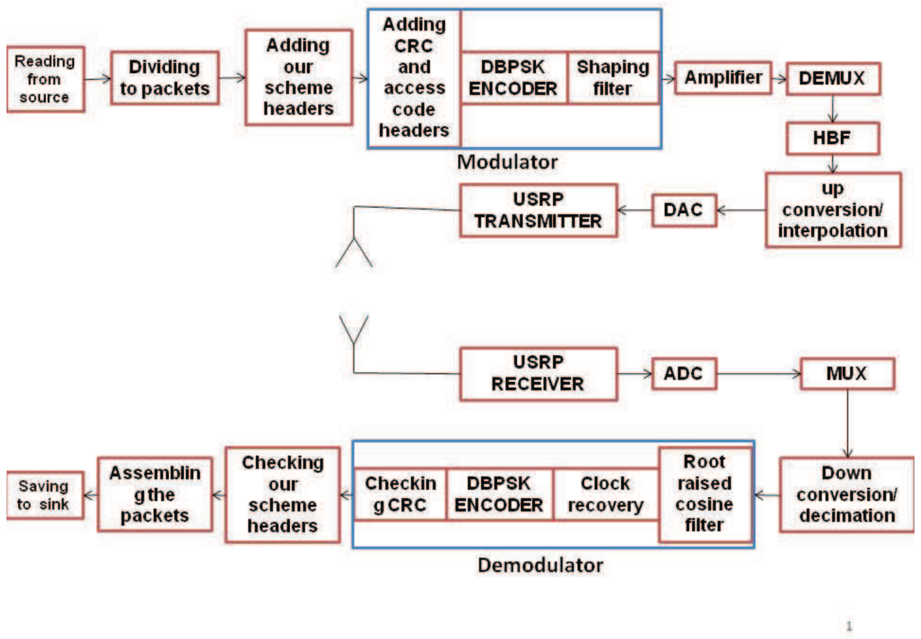


Figure 8.4.6: The Overall Block Diagram of our communication system, Implemented using GNU Radio.

8.5 Spectrum Sensor

In this section we explain our GNU Radio based spectrum sensor. This sensor is a reconfigurable sensor, which has the ability to sense any number of channels with any required bandwidth per channel. All it takes is to change its parameters. This sensor works in two modes, Noise calibration and sensing mode.
Any sensing period starts by noise calibration mode to decide the noise variance which is necessary to calculate the threshold. Depending on how accurate do we want the observed noise variance to be, we should increase the number of averaged samples of the noise. Therefore, this mode is operated with an option for increasing the number of iteration independently from the sensing mode. Before operating in this mode we should assure that the radio environment is free from any signals, especially in the sensed sub-channels. Otherwise all available signals will be considered under the noise level, thus, we will not be able to detect them.
After it finishes the noise calibration mode the sensor goes automatically into the sensing mode. There are many factors to be considered for deciding how this mode is operating.

- The frequency and time resolution: which decide how many FFT bins and the corresponding number of sensed channels per one FFT block. If the highest sampling rate is considered, for USRP is 8M, considering an overlapping factor is 0.25, and the bandwidth per sub-channel is 1 M, then we can sense up to 6 sub-channels at a time. This sub-channels can be scanned per one chunk. If we increase the overall sensed spectrum to 100 sub-channels, we have to change the center frequency of the devise $[100/6] = 17$ times.

- The overlapping between the sensed channels. The larger the overlapped period, till a certain value, the better the sensing, however it increases the required sensing time.

Figure 8.5.1 shows the flow graph of the sensor. As in the transceiver part each of these blocks created either by python or C++. In the following we explain the task of each block individually

- $Hardware$: The sensed signal comes from the USRP. Then we input it to a stream-to-vector transformer.

- $s2v$: This block converts a stream of items into a stream of blocks containing $nitems\_per\_block$. So we use it to convert the inputed stream signal to series vectors (blocks). Each of those blocks have the same size of FFT.

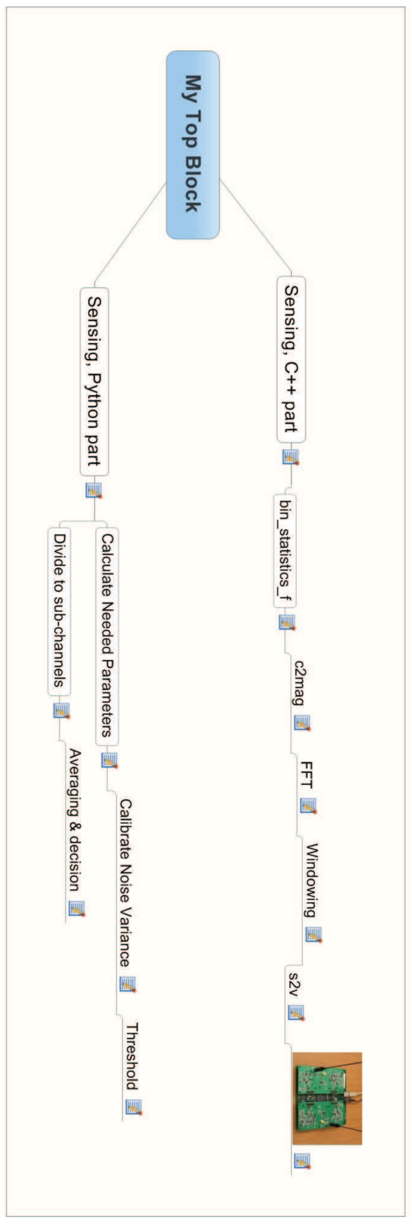


Figure 8.5.1: The Overall Block Diagram of our Sensing system, Implemented using GNU Radio.

- $Windowing$: We use $black\_man\_harris$ window to smoothen the time signal. Without using a time window we cannot reduce the leakage in the filter's frequency response.

- FFT: This block computes the forward or reverse FFT, with complex input and output. According to the size of FFT and the sampling rate we divide the whole spectrum into sub-channels.

- $c2mag$: It computes the magnitude and square for the complex input.
- $bin\_statistics\_f$: This block has many tasks, it can be considered as the heart of this flow graph, some of these tasks are:
  - It keeps track of the signal output of the squaring device.
  - It waits for a pre-specified tune delay to make sure that we obtained the samples corresponding to the assigned center frequency.
  - It also waits for a dwelling time, is needed to observe a pre-specified number of samples for averaging.
  - Finally, it calls the python's tuning function to tune and change the frequency to the next value (this is effective in case of sensing a wide spectrum).
- $Sensing,\ C++\ part$: This is not a signal processing block, but it is responsible to deliver the data obtained by the C++ signal processing blocks to the python blocks.
- $Sensing,\ Python\ part$: Starting from this block we begin the python part. It takes the sensed values from '$bin\_statistics\_f$' block.
- $Calculated\ Needed\ Parameters$: In this block we discuss some hardware considerations and parameters that affect the spectrum sensing process. These parameters have variable or fixed values. Some of these values are expressed as:
  - Tune Delay Time (1ms default):
    To achieve a correct sensing performance we should wait (a delay time) for the ADC samples of the specified wanted centered frequency. This delay occur because of many delays along the digitization path (RF synthesizer, settling time and propagation pipe-FPGA and USB transferring time).
  - Dwell Delay Time (10ms default):
    The purpose of this delay time is to stay at the same frequency to get enough samples to achieve a certain detection sensitivity. The more we increase this time the more samples we have, to be averaged, this improve the performance of sensing.

- USB Limitation:
  As we have discussed in a previous section, USRP is using a USB micro-controller to interface between the USRP and the USB port. Because of the speed limitation, the USB cannot keep up with the USRP speed (64 Mbit/sec). This obligates us to use the decimation rate higher than 8, results in a final sampling rate $= 64M/8 = 8M$.

- $Calibrate\ Noise\ Variance$: This block is the first block in the process of sensing. We make sure that there is no primary user's signals and then start assembling the noise samples. This is assembling process is average over all sample to get a noise variance as close as possible to the real one. If we are sensing a wide bandwidth we separate the bands and then average over all of them. Therefore, this process takes more than it does take us to scan all channels.

- $Threshold$: In this block we use both the estimated noise variance and the number of averaged samples to calculated the threshold, as given by

$$\gamma_{thres,FA} = \sigma_n^2(1 + \frac{Q^{-1}(P_{FA})}{\sqrt{N/2}}). \tag{8.5.1}$$

  where σ_n^2 is the noise variance which has been estimated in a previous block, Q is the macrum function and N is the number of averaged samples.

- $Divide\ into\ sub-channels$: If we are sensing a wide frequency band, in this block we divide the samples into different sub-bands and sense each band individually.

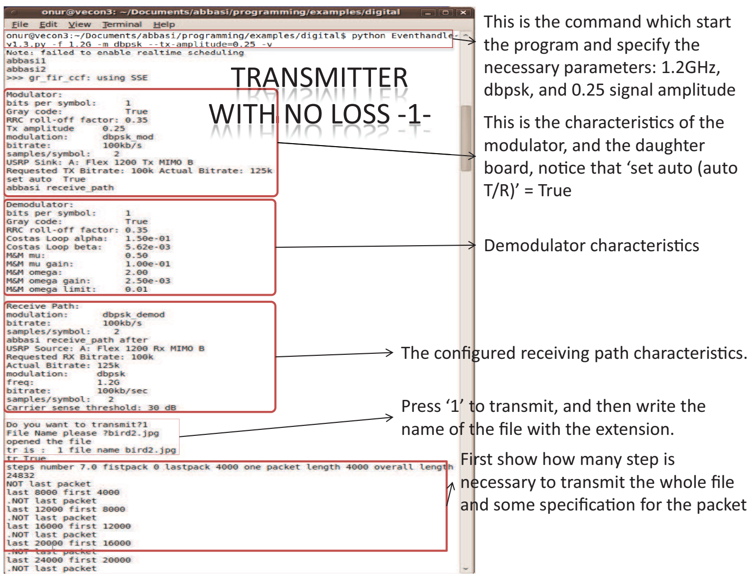
- $Averaging\ \&\ decision$: We just average each sub-channel's samples to find the test statistic. After that we compare the test statistic to a threshold to make the decision, either the primary user exist or not.

8.6 Transceiver and Sensor Results

In this section we show the results of both the transceiver and the sensor GNU Radio based devices. Some results are only snap shots of the output command window, which is to show how the system works. While others are performance evaluating results.

8.6.1 Transceiver results

At first we show the transmitter with no lost packet. In this case the receiver will receive all the packets correctly and then transmit an ACK packet to the transmitter, which is going to show it in the output command line window. Figure 8.6.1 shows the system parameters and then starting transmitting.


onur@vecon3: ~/Documents/abbasi/programming/examples/digital
File Edit View Terminal Help
TRANSMITTER
WITH NO LOSS -1-

```
onur@vecon3:~/Documents/abbasi/programming/examples/digital$ python Eventhandle-
v1.3.py -f 1.2G -m dbpsk --tx-amplitude=0.25 -v
Note: failed to enable realtime scheduling
abbasi1
abbasi2
>>> gr_fir_ccf: using SSE
Modulator:
bits per symbol:     1
Gray code:           True
RRC roll-off factor: 0.35
Tx amplitude      0.25
modulation:       dbpsk_mod
bitrate:          100kb/s
samples/symbol:     2
USRP Sink: A: Flex 1200 Tx MIMO B
Requested TX Bitrate: 100k Actual Bitrate: 125k
set auto  True
abbasi receive_path
Demodulator:
bits per symbol:     1
Gray code:           True
RRC roll-off factor: 0.35
Costas Loop alpha:   1.50e-01
Costas Loop beta:    5.62e-03
M&M mu:              0.50
M&M mu gain:         1.00e-01
M&M omega:           2.00
M&M omega gain:      2.50e-03
M&M omega limit:     0.01
Receive Path:
modulation:       dbpsk_demod
bitrate:          100kb/s
samples/symbol:     2
abbasi receive_path after
USRP Source: A: Flex 1200 Rx MIMO B
Requested RX Bitrate: 100k
Actual Bitrate: 125k
modulation:      dbpsk
freq:            1.2G
bitrate:         100kb/sec
samples/symbol:    2
Carrier sense threshold: 30 dB
Do you want to transmit?1
File Name please ?bird2.jpg
opened the file
tr is :  1 file name bird2.jpg
tr True
steps number 7.0 fistpack 0 lastpack 4000 one packet length 4000 overall length
24832
NOT last packet
last 8000 first 4000
.NOT last packet
last 12000 first 8000
.NOT last packet
last 16000 first 12000
.NOT last packet
last 20000 first 16000
.NOT last packet
last 24000 first 20000
.NOT last packet
```

This is the command which start the program and specify the necessary parameters: 1.2GHz, dbpsk, and 0.25 signal amplitude

This is the characteristics of the modulator, and the daughter board, notice that 'set auto (auto T/R)' = True

Demodulator characteristics

The configured receiving path characteristics.

Press '1' to transmit, and then write the name of the file with the extension.

First show how many step is necessary to transmit the whole file and some specification for the packet

Figure 8.6.1: Transceiver Results, Transmitter side # 1.

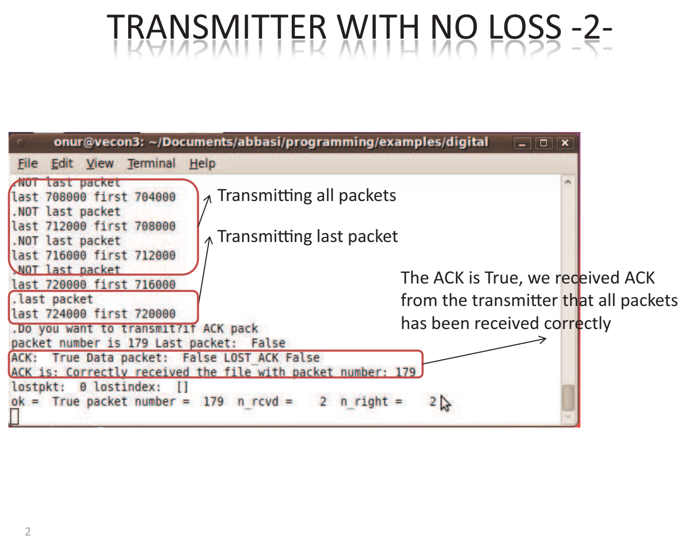
Figure 8.6.2 shows the last part of the transmitting process.

Figure 8.6.3 shows the receiver system parameters and starting of the packet receiving process.

Figure 8.6.4 shows the last stage of receiving the transmitted packet. After finishing receiving it starts to send ACK.

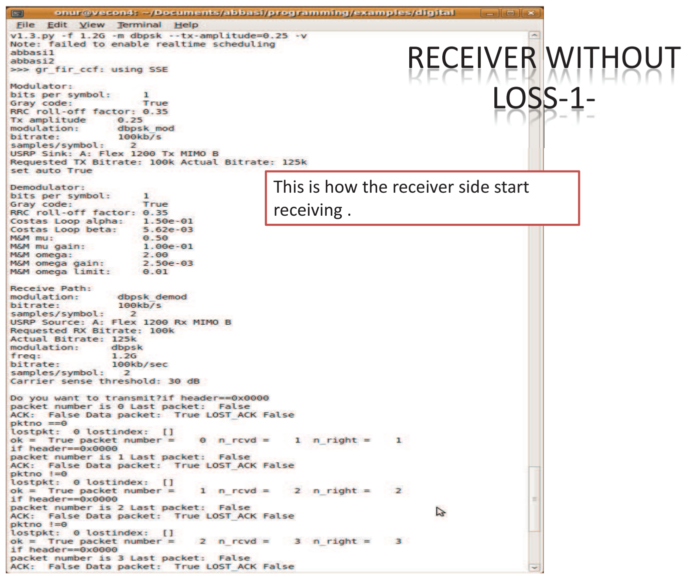
Figure 8.6.5 shows the receiver side going from a receiver to a transmitter.

Figure 8.6.6 shows the receiver side when there are lost packets. These packets lost because of the sever fading channel or low SNR environment.


TRANSMITTER WITH NO LOSS -2-
onur@vecon3: ~/Documents/abbasi/programming/examples/digital
File Edit View Terminal Help
Transmitting all packets
Transmitting last packet
The ACK is True, we received ACK from the transmitter that all packets has been received correctly
2

```
.NOT last packet
last 708000 first 704000
.NOT last packet
last 712000 first 708000
.NOT last packet
last 716000 first 712000
.NOT last packet
last 720000 first 716000
.last packet
last 724000 first 720000
.Do you want to transmit?if ACK pack
packet number is 179 Last packet:  False
ACK:  True Data packet:  False LOST_ACK False
ACK is: Correctly received the file with packet number: 179
lostpkt:  0 lostindex:  []
ok =  True packet number =  179  n_rcvd =    2  n_right =    2
```

Figure 8.6.2: Transceiver Results, Transmitter side # 2.

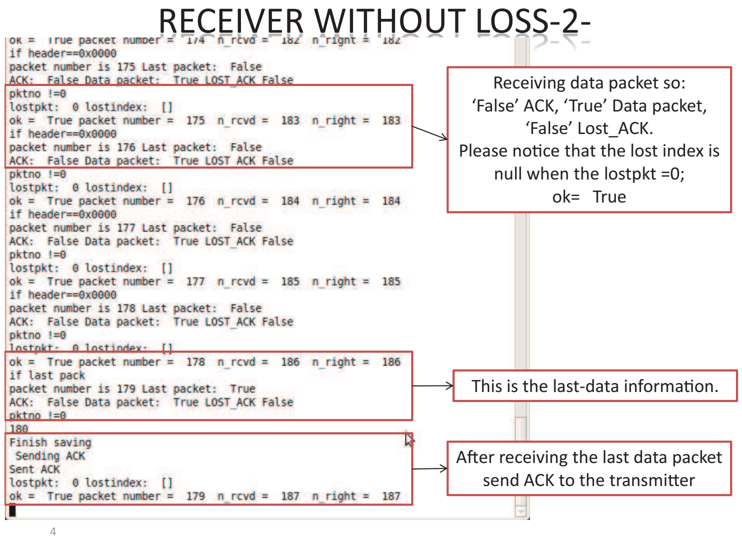

onur@vecon4: ~/Documents/abbasi/programming/examples/digital
File Edit View Terminal Help
RECEIVER WITHOUT LOSS-1-
This is how the receiver side start receiving .

```
v1.3.py -f 1.2G -m dbpsk --tx-amplitude=0.25 -v
Note: failed to enable realtime scheduling
abbasi1
abbasi2
>>> gr_fir_ccf: using SSE

Modulator:
bits per symbol:      1
Gray code:            True
RRC roll-off factor: 0.35
Tx amplitude      0.25
modulation:       dbpsk_mod
bitrate:          100kb/s
samples/symbol:     2
USRP Sink: A: Flex 1200 Tx MIMO B
Requested TX Bitrate: 100k Actual Bitrate: 125k
set auto True

Demodulator:
bits per symbol:      1
Gray code:            True
RRC roll-off factor: 0.35
Costas Loop alpha:    1.50e-01
Costas Loop beta:     5.62e-03
M&M mu:               0.50
M&M mu gain:          1.00e-01
M&M omega:            2.00
M&M omega gain:       2.50e-03
M&M omega limit:      0.01

Receive Path:
modulation:       dbpsk_demod
bitrate:          100kb/s
samples/symbol:     2
USRP Source: A: Flex 1200 Rx MIMO B
Requested RX Bitrate: 100k
Actual Bitrate: 125k
modulation:      dbpsk
freq:            1.2G
bitrate:         100kb/sec
samples/symbol:    2
Carrier sense threshold: 30 dB

Do you want to transmit?if header==0x0000
packet number is 0 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno ==0
lostpkt:  0 lostindex:  []
ok =  True packet number =    0  n_rcvd =    1  n_right =    1
if header==0x0000
packet number is 1 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =    1  n_rcvd =    2  n_right =    2
if header==0x0000
packet number is 2 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =    2  n_rcvd =    3  n_right =    3
if header==0x0000
packet number is 3 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
```

Figure 8.6.3: Transceiver Results, Receiver side # 3.

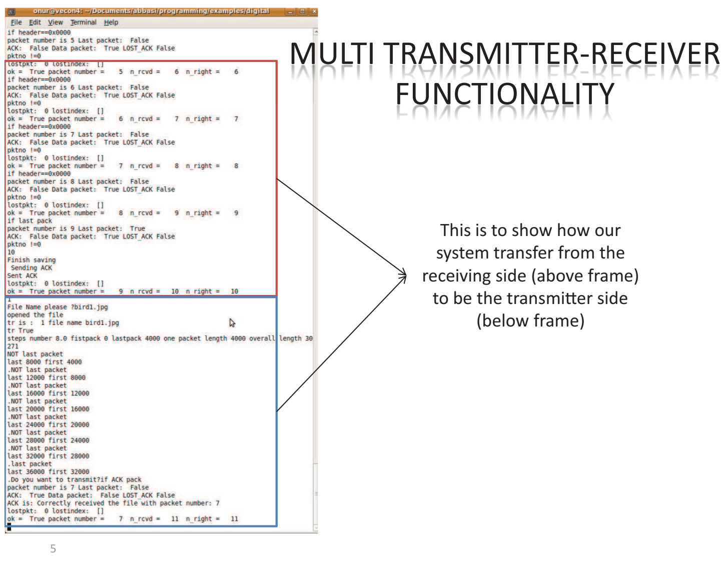
Figure 8.6.7 shows the last stage of receiving the lost packets. After that it sends NACK to notify the transmitter about the lost packets. Then it receives only the lost packets.

Figure 8.6.8 shows the last stage of receiving the lost packets after retransmitting them by the transmitter. Then it transmit an ACK to the transmitter.


RECEIVER WITHOUT LOSS-2-
Receiving data packet so: 'False' ACK, 'True' Data packet, 'False' Lost_ACK. Please notice that the lost index is null when the lostpkt =0; ok= True
This is the last-data information.
After receiving the last data packet send ACK to the transmitter
4

```
ok =  True packet number =   174  n_rcvd =   182  n_right =   182
if header==0x0000
packet number is 175 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =   175  n_rcvd =   183  n_right =   183
if header==0x0000
packet number is 176 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =   176  n_rcvd =   184  n_right =   184
if header==0x0000
packet number is 177 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =   177  n_rcvd =   185  n_right =   185
if header==0x0000
packet number is 178 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =   178  n_rcvd =   186  n_right =   186
if last pack
packet number is 179 Last packet:  True
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
180
Finish saving
 Sending ACK
Sent ACK
lostpkt:  0 lostindex:  []
ok =  True packet number =   179  n_rcvd =   187  n_right =   187
```

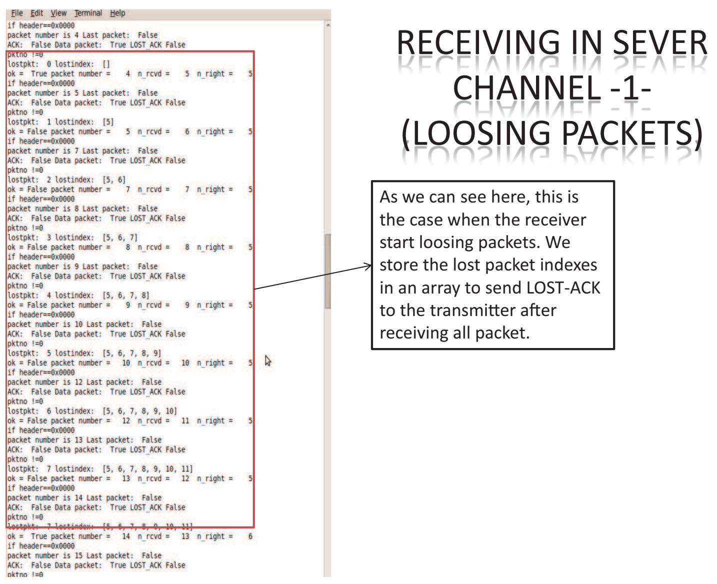
Figure 8.6.4: Transceiver Results, Receiver side # 4.


onur@vecon4: ~/Documents/abbasi/programming/examples/digital
File Edit View Terminal Help
MULTI TRANSMITTER-RECEIVER FUNCTIONALITY
This is to show how our system transfer from the receiving side (above frame) to be the transmitter side (below frame)
5

```
if header==0x0000
packet number is 5 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =    5  n_rcvd =    6  n_right =    6
if header==0x0000
packet number is 6 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =    6  n_rcvd =    7  n_right =    7
if header==0x0000
packet number is 7 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =    7  n_rcvd =    8  n_right =    8
if header==0x0000
packet number is 8 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =    8  n_rcvd =    9  n_right =    9
if last pack
packet number is 9 Last packet:  True
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
10
Finish saving
 Sending ACK
Sent ACK
lostpkt:  0 lostindex:  []
ok =  True packet number =    9  n_rcvd =   10  n_right =   10
1
File Name please ?bird1.jpg
opened the file
tr is :  1 file name bird1.jpg
tr True
steps number 8.0 fistpack 0 lastpack 4000 one packet length 4000 overall length 30
271
NOT last packet
last 8000 first 4000
.NOT last packet
last 12000 first 8000
.NOT last packet
last 16000 first 12000
.NOT last packet
last 20000 first 16000
.NOT last packet
last 24000 first 20000
.NOT last packet
last 28000 first 24000
.NOT last packet
last 32000 first 28000
.last packet
last 36000 first 32000
.Do you want to transmit?if ACK pack
packet number is 7 Last packet:  False
ACK:  True Data packet:  False LOST_ACK False
ACK is: Correctly received the file with packet number: 7
lostpkt:  0 lostindex:  []
ok =  True packet number =    7  n_rcvd =   11  n_right =   11
```

Figure 8.6.5: Transceiver Results, Multi-Transmitter-Receiver functionality, # 1.

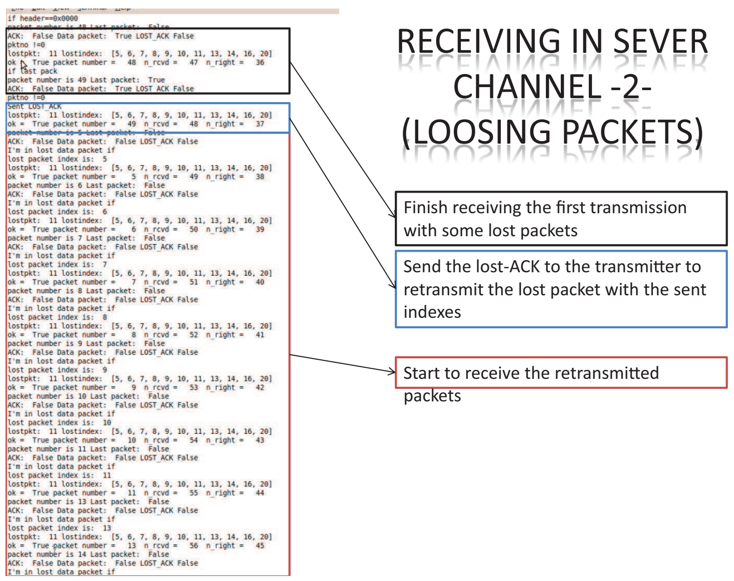
Figure 8.6.9 shows the transmitter side after receiving NACK then it retransmit only the lost packets, using their sent indexes. Finally, it receives ACK from the receiver side.


File Edit View Terminal Help
RECEIVING IN SEVER CHANNEL -1- (LOOSING PACKETS)

```
if header==0x0000
packet number is 4 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  0 lostindex:  []
ok =  True packet number =    4  n_rcvd =    5  n_right =    5
if header==0x0000
packet number is 5 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  1 lostindex:  [5]
ok = False packet number =    5  n_rcvd =    6  n_right =    5
if header==0x0000
packet number is 7 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  2 lostindex:  [5, 6]
ok = False packet number =    7  n_rcvd =    7  n_right =    5
if header==0x0000
packet number is 8 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  3 lostindex:  [5, 6, 7]
ok = False packet number =    8  n_rcvd =    8  n_right =    5
if header==0x0000
packet number is 9 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  4 lostindex:  [5, 6, 7, 8]
ok = False packet number =    9  n_rcvd =    9  n_right =    5
if header==0x0000
packet number is 10 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  5 lostindex:  [5, 6, 7, 8, 9]
ok = False packet number =   10  n_rcvd =   10  n_right =    5
if header==0x0000
packet number is 12 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  6 lostindex:  [5, 6, 7, 8, 9, 10]
ok = False packet number =   12  n_rcvd =   11  n_right =    5
if header==0x0000
packet number is 13 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  7 lostindex:  [5, 6, 7, 8, 9, 10, 11]
ok = False packet number =   13  n_rcvd =   12  n_right =    5
if header==0x0000
packet number is 14 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  7 lostindex:  [5, 6, 7, 8, 9, 10, 11]
ok =  True packet number =   14  n_rcvd =   13  n_right =    6
if header==0x0000
packet number is 15 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
```

As we can see here, this is the case when the receiver start loosing packets. We store the lost packet indexes in an array to send LOST-ACK to the transmitter after receiving all packet.

Figure 8.6.6: Transceiver Results, Receiver side with Lost Packets # 1.



```
if header==0x0000
packet number is 48 Last packet:  False
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   48  n_rcvd =   47  n_right =   36
if last pack
packet number is 49 Last packet:  True
ACK:  False Data packet:  True LOST_ACK False
pktno !=0
Sent LOST_ACK
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   49  n_rcvd =   48  n_right =   37
packet number is 5 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  5
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =    5  n_rcvd =   49  n_right =   38
packet number is 6 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  6
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =    6  n_rcvd =   50  n_right =   39
packet number is 7 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  7
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =    7  n_rcvd =   51  n_right =   40
packet number is 8 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  8
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =    8  n_rcvd =   52  n_right =   41
packet number is 9 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  9
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =    9  n_rcvd =   53  n_right =   42
packet number is 10 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  10
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   10  n_rcvd =   54  n_right =   43
packet number is 11 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  11
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   11  n_rcvd =   55  n_right =   44
packet number is 13 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  13
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   13  n_rcvd =   56  n_right =   45
packet number is 14 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
```

Finish receiving the first transmission with some lost packets

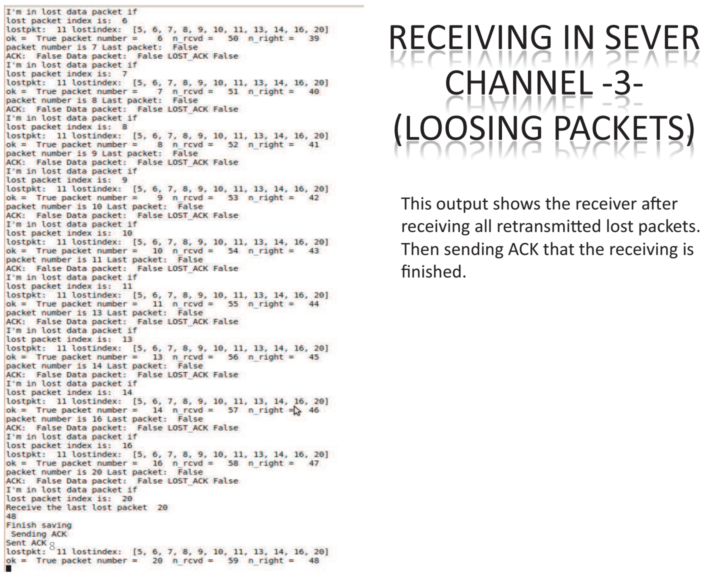
Send the lost-ACK to the transmitter to retransmit the lost packet with the sent indexes

Start to receive the retransmitted packets

Figure 8.6.7: Transceiver Results, Receiver side with Lost Packets # 2.

8.6.2 Sensor Results

In this subsection we are going to start by showing the command line window environment of the sensing process. Then we show result of evaluating the total observation time of sensing 102 channels versus the Dwelling time. After that, we show the results of evaluating the USRP sensitivity versus the number of averag-

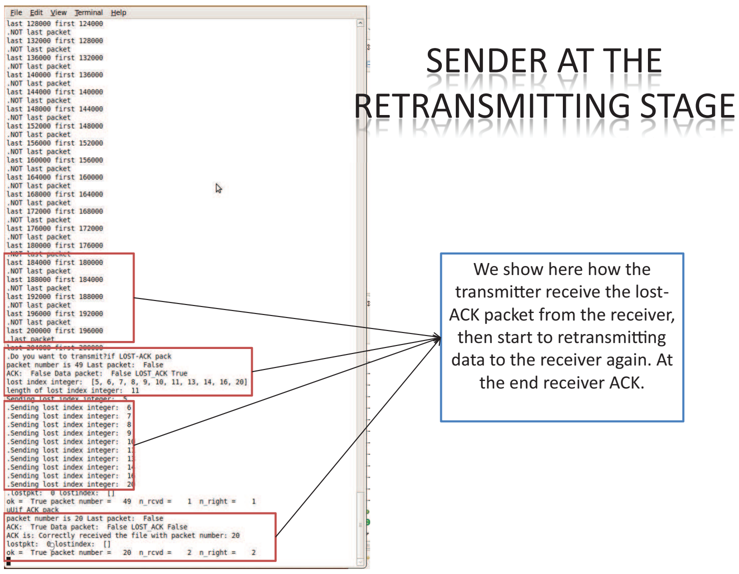


RECEIVING IN SEVER CHANNEL -3- (LOOSING PACKETS)

```
I'm in lost data packet if
lost packet index is:  6
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =    6  n_rcvd =   50  n_right =   39
packet number is 7 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  7
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =    7  n_rcvd =   51  n_right =   40
packet number is 8 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  8
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =    8  n_rcvd =   52  n_right =   41
packet number is 9 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  9
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =    9  n_rcvd =   53  n_right =   42
packet number is 10 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  10
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   10  n_rcvd =   54  n_right =   43
packet number is 11 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  11
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   11  n_rcvd =   55  n_right =   44
packet number is 13 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  13
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   13  n_rcvd =   56  n_right =   45
packet number is 14 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  14
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   14  n_rcvd =   57  n_right =   46
packet number is 16 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  16
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   16  n_rcvd =   58  n_right =   47
packet number is 20 Last packet:  False
ACK:  False Data packet:  False LOST_ACK False
I'm in lost data packet if
lost packet index is:  20
Receive the last lost packet  20
48
Finish saving
 Sending ACK
Sent ACK
lostpkt:  11 lostindex:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
ok =  True packet number =   20  n_rcvd =   59  n_right =   48
```

This output shows the receiver after receiving all retransmitted lost packets. Then sending ACK that the receiving is finished.

Figure 8.6.8: Transceiver Results, Receiver side with Lost Packets # 3.



SENDER AT THE RETRANSMITTING STAGE

```
last 128000 first 124000
.NOT last packet
last 132000 first 128000
.NOT last packet
last 136000 first 132000
.NOT last packet
last 140000 first 136000
.NOT last packet
last 144000 first 140000
.NOT last packet
last 148000 first 144000
.NOT last packet
last 152000 first 148000
.NOT last packet
last 156000 first 152000
.NOT last packet
last 160000 first 156000
.NOT last packet
last 164000 first 160000
.NOT last packet
last 168000 first 164000
.NOT last packet
last 172000 first 168000
.NOT last packet
last 176000 first 172000
.NOT last packet
last 180000 first 176000
.NOT last packet
last 184000 first 180000
.NOT last packet
last 188000 first 184000
.NOT last packet
last 192000 first 188000
.NOT last packet
last 196000 first 192000
.NOT last packet
last 200000 first 196000
.last packet
last 204000 first 200000
.Do you want to transmit?if LOST-ACK pack
packet number is 49 Last packet:  False
ACK:  False Data packet:  False LOST_ACK True
lost index integer:  [5, 6, 7, 8, 9, 10, 11, 13, 14, 16, 20]
length of lost index integer:  11
Sending lost index integer:  5
.Sending lost index integer:  6
.Sending lost index integer:  7
.Sending lost index integer:  8
.Sending lost index integer:  9
.Sending lost index integer:  10
.Sending lost index integer:  11
.Sending lost index integer:  13
.Sending lost index integer:  14
.Sending lost index integer:  16
.Sending lost index integer:  20
.lostpkt:  0 lostindex:  []
ok =  True packet number =   49  n_rcvd =    1  n_right =    1
uUif ACK pack
packet number is 20 Last packet:  False
ACK:  True Data packet:  False LOST_ACK False
ACK is: Correctly received the file with packet number: 20
lostpkt:  0 lostindex:  []
ok =  True packet number =   20  n_rcvd =    2  n_right =    2
```

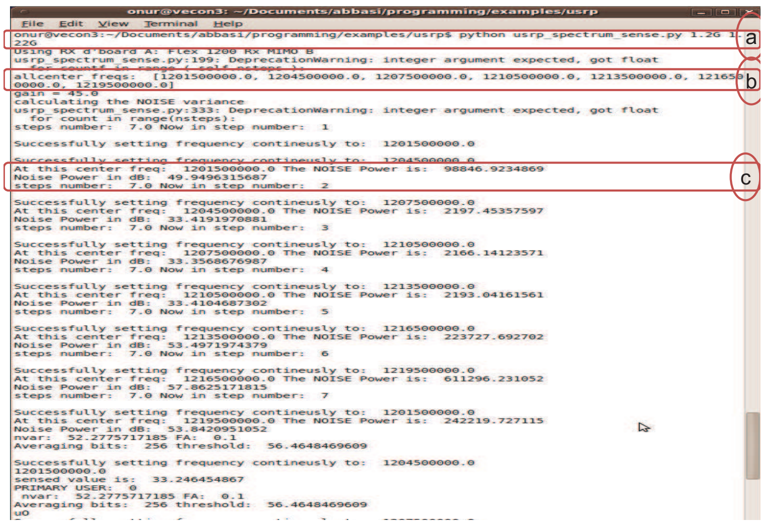
We show here how the transmitter receive the lost-ACK packet from the receiver, then start to retransmitting data to the receiver again. At the end receiver ACK.

Figure 8.6.9: Transceiver Results, Transmitter side with Lost Packets ACK (NACK)# 1.

ing samples. At last, we present the results of P_D versus received power (dBm).

Figure 8.6.10 shows the sensor in noise calibration mode. Some notes are needed here:

- a. This command start the sensing from 1.2 GHz to 1.22 GHz.


onur@vecon3: ~/Documents/abbasi/programming/examples/usrp
File Edit View Terminal Help
a
b
c

```
onur@vecon3:~/Documents/abbasi/programming/examples/usrp$ python usrp_spectrum_sense.py 1.2G 1.
22G
Using RX d'board A: Flex 1200 Rx MIMO B
usrp_spectrum_sense.py:199: DeprecationWarning: integer argument expected, got float
  for countf in range ( self.nsteps ):
allcenter_freqs:  [1201500000.0, 1204500000.0, 1207500000.0, 1210500000.0, 1213500000.0, 121650
0000.0, 1219500000.0]
gain = 45.0
calculating the NOISE variance
usrp_spectrum_sense.py:333: DeprecationWarning: integer argument expected, got float
  for count in range(nsteps):
steps number:  7.0 Now in step number:  1

Successfully setting frequency contineusly to:  1201500000.0

Successfully setting frequency contineusly to:  1204500000.0
At this center freq:  1201500000.0 The NOISE Power is:  98846.9234869
Noise Power in dB:  49.9496315687
steps number:  7.0 Now in step number:  2

Successfully setting frequency contineusly to:  1207500000.0
At this center freq:  1204500000.0 The NOISE Power is:  2197.45357597
Noise Power in dB:  33.4191970881
steps number:  7.0 Now in step number:  3

Successfully setting frequency contineusly to:  1210500000.0
At this center freq:  1207500000.0 The NOISE Power is:  2166.14123571
Noise Power in dB:  33.3568676987
steps number:  7.0 Now in step number:  4

Successfully setting frequency contineusly to:  1213500000.0
At this center freq:  1210500000.0 The NOISE Power is:  2193.04161561
Noise Power in dB:  33.4104687302
steps number:  7.0 Now in step number:  5

Successfully setting frequency contineusly to:  1216500000.0
At this center freq:  1213500000.0 The NOISE Power is:  223727.692702
Noise Power in dB:  53.4971974379
steps number:  7.0 Now in step number:  6

Successfully setting frequency contineusly to:  1219500000.0
At this center freq:  1216500000.0 The NOISE Power is:  611296.231052
Noise Power in dB:  57.8625171815
steps number:  7.0 Now in step number:  7

Successfully setting frequency contineusly to:  1201500000.0
At this center freq:  1219500000.0 The NOISE Power is:  242219.727115
Noise Power in dB:  53.8420951052
nvar:  52.2775717185 FA:  0.1
Averaging bits:  256 threshold:  56.4648469609

Successfully setting frequency contineusly to:  1204500000.0
1201500000.0
sensed value is:  33.246454867
PRIMARY USER:  0
 nvar:  52.2775717185 FA:  0.1
Averaging bits:  256 threshold:  56.4648469609
uO
```

1

Figure 8.6.10: Sensing Results / Noise calibration mode.

- b. This is an array of all the center frequencies which are going to be scanned.
- c. This is the stage where we measure the energy of the noise to get the noise variance value.

Figure 8.6.11 shows the sensing mode with no PU occupying the channels. It tunes the center frequency to scan the required channels.

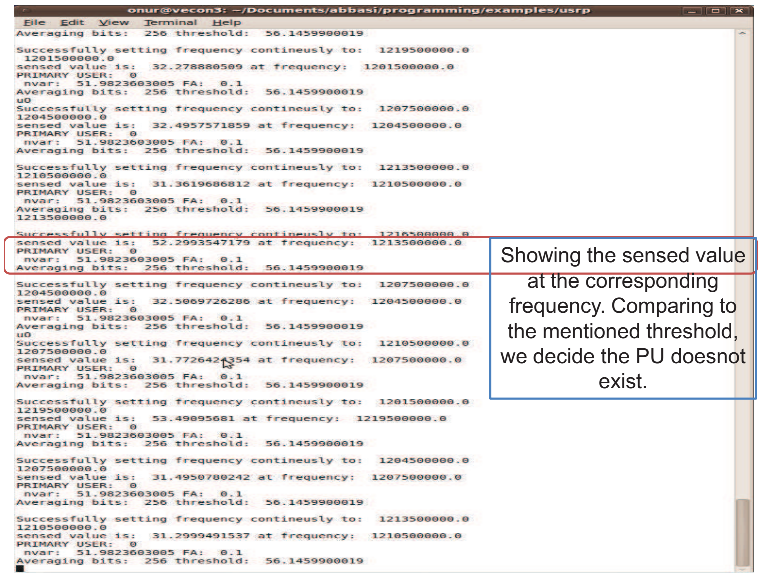
Figure 8.6.12 shows the sensing mode with PU occupying some channels and not occupying others.

Table 8.6.1: Sensor Parameters for Evaluating Sensing Time.

Parameter Name	**Value**
FFT size	5120/10240
Decimation rate	8
# Tuning steps for 100 MHz	17
sampling Freq	6Mbit

Considering the system parameters presented in Table 8.6.1, Fig.8.6.13 evaluates the total observation period (msec) versus the dwelling time (observation per one chunk). We can see how the total observation period increases by increasing the dwelling time for one chunk.

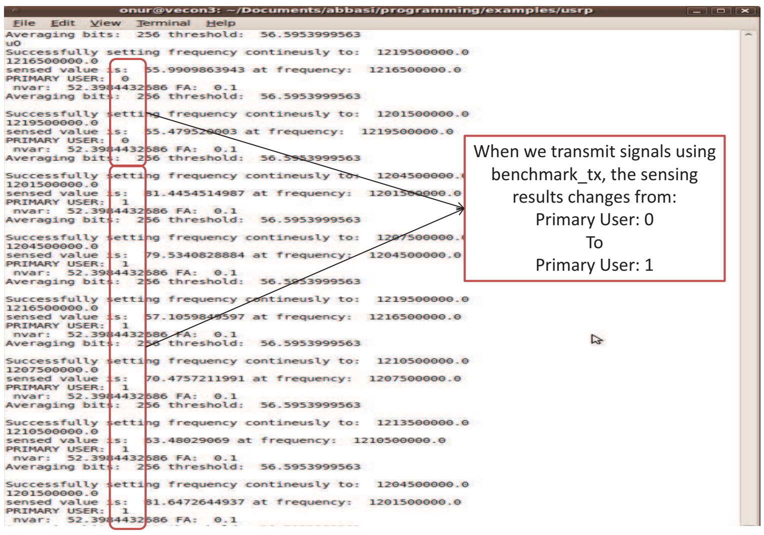
We changed the system parameters (for simplification) to be as shown in Table 8.6.2. Then, in Fig.8.6.14 we evaluate the system P_D versus the received power.


onur@vecon3: ~/Documents/abbasi/programming/examples/usrp
File Edit View Terminal Help

```
Averaging bits:  256 threshold:  56.1459900019

Successfully setting frequency contineusly to:  1219500000.0
 1201500000.0
sensed value is:  32.278880509 at frequency:  1201500000.0
PRIMARY USER:  0
 nvar:  51.9823603005 FA:  0.1
Averaging bits:  256 threshold:  56.1459900019
uO
Successfully setting frequency contineusly to:  1207500000.0
1204500000.0
sensed value is:  32.4957571859 at frequency:  1204500000.0
PRIMARY USER:  0
 nvar:  51.9823603005 FA:  0.1
Averaging bits:  256 threshold:  56.1459900019

Successfully setting frequency contineusly to:  1213500000.0
1210500000.0
sensed value is:  31.3619686812 at frequency:  1210500000.0
PRIMARY USER:  0
 nvar:  51.9823603005 FA:  0.1
Averaging bits:  256 threshold:  56.1459900019
1213500000.0

Successfully setting frequency contineusly to:  1216500000.0
sensed value is:  52.2993547179 at frequency:  1213500000.0
PRIMARY USER:  0
 nvar:  51.9823603005 FA:  0.1
Averaging bits:  256 threshold:  56.1459900019

Successfully setting frequency contineusly to:  1207500000.0
1204500000.0
sensed value is:  32.5069726286 at frequency:  1204500000.0
PRIMARY USER:  0
 nvar:  51.9823603005 FA:  0.1
Averaging bits:  256 threshold:  56.1459900019
uO
Successfully setting frequency contineusly to:  1210500000.0
1207500000.0
sensed value is:  31.7726424354 at frequency:  1207500000.0
PRIMARY USER:  0
 nvar:  51.9823603005 FA:  0.1
Averaging bits:  256 threshold:  56.1459900019

Successfully setting frequency contineusly to:  1201500000.0
1219500000.0
sensed value is:  53.49095681 at frequency:  1219500000.0
PRIMARY USER:  0
 nvar:  51.9823603005 FA:  0.1
Averaging bits:  256 threshold:  56.1459900019

Successfully setting frequency contineusly to:  1204500000.0
1207500000.0
sensed value is:  31.4950780242 at frequency:  1207500000.0
PRIMARY USER:  0
 nvar:  51.9823603005 FA:  0.1
Averaging bits:  256 threshold:  56.1459900019

Successfully setting frequency contineusly to:  1213500000.0
1210500000.0
sensed value is:  31.2999491537 at frequency:  1210500000.0
PRIMARY USER:  0
 nvar:  51.9823603005 FA:  0.1
Averaging bits:  256 threshold:  56.1459900019
█
```

Showing the sensed value at the corresponding frequency. Comparing to the mentioned threshold, we decide the PU doesnot exist.

Figure 8.6.11: Sensing Results / PU Detection with no PU signal occupying the channels.


onur@vecon3: ~/Documents/abbasi/programming/examples/usrp
File Edit View Terminal Help

```
Averaging bits:  256 threshold:  56.5953999563
uO
Successfully setting frequency contineusly to:  1219500000.0
1216500000.0
sensed value is:  55.9909863943 at frequency:  1216500000.0
PRIMARY USER:  0
 nvar:  52.3984432686 FA:  0.1
Averaging bits:  256 threshold:  56.5953999563

Successfully setting frequency contineusly to:  1201500000.0
1219500000.0
sensed value is:  55.479520003 at frequency:  1219500000.0
PRIMARY USER:  0
 nvar:  52.3984432686 FA:  0.1
Averaging bits:  256 threshold:  56.5953999563

Successfully setting frequency contineusly to:  1204500000.0
1201500000.0
sensed value is:  81.4454514987 at frequency:  1201500000.0
PRIMARY USER:  1
 nvar:  52.3984432686 FA:  0.1
Averaging bits:  256 threshold:  56.5953999563

Successfully setting frequency contineusly to:  1207500000.0
1204500000.0
sensed value is:  79.5340828884 at frequency:  1204500000.0
PRIMARY USER:  1
 nvar:  52.3984432686 FA:  0.1
Averaging bits:  256 threshold:  56.5953999563

Successfully setting frequency contineusly to:  1219500000.0
1216500000.0
sensed value is:  57.1059849597 at frequency:  1216500000.0
PRIMARY USER:  1
 nvar:  52.3984432686 FA:  0.1
Averaging bits:  256 threshold:  56.5953999563

Successfully setting frequency contineusly to:  1210500000.0
1207500000.0
sensed value is:  70.4757211991 at frequency:  1207500000.0
PRIMARY USER:  1
 nvar:  52.3984432686 FA:  0.1
Averaging bits:  256 threshold:  56.5953999563

Successfully setting frequency contineusly to:  1213500000.0
1210500000.0
sensed value is:  63.48029069 at frequency:  1210500000.0
PRIMARY USER:  1
 nvar:  52.3984432686 FA:  0.1
Averaging bits:  256 threshold:  56.5953999563

Successfully setting frequency contineusly to:  1204500000.0
1201500000.0
sensed value is:  81.6472644937 at frequency:  1201500000.0
PRIMARY USER:  1
 nvar:  52.3984432686 FA:  0.1
```

When we transmit signals using benchmark_tx, the sensing results changes from:
Primary User: 0
To
Primary User: 1

4

Figure 8.6.12: Sensing Results / PU Detection with PU signal occupying the some channels.

To be able to obtain an exact P_D we had to start by fixing the real P_{FA}. From Fig. 8.6.14 we confirm that for -86 dBm we get around 0.8 P_D. This means that we need more samples to improve the performance, and be able to detect -100 dBm

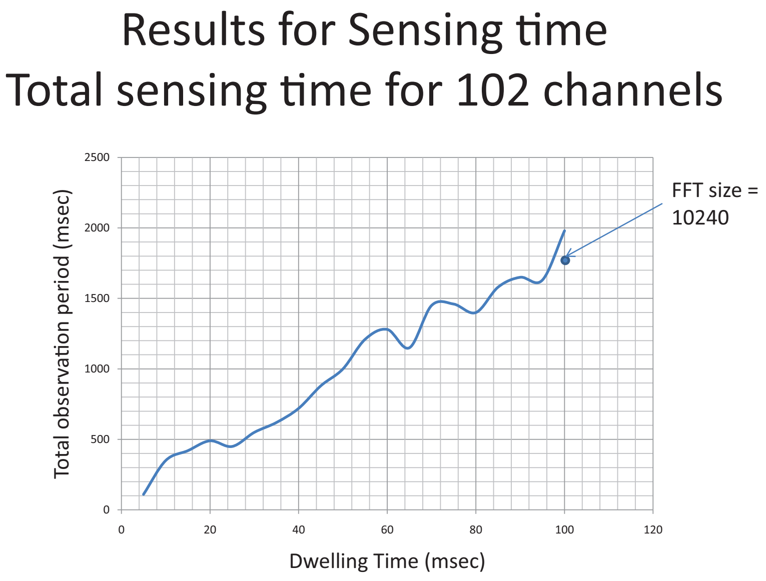


Figure 8.6.13: Sensing Results / Total Observation Time versus One Dwelling Time.

Table 8.6.2: Simulation Parameters for GNU Radio Based Sensor (P_D V.S Received Power).

Parameter Name	**Value**
# used antennas	1
Averaging samples	1000
Dewlling Time	7msec
# sent samples/one probability value	1000
Averaged samples for estimating the noise variance	50000
Actual P_{FA}	0.1
USRP Compensated Gain	45 dB
Bandwidth per chunk	250KHz
# of Sensed sub-bands	100
Sub-band Bandwidth	250KHz

signal level.

Figure 8.6.15 we evaluates the sensor sensitivity level versus the number of the averaged samples. Consider the Y-Axis values is the SNR (in - dB), it follows the same parameters of table 8.6.2 without fixing the real P_{FA}. We can confirm that

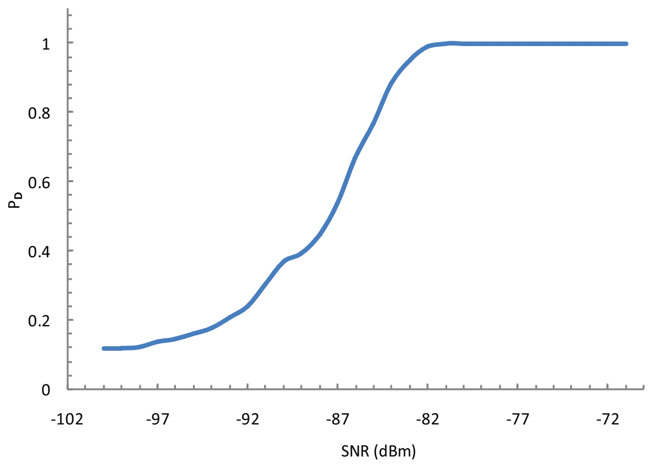


Figure 8.6.14: P_D versus received power dBm for GNU Radio based sensor.

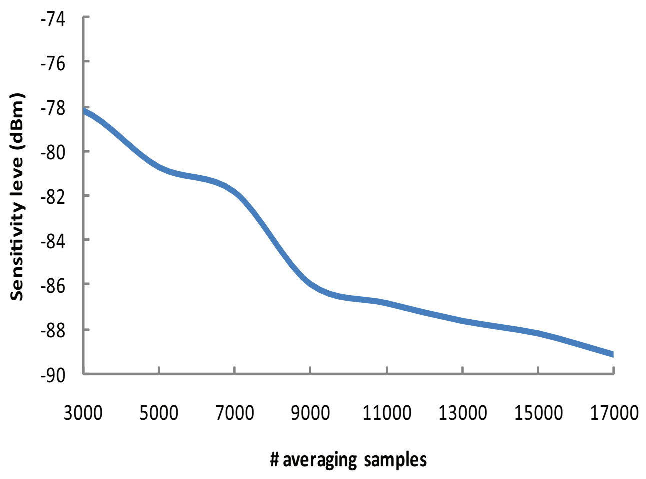


Figure 8.6.15: Sensing Results / USRP sensitivity levels versus # Averaging Samples

by increasing the number of averaging samples the sensitivity level decreases and therefore the sensor can sense more low level PU signal. This results is realistic since we gather more signal's energy. The fluctuating in the slope of this curve is caused by two factors:

- Not enough averaging time per sensed sub-channel.
- Not fixed real False Alarm probability.

Chapter 9

Conclusion

Finally, we conclude this thesis by pointing to the fact that we mostly achieve the objective of this research, which is using multi-antennas for blind spectrum sensing in cognitive radio. In this paper we proposed two novel schemes for primary user detection using multi-antennas. Using multi-antennas we achieve diversity plus combining gain, and utilizing two new ideas based on both EGC and MRC we managed to achieve blind detection. The first proposed scheme based on EGC, it selects equal gain antennas' weights, and we call it QUAL scheme. Using QUAL weights creates a problem of varying P_{FA}, because we select the maximum branch out of multiple (equal to the number of weights times number of antennas). Therefore, we had to derive a new threshold based on extreme value distribution to fix the P_{FA}. The second proposed scheme is by considering the fact that the eigenvectors of the received variance matrix are corresponding to the MRC weights. At the end, the simulation results show that the new proposed schemes achieve a diversity combining gains without the necessity of the channel information. After we finish the theoretical part of this research, and using a primary candidate for deploying SDR systems called GNU Radio, we implemented both communication and sensing SDR systems. We derived the sensitivity of the sensor versus the number of averaging samples and the SNR level.

Bibliography

[1] Theodore S. Rappaport, *Wireless Communication Principles and Practice* Prentice-Hall, 2002.

[2] D. Cabric, S.M. Mishra and R.W. Brodersen, “Implementation issues in spectrum sensing for cognitive radios,” in *Asilomar Conf. on Signals, Systems and Computers*, vol. 1, Nov. 2004, pp. 772 - 776.

[3] F.F. Digham, A.H. Tewfik and M.-S Alouini. “Detection of known and unknown signals over fading channels,” in *Proc. IEEE GLOBECOM*, vol. 3, Dec. 2003, pp. 1756 - 1760.

[4] F. F. Digham , M.-S. Alouini, and M. K. Simon, “On the energy detection of unknown signals over fading channels,” in *Proc. IEEE Int. Conf. Commun.*, Anchorage, AK, vol. 5, May 2003, pp. 3575 - 3579.

[5] N. M. Neihart, S. Roy and D. J. Allstot, ”A Parallel, Multi-Resolution Sensing Technique for Multiple Antenna Cognitive Radios”, ISCAS 2007.

[6] H. Urkowitz, “Energy detection of unknown deterministic signals,” in *Proc. of IEEE Journal*, vol. 55, no. 4, pp. 523 - 531, April 1967.

[7] M. K. Steven, *Fundamental of Statistical Signal Processing, Volume 2* Prentice-Hall, 1998.

[8] Ye Zhuan, G. Memik and J. Grosspietsch, “Energy detection using estimated noise variance for spectrum sensing in cognitive radio networks,” in *Proc. IEEE. WCNC*, April 2008, pp. 711 - 716.

[9] A. Goldsmith, *Wireless Communication*, Cambridge University Press, 2005

[10] K. Samuel and N. Salalees, *Extreme Value Distributions*, Theory and Applications, 2001, London.

[11] L.C. Godara, “Application of antenna arrays to mobile communication.II. Beam-forming and direction-of-arrival considerations,” in *Proc. of the IEEE Journal*, vol. 85, no. 8, pp.1195 - 1245, Aug. 1997.

[12] T. David and V. Pramod, *Fundamentals of Wireless Communication*, Cambridge University, 2005.

[13] S. Constantin and B. Steven, *Handbook on Advancements in Smart Antenna Technologies for Wireless Networks*, Information Science Reference (an imprint of IGI Global), Aug. 2008, ch. 1, pp. 1 - 32.

[14] A.R. Al-Abbasi and T. Fujii, “A novel spectrum sensing method using multi-antennas without channel state information,” in *Proc. IEEE ISWCS*, Sept. 2009, pp. 373 - 377.

[15] J. Ma, G. Ye Li and B. Hwang Juang, “Signal Processing in Cognitive Radio,” in *Proc. of the IEEE Journal*, vol. 97, no. 5, pp.805 - 823, May 2009.

[16] K. Nishimori, N. Kikuma, N. Inagaki “The Differential CMA Adaptive Array Antenna Using an Eigen-Beamspace System,” in *IEICE Trans. Comm.*, vol. E78-B, no. 11, pp. 1480 - 1488, Nov. 1995.

[17] C. Cordeiro, K. Challapali and D. Birru ”‘IEEE 802.22: An Introduction to the First Wireless Standard based on Cognitive Radios,”’ in *IEEE Journal of COMMUNICATIONS*, vol. 1, no. 1, pp. 38 - 47, April 2006

[18] http://gnuradio.org

[19] http://www.ettus.com/

Appendix A

Install GNU Radio

In this section we discuss the installation of GNU Radio. We will try to be as direct as possible to reach the target point, so it is advisable that the reader is familiar with Ubuntu system. There are two ways to install GNU Radio on your PC. [18]

- Download the packages from GNU Radio website and compile them. We are not explaining this method, because we did not use it in our installation.
- Using the already compiled binary package. In the following we are going to explain this method.

At the beginning we mention some pre-requisites packages needed to be taken care of before the installation of GNU Radio binary packages.

- Development Tools (need for compilation)
  - g++
  - subversion
  - make
  - autoconf, automake, libtool
  - sdcc (from ”universe”; 2.4 or newer)
  - guile (1.6 or newer)
  - ccache (not required, but recommended if you compile frequently)
- Libraries (need for runtime and for compilation)
  - python-dev
  - FFTW 3.X (fftw3, fftw3-dev)

  - cppunit (libcppunit and libcppunit-dev)
  - Boost 1.35 (or later)
  - libusb and libusb-dev
  - wxWidgets (wx-common) and wxPython (python-wxgtk2.8)
  - python-numpy (via python-numpy-ext) (for SVN on or after 2007-May-28)
  - ALSA (alsa-base, libasound2 and libasound2-dev)

- SWIG
- QWT

To install the needed packages you first update the local dpkg cache using the command

```
$ sudo apt-get update
```

We used Ubuntu-Karmic 9.10, then the installation using command line will be as

```
sudo apt-get -y install swig g++ automake libtool python-dev libfftw3-dev \
libcppunit-dev libboost1.38-dev libusb-dev fort77 sdcc sdcc-libraries \
libsdl1.2-dev python-wxgtk2.8 subversion git-core guile-1.8-dev \
libqt4-dev python-numpy ccache python-opengl libgsl0-dev \
python-cheetah python-lxml doxygen qt4-dev-tools \
libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools
```

Then we install QWT 5.0.2 using commands

```
$ sudo wget http://superb-east.dl.sourceforge.net/sourceforge/qwt/qwt-5.0.2.tar.bz2
$ sudo tar jxf qwt-5.0.2.tar.bz2
$ cd qwt-5.0.2
```

Be carefull that the source URL might change from time to time. Now edit qwt-config.pri. Change the unix version of ”INSTALLBASE” to ”/usr/local”. Change ”doc.path” to ”$$INSTALLBASE/doc/qwt”. Then,

```
$ sudo qmake
$ sudo make
$ sudo make install
$ cd ..
```

After that we install Boost package. After we download boost_1_37_0.tar.bz2 from boost.sourceforge.net and unrar it as we have done with QWT. After that, we follow the following commands

```
$ cd boost_1_37_0
$ BOOST_PREFIX=/opt/boost_1_37_0
$ ./configure –prefix=$BOOST_PREFIX –with-libraries=thread,date_time,program_options

$ sudo make
$ sudo make install
$ cd ..
```

Installation of GNU Radio:

For installing GNU Radio packages, we follow the binary package installation. At first please make sure that you un-installed any compiled GNU Radio packages using the following command

```
$ sudo make uninstall
```

Then, to track the stable release branch, insert the following in the deb manager.

```
deb http://gnuradio.org/ubuntu stable main

deb-src http://gnuradio.org/ubuntu stable main
```

To track the unstable release branch, insert the following.

```
deb http://gnuradio.org/ubuntu unstable main

deb-src http://gnuradio.org/ubuntu unstable main
```

Then update the package list using

```
$ sudo aptitude update
```

Finally, we Install the GNU Radio packages using,

```
$ sudo aptitude install gnuradio gnuradio-companion
```

At this point you will probably face many dependent packages. To solve this problem, use the package manager in your Ubuntu release (in our case it is Synaptic Package Manager) to un-install the conflict packages and install the required correct release packages.

Now GNU Radio is installed, however for a regular user to be able to use it freely we have to create a group for this user as following.

```
$ sudo addgroup <USERNAME>usrp
```

Appendix B

Publications

1. A. AL-ABBASI, T. FUJII "A Novel Spectrum Sensing Method using Multi-Antennas without Channel State Information,". IEICE. Technical Committee on Software Radio (SR), May.2009.

2. A. AL-ABBASI, T. FUJII. "A Novel Spectrum Sensing Method using Multi-Antennas without Channel State Information,". IEEE, International Symposium on Wireless Communication Systems. ISWCS, Sept. 2009.

3. A. AL-ABBASI, T. FUJII. "A Novel Blind Diversity Detection Scheme for Multi-antenna Cognitive Radio Spectrum Sensing,". IEEE 72nd Vehicular Technology Conference. VTC Sept. 2010 (accepted and to be published).

4. A. AL-ABBASI, O. Altintas, T. FUJII and other authors "Implementation and Evaluation of Distributed Control and Data Channel Coordination Algorithms for V2V Dynamic Spectrum Access," SDR'10 (accepted abstract, paper submitted).

Appendix C

Simulation Program

After the Acknowledgment page, the simulation program is attached.

Acknowledgment

I am deeply indebted to a number of individuals who assisted me all along the research presented in this paper was going on. To the following people, for their support beyond university activities, I owe what i don't know how to pay back:
- My supervisor, Associate Professor Takeo Fujii
- Professor Yasushi Yamao from AWCC
- Dr Suzuki Masahisa Mabo from the UEC international center
- Professor Yoshio Karasawa from AWCC
- The university staff

I would like to express my thanks to all the members of the Advanced Wireless Communication research center (AWCC) for their constant support in the laboratory and academic activities.

I wish to express my gratitude and appreciation to all of my lab members who always stood close to me and supported me.

Finally, I owe my deepest gratitude to my mother, father and family for their understanding and advices.