2019-12

IMPLEMENTATION OF PHYSICAL LAYER SECURITY FOR MULTIPLE-INPUT, MULTIPLE-OUTPUT COMMUNICATIONS VIA A WAVELET PACKET MODULATED SOFTWARE-DEFINED RADIO SYSTEM WITH NOISE MASKING

Baker, Kyle A.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMPLEMENTATION OF PHYSICAL LAYER SECURITY FOR MULTIPLE-INPUT, MULTIPLE-OUTPUT COMMUNICATIONS VIA A WAVELET PACKET MODULATED SOFTWARE-DEFINED RADIO SYSTEM WITH NOISE MASKING

by

Kyle A. Baker

December 2019

Thesis Advisor: Frank E. Kragh
Co-Advisor: Herschel H. Loomis

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
IMPLEMENTATION OF PHYSICAL LAYER SECURITY FOR MULTIPLE-INPUT, MULTIPLE-OUTPUT COMMUNICATIONS VIA A WAVELET PACKET MODULATED SOFTWARE-DEFINED RADIO SYSTEM WITH NOISE MASKING

5. FUNDING NUMBERS

6. AUTHOR(S) Kyle A. Baker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	**12b. DISTRIBUTION CODE**
Approved for public release. Distribution is unlimited.	A

13. ABSTRACT (maximum 200 words)

This thesis discusses the theory, design, and implementation of a physical layer security (PLS) system on a multiple-input, multiple-output (MIMO) software defined radio (SDR) with wavelet packet modulation (WPM). The PLS implementation leverages channel state information, gained via channel reciprocity, and applies it to an artificial noise masking algorithm formulated for MIMO SDR systems. The masking algorithm is realized with static elements as well as random elements that could prevent an adversary from estimating the additive mask components. Over-the-air testing provides evidence that the artificial noise mask is effective at obscuring a transmitted signal from an eavesdropper while remaining transparent to the intended receiver. Bit error rate calculations are performed for varied signal-to-mask ratio and LOS vs. BLOS conditions.

14. SUBJECT TERMS	**15. NUMBER OF PAGES**
software-defined radio, SDR, wavelet packet modulation, WPM, space-time block coding, STBC, Alamouti encoding, multiple-input multiple-output, MIMO, physical layer security, noise masking, channel state information, CSI, channel reciprocity	89
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

IMPLEMENTATION OF PHYSICAL LAYER SECURITY FOR MULTIPLE-INPUT, MULTIPLE-OUTPUT COMMUNICATIONS VIA A WAVELET PACKET MODULATED SOFTWARE-DEFINED RADIO SYSTEM WITH NOISE MASKING

Kyle A. Baker
Lieutenant Commander, United States Navy
BS, Pennsylvania State University, 2007
MEng, Pennsylvania State University, 2014

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2019

Approved by: Frank E. Kragh
Advisor

Herschel H. Loomis
Co-Advisor

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis discusses the theory, design, and implementation of a physical layer security (PLS) system on a multiple-input, multiple-output (MIMO) software defined radio (SDR) with wavelet packet modulation (WPM). The PLS implementation leverages channel state information, gained via channel reciprocity, and applies it to an artificial noise masking algorithm formulated for MIMO SDR systems. The masking algorithm is realized with static elements as well as random elements that could prevent an adversary from estimating the additive mask components. Over-the-air testing provides evidence that the artificial noise mask is effective at obscuring a transmitted signal from an eavesdropper while remaining transparent to the intended receiver. Bit error rate calculations are performed for varied signal-to-mask ratio and LOS vs. BLOS conditions.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ADC	Analog-To-Digital Conversion
ARQ	Automatic Repeat Request
BER	Bit Error Rate
BLOS	Beyond Line of Sight
BPSK	Binary Phase Shift Keying
CFO	Carrier Frequency Offset
CSI	Channel State Information
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DSP	Digital Signal Processing
DWT	Discrete Wavelet Transform
FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GPP	General Purpose Processor
GRC	GNU Radio Companion
GWN	GNU Radio Wireless Network
I/O	Input/Output
ICI	Inter-channel Interference
IDWT	Inverse Discrete Wavelet Transform
IQ	In-phase and quadrature
ISI	Intersymbol Interference
LAN	Local Area Network
LOS	Line-Of-Sight
MCM	Multicarrier Modulation
MIMO	Multiple-Input, Multiple-Output
MISO	Multiple-Input, Single-Output
MRC	Maximal-Ratio Combining
OTA	Over-The-Air
PAPR	Peak-to-Average Power Ratio

PLS	Physical Layer Security
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
SCA	Software Communications Architecture
SMR	Signal-to-Mask Ratio
SNR	Signal-to-Noise Ratio
USRP	Universal Software Radio Peripheral
WPM	Wavelet Packet Modulation

ACKNOWLEDGMENTS

To my wife, Julie, and my daughter, Magdalene, thank you both for your love, laughter, and impeccable timing.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Information security has always been an important topic in warfare, with ancient strategists such as Sun Tzu describing it in terms of spying and obscuring evidence of a commander's true intentions [1]. While human espionage and military deception are still important to information security, the advance of digital communication technologies have opened up additional avenues for an adversary to gain and exploit privileged communications. The 2019 National Defense Strategy cites information security as a continued concern in the modern military as information warfare allows adversaries to disrupt operations on and off the battlefield [2].

Wireless communications present enemy forces with an opportunity to intercept transmitted messages, without the sender knowing and without being in physical proximity to the intended recipient [3]. Fortunately, information security techniques have been developed to secure wireless information and inhibit unauthorized retrieval. In general, information security can be explained in terms of confidentiality, integrity, availability, non-repudiation, and authentication (CIANA) as follows [4]:

- Confidentiality – Access to information for authorized personnel only
- Integrity – Information remains uncorrupted and usable
- Availability – System, and its information, remain accessible when needed
- Nonrepudiation – Creation of an audit trail for all communications
- Authentication – Verification of sender and receiver identities

While all aspects of CIANA are important, this thesis is primarily concerned with the aspect of confidentiality. One common method of preserving and protecting information from an opposition force is cryptography, an application of mathematics and logic to encrypt data, preventing unauthorized access [5]. Cryptography relies on the assumption that it is computationally expensive for an eavesdropper to decode an encrypted

signal. This technique is being challenged by recent innovations such as quantum computing [6].

One additional confidentiality technique that could be used instead of—or in conjunction with—cryptography is the practice of Physical Layer Security (PLS) [6]. The fundamental premise of PLS is the implementation of communication security through exploitation of the physical properties of the communication channel [7]. The benefit here is that while cryptography requires the adversary to have a computational limitation when attempting to decode an acquired signal, PLS directly affects an adversary's ability to receive the signal in the first place [7]. Additionally, each technique can be transparent to each other allowing the practice of defense-in-depth (i.e., a multilayered protection system) to be applied [8]. One method to employ PLS is through the use of an artificial noise mask that degrades the eavesdropper acquired signal [3]. In this case, the noise mask artificially increases the background noise level of an eavesdropper, thereby reducing its signal-to-noise ratio (SNR), which can prevent message reception [9]. The secure PLS radio communications system, described herein, provides confidentiality by undermining the data integrity of eavesdroppers.

B. OBJECTIVE

The objective of this research was to implement the artificial noise mask outlined in [3] and [10] on the radio communications system created in [11]. This new secure communications system contains all the key features of the original radio such as Wavelet Packet Modulation (WPM), Multiple-Input, Multiple-Output (MIMO) antenna configuration, Alamouti Encoding, and Software Defined Radio (SDR) flexibility along with the added benefits of concealed communication and flow control. The radio has been subjected to over-the-air (OTA) testing in order to investigate the impact of the added artificial noise masking technique in a real world environment.

C. THESIS CHAPTER ORGANIZATION

This thesis begins with two background chapters covering major areas of interest for this research. A background on communications systems is given in Chapter II, with a focus on applicable architectures, Multicarrier Modulation (MCM) techniques, MIMO

systems, Communications Channels, and SDR. Chapter III provides a brief introduction to Medium Access Control (MAC), flow control techniques, and highlights their usage in SDR applications. Advanced communications engineers familiar with GNU Radio may wish to skip Chapter II and III.

In Chapter IV, the fundamental previous work accomplished in [3], [10], and [11] will be examined in order to better explain the implementation of the new secure radio communications system in Chapter V. The results of OTA testing will be demonstrated in Chapter VI followed by a summary of the research accomplished, conclusions, and recommendations for future improvements in Chapter VII.

THIS PAGE INTENTIONALLY LEFT BLANK

II. COMMUNICATION SYSTEMS

A. BASELINE MODELS

In order to convey complex ideas or describe intricate systems it is helpful to ensure that a common vernacular is used and that there is a basic understanding of the key concepts applicable to the situation at hand. In support of this effort, it is often useful to employ conceptual models as an abstraction of a real world item or process [12]. Two important models for this research, both of which highlight relevant aspects of communications systems, are the Open Systems Interconnection (OSI) Model and the Digital Communications Model [13]–[15].

1. OSI Model

Reference [13] notes that the International Standards Organization developed the OSI model in order to make solving data transmission problems more manageable. This was accomplished through decomposition of the overall communication architecture into seven separate parts, commonly referred to as the OSI model layers. The model addressed the need for a common parlance and reference for protocol development. This enabled systems that conform to the associated family of standards and protocols for the model to be capable of communication interoperability. In support of this, a communications engineer will need to ensure that each terminal involved in a communication system is utilizing similar protocols at each layer of the model as shown in Figure 1. Each layer further down the model essentially encapsulates the information from the layer above it within applicable protocols that need to be unwrapped at the distant end, layer by layer [13]. Normally communication flows from the user at the highest level, the application layer, down through all applicable layers until it reaches the physical layer where the actual signal transmission takes place. At the receiver this signal is acquired at the physical layer and is decoded by protocols at each higher level until it has reached the distant end user or process. An important concept is that at each layer, the protocols are independent, ideally allowing changes to protocols at each level to be transparent to its neighboring layers [16]. While the terminology used in the OSI model is typically

introduced for networked systems, the basic ideas and concepts are extensible for communication systems as a whole. Reference [17] notes that outside of simple point-to-point systems, most modern radios interface at the network or application layers of the OSI model.

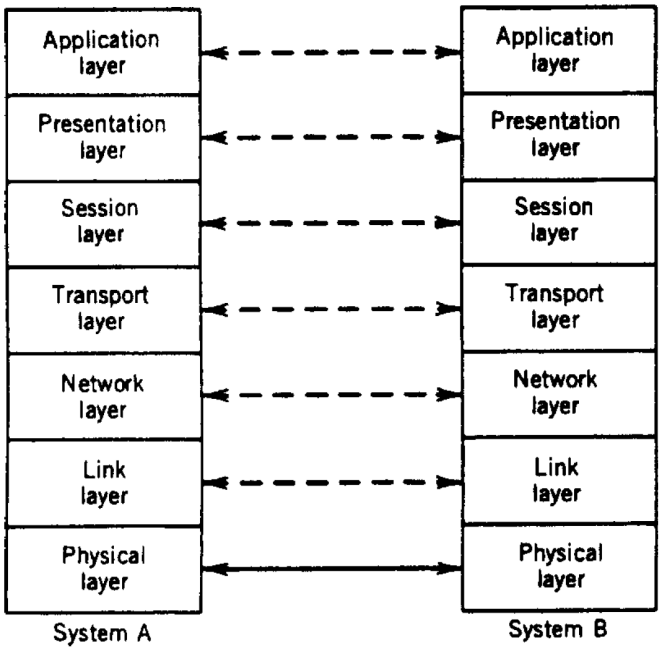

Figure 1. The OSI Model. Source: [13].

a. Physical Layer

The physical layer of the OSI model, also known as layer 1, is responsible for the electrical and mechanical interfaces that constitute the physical connectivity between two communication terminals. Major functional areas of this layer include interface characteristics, how bits are represented (or modulated), transmission rate, and synchronization. Additionally, the physical layer is concerned with the link configuration, physical topology, and transmission mode [14].

The physical layer serves as the input/output (I/O) interface to the entire communication system; receiving data from the distant end and providing information to the layer above it or transmitting messages that flow down from higher levels. Mechanical

interfaces are common in systems which utilize cabling and connectors, like a wired Local Area Network (LAN), whereas a wireless system may employ an electromagnetic interface via antenna transmission and reception [13].

b. Data Link Layer

The next layer in the model, also known as layer 2, is the data link layer. This layer is responsible for the procedures and coordination for directly communicating terminals. Major functional areas of the data link layer include framing, physical addressing, and error control. Another two aspects, which were of particular importance to this thesis research, are flow control and control of access to a shared medium [14].

The overarching theme for this layer is reliable communication exchanges using connections established by the physical layer. The protocols at this layer typically provide rules which mediate communication between terminals trying to access a shared channel. Further discussion on this topic is featured in III.

c. Higher Layers

The upper layers of the OSI model, in ascending order, are the network, transport, session, presentation, and application layers. Each one houses a family of protocols and provides distinct functionality. While the majority of the research conducted in this thesis focused on layers 1 and 2, it should be noted that future work could modify the radio design to address the other layers. It should also be noted that in some communications architectures, such as Transport Control Protocol/Internet Protocol (TCP/IP), the upper three layers are combined into the application layer [14]. A brief summary of layers 3 through 7, per [13], [16], is provided below:

- Network (Layer 3) – As the name suggests, this layer is primarily concerned with the end-to-end communication required to route data through a network. Some specific functions are logical addressing and routing. At this layer data is grouped into units called packets.

- Transport (Layer 4) – This layer is responsible for overall quality-of-service in support of the upper layers of the OSI model. Some specific functions include segmentation, reassembly, and connection management.
- Session (Layer 5) – This layer provides for communication session creation, synchronization, and teardown for end user applications and processes.
- Presentation (Layer 6) – This layer involves data interpretation, formatting, and syntax in support of application layer functions.
- Application (Layer 7) – The application layer is the highest layer in the OSI model. It provides an interface point for user applications and processes to utilize the communication system.

2. Digital Communications Model

Much like how the OSI model provided a baseline for the overall communications process, the Digital Communications Model can be used to describe communication elements specific to digital data transmission. These elements encompass key parameters and decisions involved in the design of a radio system. An example of a typical Digital Communications Model is featured in Figure 2.

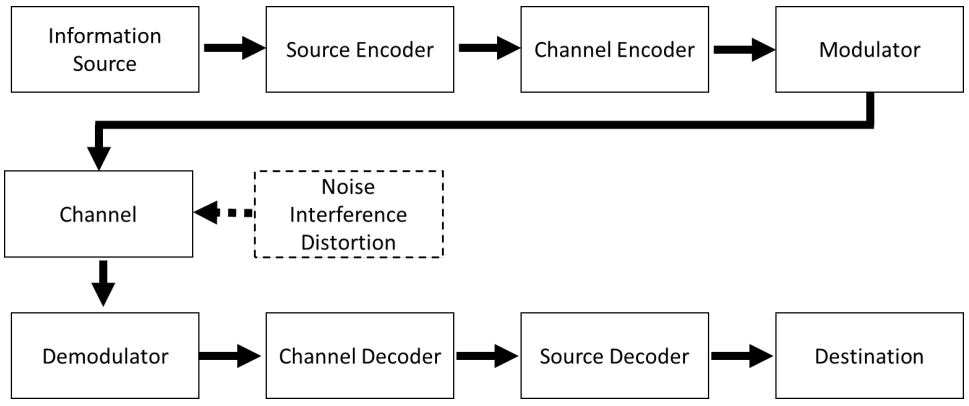

Figure 2. Digital Communications Model. Adapted from [15].

In Figure 2 the top row represents the transmitter elements (i.e., the transmit path), the middle section accounts for the effects of the transmission channel, and the bottom row represents the receiver elements (i.e., the receive path). As demonstrated in the figure, the elements of the model are typically mirrored between the transmitter and the receiver. That is to say, the signal processing transformations that the data or signal undergo at the transmitter are generally reversed at the receiver [16]. As such, most of the following descriptions will occur in pairs.

a. Information Source and Destination

The information source can come in many varieties and it depends on the particular communication system being examined and its specific purpose. It could be analog or digital, human or machine generated, among a myriad of other options [15]. For example, a telephony system could take in a human voice as its information source whereas the input to a satellite telemetry, tracking, and commanding system may be a stream of digital ephemeris data. The destination is the sink where the communication receiver transfers received information for further use, for instance, a telephone handset or storage media.

b. Source Encoder and Decoder

The output of the information source is the input to a source encoder which creates an information sequence. This information sequence is a digital representation of the source data [15]. In some cases, like voice transmission, analog-to-digital conversion (ADC) is needed. In other instances, digital data will need to be transformed into a different digital format for transmission. The decoder in the receiver will reverse the source encoder process, as needed.

c. Channel Encoder and Decoder

The information sequence generated by the source encoder is delivered to the channel encoder to improve reliability and decoding success via controlled redundancy. This process of channel coding introduces extra bits into the information sequence, creating codewords for error detection and, in some cases, error correction [18]. The decoder seeks

to reverse the encoding process and produce an estimate of the transmitted information sequence [15].

d. Modulator and Demodulator

The modulation process takes into account the features and characteristics of the expected transmission channel in order to produce an appropriate waveform or modulated signal. The modulator element creates the modulated signal or symbols which contains the message information. The demodulator in the receive path reverses this operation as the first step in message recovery [18].

e. Channel

The channel is the medium over which the signal is transmitted after modulation. In wired systems this could be an Ethernet cable or optical fiber and in wireless systems the channel may be air, water, or even the vacuum of space. Each of the potential media over which digital communication signals are transmitted are subject to various types of disturbances [15]. These will be briefly discussed in Section B.

B. COMMUNICATIONS CHANNEL

The middle row of Figure 2 shows a set of inputs coming from a block made with a dotted-line. These represent the primary causes of disturbance to the transmitted signal while traversing the communications channel. The channel subjects signals to noise, interference, and distortion. Noise occurs due to natural, random, and unpredictable electromagnetic phenomena. The sources of these electric signals can be either internal or external to the digital communications system. Interference, unlike noise, is an electric signal produced by a man-made source that impacts the communication signal waveform. Lastly, distortion is related to the phenomena by which the characteristics of the channel itself alter the transmitted signal. When this occurs the signal acquired by the receiver becomes inflicted with perturbations that may or may not be correctable with an equalizer [18].

1. Fading Channels

One particular set of channel distortions is commonly characterized by what is known as a fading channel. Fading is an environmental effect in which the transmitted signal magnitude or phase is changed due to time-varying and space-varying conditions [13]. These circumstances encompass, but are not limited to, two main categories described by Doppler effects and multipath effects. Doppler effects are related to the relative motion between transmitter and receiver whereas multipath effects occur due to signal reflection and diffraction from physical objects in the environment. In the multipath situation the received signal may not be dominated by a line-of-sight (LOS) main beam transmission. This can lead to negative effects in digital systems such as intersymbol interference (ISI), where energy from one symbol is acquired at the receiver during the succeeding symbols time period [19]. This can lead to bit decoding error in the receive path [13].

Fades can vary in extent, period, and frequency and they are affected by numerous facets of the environment including path length, transmission frequency, weather, and local topography [13]. Furthermore, fading channels can be subdivided into four classes [19]:

- Slow Fading, featuring random magnitude and phase changes. Here the coherence time, or timeframe over which transmission phase is relatively predictable, is greater than the channel estimation time.
- Fast Fading, featuring random magnitude and appreciable phase error. Here the coherence time is shorter than the channel estimation time.
- Flat Fading, featuring random magnitude and negligible ISI. Here the coherence bandwidth, or frequency range over which similar amplitude effects from fading is experienced, is greater than the signal bandwidth.
- Frequency Selective Fading, featuring random magnitude and appreciable ISI. Here the coherence bandwidth is less than the signal bandwidth.

By definition, it can be seen that slow and fast fading are related to time-varying effects while flat and frequency selective fading are driven by space-varying effects. While

fading is never ideal, the impact of slow and flat fading channels can be more easily mitigated, potentially allowing greater communication reliability in their presence. Additionally, it should be noted that in some communications schemes, such as Long Term Evolution (LTE), each user's unique fading channel is leveraged in order to increase channel capacity[20].

2. CSI Estimation via Training Sequences

In order to counteract the effects of a fading channel, a communications receiver can measure the effects of the channel on the received symbols. These effects are quantified into what is known as Channel State Information (CSI). One well known technique by which the CSI can be estimated is the use of training symbols. The transmitter transmits known symbols to the receiver that can be used to estimate the channel impact on the magnitude and phase rotation of these training symbols. The estimation obtained from this process is used to correct, or equalize, the effects of the transmission channel. In some cases, where the communication system has multiple channels, an entire channel may be used to transmit a training subchannel, or pilot carrier, which is used solely for channel estimation [21].

3. Channel Reciprocity

While training symbols and training subcarriers are major techniques by which equalization occurs at the receiver, they are far from the only methodology. Refs [22], [23] cite channel reciprocity as another important method by which the channel state can be estimated for subsequent equalization. Channel reciprocity essentially states that for a given frequency, the signal perturbations across a channel will be identical in both directions of communication. This principle can allow the transmitter to compensate for the channel affects prior to signal transmission, potentially making the channel estimation and equalization process at the receiver unnecessary if its own transmission has already been characterized at the transmitter. Note that this process is subject to time and space variations in the channel.

C. MULTICARRIER MODULATION

The use of MCM has been espoused for decades due to its numerous benefits [24]. Splitting a single carrier over several subcarriers allows a transmitter to keep the same overall data transmission rate while still having each individual subcarrier run at a slower symbol rate. This lengthens the overall symbol duration which allows each individual subcarrier to be more resistant to ISI caused by fading effects [24].

1. OFDM

One prolific form of MCM is Orthogonal Frequency Division Multiplexing (OFDM). In this modulation scheme, the Inverse Fourier Transform is used to create several orthogonal bases, or subcarriers, upon which the data is modulated. In order to demodulate, a Forward Fourier Transform is utilized [25], [26].

2. WPM

While OFDM is extremely popular and finds usage in cellular phones, internet standards, and general communication practices, it is not the only MCM available for use. In recent years, Wavelet Packet Modulation (WPM) has seen increased experimentation [27], [28]. WPM is based on the idea of using short compact support waveforms, called wavelets, to create orthogonal bases and is analogous in application to OFDM. Much like how the Fourier Transform is used for Frequency Domain Analysis, the Wavelet transform is used for Wavelet Analysis, usually in Multiresolution Analysis of communications signals or imagery [29]. Similar to OFDM, the Inverse Discrete Wavelet Transform (IDWT) can be used for modulating a signal while a forward Discrete Wavelet Transform (DWT) can used for demodulation in a MCM system [3].

a. Advantages of WPM over OFDM

WPM affords additional flexibility in signal construction and some useful advantages over OFDM in the area of spectrum usage. It has been demonstrated to be less sensitive to offsets in frequency and timing than OFDM. Additionally, various studies have found that OFDM has a larger Peak-to-Average Power (PAPR) ratio than WPM [30]. This indicates that in radios which employ an amplifier, an example being a satellite with a

Traveling Wave Tube Amplifier (TWTA), a WPM modulated system would potentially require less power back off to stay in the linear operating region of the amplifier. That is to say, WPM allows the amplifier to operate at a higher average power output without exceeding its peak power limitations. These advantages could result in less signal distortion overall. Furthermore, [28] noted that WPM is useful in certain situations, citing an impulsive time domain interferer operating simultaneously with narrowband jammer as an example.

The WPM modulation system is adaptable in the frequency domain as well as localized in time whereas OFDM employs sinusoidal bases which are not localized in the time domain. As such, [31] notes the following additional benefits of WPM over OFDM:

- Customizable transceiver characteristics that may better adapt to channel conditions to reduce ISI and inter-carrier interference (ICI).
- Potentially lower power requirements due to increased tolerance towards time and frequency offsets from interference.
- Greater bandwidth efficiency due to no requirement for a cyclic prefix.

While WPM does have some advantages over OFDM, it can still suffer from the general effects of carrier offset and phase noise. Refs [32]–[34], among others, begin to address the problems with carrier frequency offset and phase noise.

b. Mathematical Basis

In [27], [28], Allan Linsey notes the mathematical specifics of forming a set of orthogonal wavelet bases for a communication system. References [31], [35] provide (1)through (3) to serve as an example of wavelet base creation while Figure 3 visually depicts orthogonal bases created by those equations. Figure 4 and Figure 5 demonstrate a wavelet filter bank topology that could be employed in order to implement (1) through (3).

$$g[n] = -1^{n} h^{*}[2N-1-n] \tag{1}$$

$$h'[n] = h^{*}[2N-1-n] \tag{2}$$

$$g'[n] = (-1)^{n+1} h^*[n] \tag{3}$$

In these equations, a simple base of [1 1], is defined as the scaling function, h. This allows the wavelet function, g, to be derived. The scaling function and wavelet function can then be utilized in (2) and (3) to obtain the scaling function and wavelet inverses, h' and g', respectively. These are the basic Haar Wavelet and Scaling Functions as collected in (4) [31], [35].

$$h = [1 \quad 1], g = [1 \quad -1], h' = [1 \quad 1], g' = [-1 \quad 1] \tag{4}$$

The orthogonal carriers from (4), produced in accordance with the procedures from [28], are shown in Figure 3.

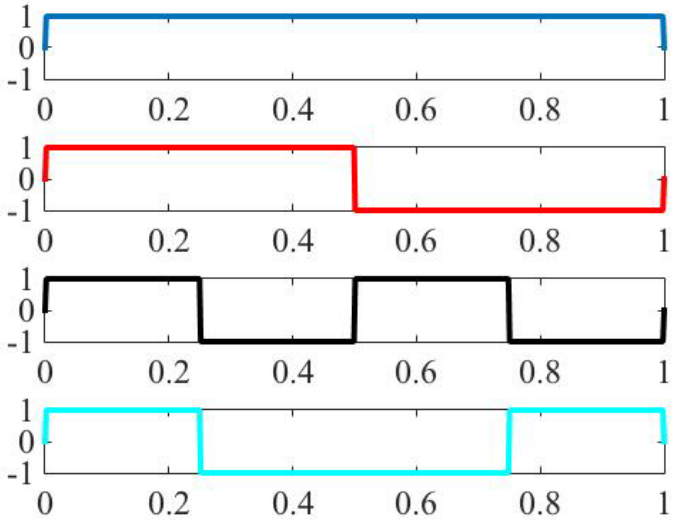

Figure 3. Wavelet Subcarriers. Source: [36].

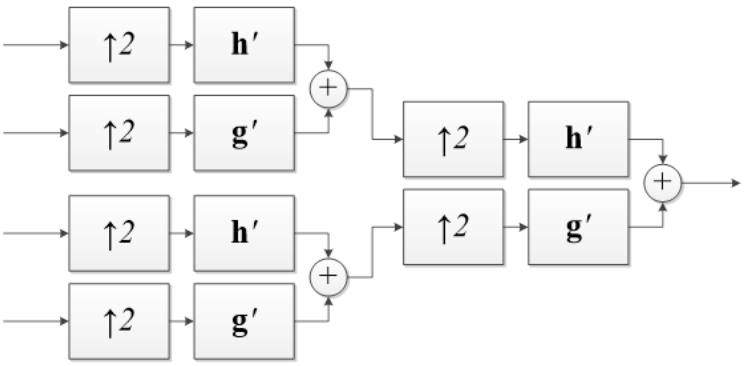

Figure 4. IDWT Filter Bank. Source: [11].

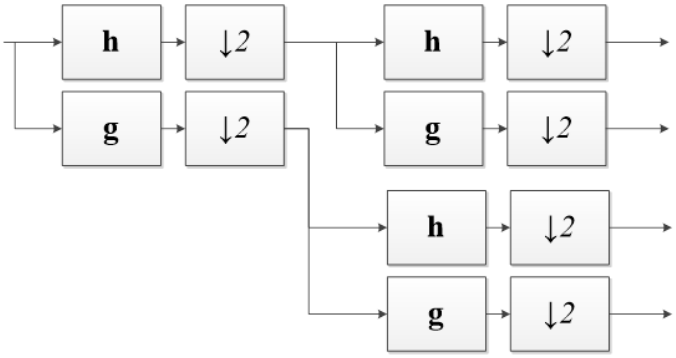

Figure 5. DWT Filter Bank. Source: [11].

Note that the wavelet packet transform may have a multitude of other base scaling functions but for the remainder of this thesis only the Haar wavelet will be considered.

D. MIMO EMPLOYMENT

1. Diversity Overview

Diversity techniques aid communication systems in combating the ill effects of multipath fading channels. Communications diversity comes in many forms including, but not limited to; pattern, polarization, temporal, spatial, and frequency [37]. This thesis research deals with the latter three techniques in particular. In temporal diversity, a symbol could be transmitted at multiple time intervals. The advantage of this is that one time interval may have a more favorable channel due to time-varying effects. With frequency diversity, a symbol could be transmitted on multiple frequencies for a comparable effect. Similarly, spatial diversity involves transmitting a symbol along separate transmission

paths. Each of these techniques essentially seeks to overcome distortion through redundancy via the employment of multiple transmission channels [37]. Two forms of multiple antenna diversity setups are pictured in Figure 6 and Figure 7. Figure 6 depicts a 2x1 Multiple-Input, Single-Output (MISO) configuration with two transmit antennas and a single receive antenna. This configuration has two inherent transmit paths for transmission diversity. Figure 7 portrays a 2x2 MIMO configuration with two transmit antennas and two receive antennas. This configuration features 4 transmit paths for increased diversity over the MISO case.

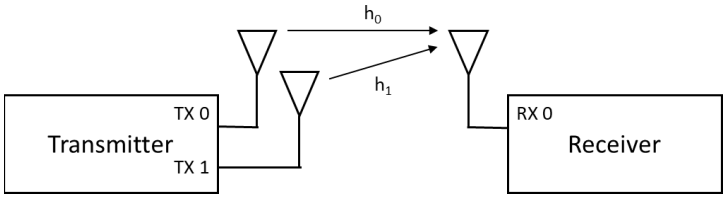

Figure 6. A 2x1 MISO Antenna Configuration

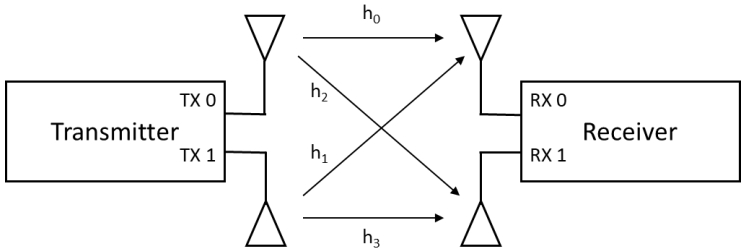

Figure 7. A 2x2 MIMO Antenna Configuration

2. Alamouti STBC Encoding

In order to better leverage the multiple antenna setup, the practice of Alamouti Encoding is commonly employed in MIMO systems. Alamouti Encoding takes the input symbol stream and creates a space-time block code (STBC) [37]. The STBC further increases redundancy by improving the spatial diversity of a multiple antenna setup and adding temporal diversity gains [19]. This scheme is portrayed in Table 1 and Table 2.

Table 1. Channel State Variables Corresponding to Figure 7. Adapted from [3], [11].

	RX Antenna 0	RX Antenna 1
TX Antenna 0	h_0	h_2
TX Antenna 1	h_1	h_3

Table 2. Transmitted Symbols. Adapted from [3], [11].

Time Step	TX Antenna 0	TX Antena 1
N	s_N	s_{N+1}
N+1	$-s^*_{N+1}$	s^*_N

This table demonstrates the symbols transmitted as part of a two branch transmit diversity Alamouti Encoding Scheme [11].

Table 1 portrays how the channel state variables utilized in this thesis correspond to the channel paths between each transmit and receive antenna pair. Table 2 shows the symbol encoding scheme utilized in an Alamouti Space Time Block Code (STBC) as depicted in Figure 8.

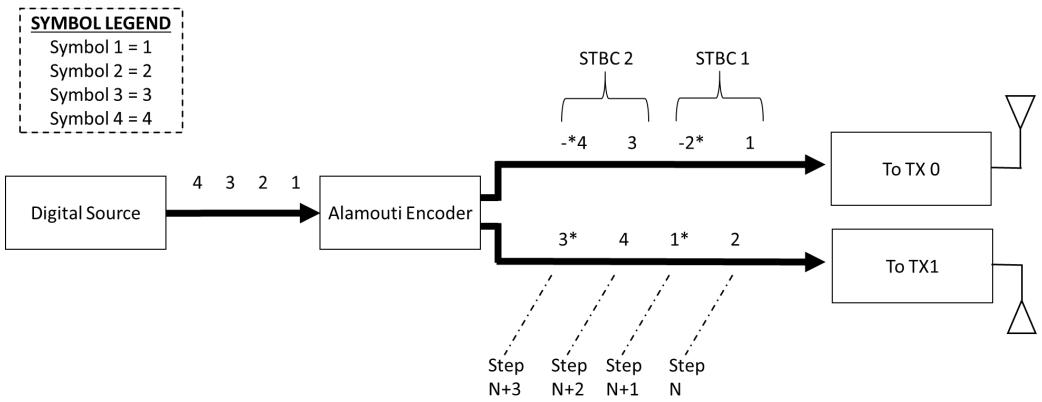

Figure 8. Visual Representation of the Alamouti STBC Encoding Scheme. Adapted from [3], [19].

In Figure 8, we can see a data stream of input symbols consisting of "1, 2, 3, 4", with '1' arriving first at the encoder and '4' arriving last. The encoder applies the schema as shown in Table 2 to these incoming symbols. Every two symbols are encoded into a single STBC. After encoding the outputs are as follows:

- At time Step N (the first time step), transmit antenna zero will send symbol one and transmit antenna one send symbol two.
- At time Step N+1, transmit antenna zero will send the negative conjugate of symbol two and transmit antenna one send the conjugate of symbol one.

3. Alamouti STBC Decoding

In the receive path, channel estimation and equalization is made possible through the use of known training sequences that are transmitted along with the message from the information source. The four transmissions that are received as part of the STBC are shown in (5) and the nomenclature for the received symbols are delineated in Table 3. Note that (5) does not show the external environmental and internal electronic noise that gets added to the signal during the communication process.

Table 3. Received Symbols. Adapted from [3], [11].

Time Step	RX Antenna 0	RX Antenna 1
N	r_0	r_2
N+1	r_1	r_3

$$\begin{aligned} r_0 &= h_0 s_N + h_1 s_{N+1} \\ r_1 &= -h_0 s^*_{N+1} + h_1 s^*_N \\ r_2 &= h_2 s_N + h_3 s_{N+1} \\ r_3 &= -h_2 s^*_{N+1} + h_3 s^*_N \end{aligned} \tag{5}$$

The training symbols in (5), s_N and s_{N+1} are used to estimate the channel state and then decoded into the decision statics of (6) and (7). These decision statics, x_N and x_{N+1}, are then compared to the set of possible symbols in order to determine the actual sent symbols for the message portion of the transmission. It is assumed that the CSI estimates are constant within both steps of a single STBC.

$$x_N = \frac{h_0^* r_0 + h_1 r_1^*}{|h_0|^2 + |h_1|^2} = \frac{h_2^* r_2 + h_3 r_3^*}{|h_2|^2 + |h_3|^2} \tag{6}$$

$$x_{N+1} = \frac{h_1^* r_0 - h_0 r_1^*}{|h_0|^2 + |h_1|^2} = \frac{h_3^* r_2 - h_2 r_3^*}{|h_2|^2 + |h_3|^2} \tag{7}$$

E. SOFTWARE DEFINED RADIO

1. Overview

The models, theory, equations, and techniques presented in this chapter thus far are applicable to a wide range of radio communication systems. One category of digital communication system of great importance to this thesis is that of the SDR. Conventional communication systems typically utilize discrete circuits and application specific integrated circuits (ASICs) to implement the elements of the digital communications model shown in Figure 7. This creates a static radio design that may be well optimized for a specific environment or function but may not lend itself to operational flexibility, dynamic

interoperability, rapid prototyping, and future modification [17]. SDRs have the distinct advantage of allocating radio characteristics, like modulation and coding schemes, to software execution rather than hardware implementation [11], [18]. This allows for variable functionality and re-configurability without the need for hardware modifications or the manufacture of multiple design architectures. While [17] points out that the exact definition of an SDR is contentious, it does note a defining characteristic is the flexibility to reprogram an SDR for functionality that was not initially intended or designed.

Peripheral hardware is still required for items such as low noise amplifiers, conversion between analog and digital signals, frequency mixing, and antenna elements [11]. These are typically implemented via a radio frequency (RF) front end with signal processing provided by a general purpose processor (GPP) or field programmable gate array (FPGA) [17]. As [18] notes, the radio design is commonly created in a high level software language or development environment and then converted to a hardware description language to run on the signal processing device. Some notable software frameworks are LabVIEW, MATLAB, SIMULINK, GNU Radio, and the Software Communications Architecture (SCA) [17], [38]. A depiction of this is presented in Figure 9.

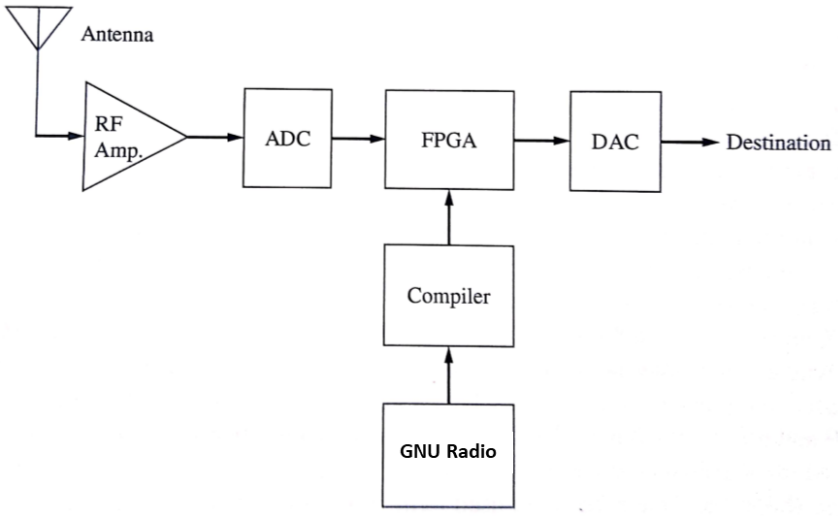

Figure 9. SDR receiver implemented on an FPGA with GNU Radio code. Adapted from [18].

2. GNU Radio

GNU Radio, a free and open-source software package, was selected to be the SDR development toolkit used in this thesis. The GNU Radio main components are its runtime environment, block class structure, and the runtime scheduler [39]. GNU Radio Companion (GRC) provides a graphical user interface (GUI) in which the signal processing blocks are arranged into a visual flow graph. These GRC flow graph blocks contain digital signal processing (DSP) modules and low-level algorithms needed for signal processing applications [40]. An example flow graph for a WPM MIMO SDR is shown in Figure 10. It contains blocks which take in a text file, perform source and channel encoding, apply Alamouti Encoding and the IDWT, and eventually send the output to the USRP Sink block.

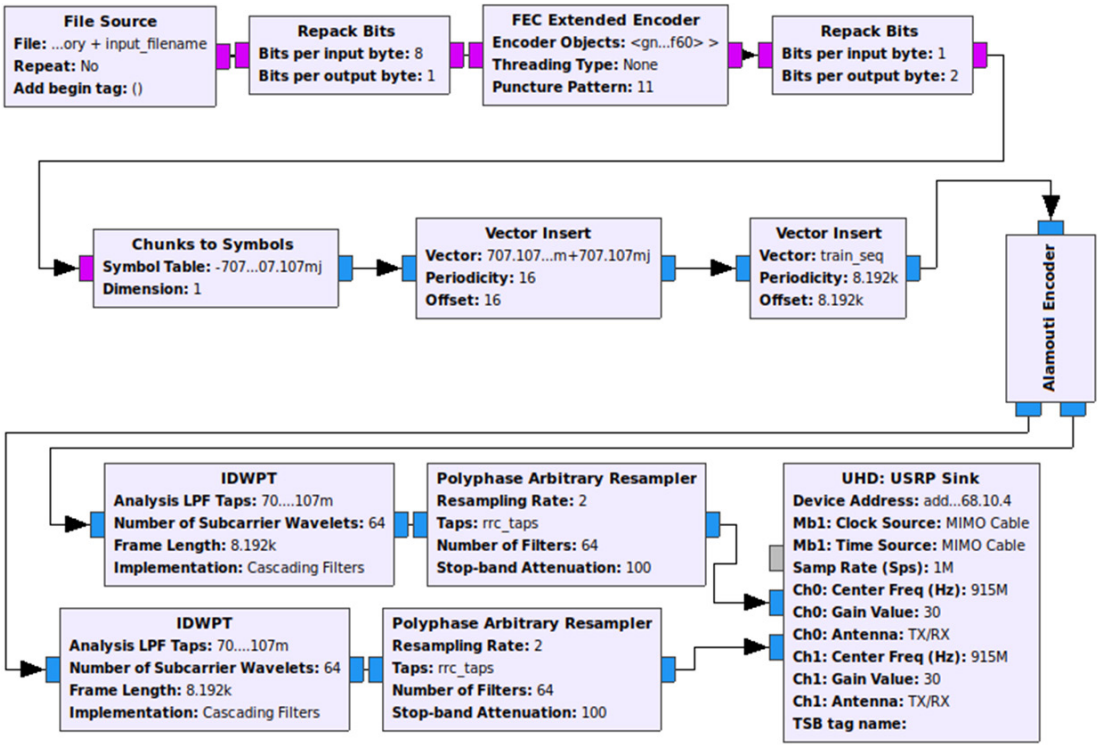

Figure 10. Example GNU Radio Flow Graph. Adapted from [11].

A GNU Radio flow graph can be executed as a standalone simulation or coupled with an RF front-end, like the USRP Sink in Figure 10, in order to transmit or receive OTA [39]. This is an analogous relationship to MATLAB and SIMULINK, where the

former provides the underlying code structure and the latter provides a graphical programming environment.

The core package of GNU Radio features a library of blocks including waveform generators, modulators, instrumentation and measurement sinks, math operators, filters, and Fourier Analysis tools [41]. In GRC the various blocks are arranged in order such that the streaming data passes through the flow graph sequentially in order to produce the desired digital signal processing effects. The GNU Radio software utilizes Python as a scripting language to organize and connect the functional flow graph blocks while processing-intensive functions, such as the scheduler and DSP blocks themselves, are implemented via C++ code [39], [40]. The interface between Python and C++ is created using the Simplified Wrapper and Interface Generator (SWIG). According to [42], SWIG will "generate the glue code" to allow higher level languages and compiled programing environments to call the underlying C++ code. Additionally, GNU Radio has many libraries that provide underlying features. One example of this is the C++ Boost library which provides access to smart pointers. These pointers track shared dynamically allocated objects in the code and help prevent exceptions by ensuring their proper destruction [43].

3. Runtime Scheduler

Knowledge of the GNU Radio scheduler is vital to understanding the GNU Radio capabilities and limitations. The runtime scheduler attempts to optimize flow graph throughput by processing large blocks of samples at a time via parallel computational nodes vice using a sample-by-sample processing approach [44], [45]. This allows more efficiencies in calculating large batches of samples but can also lead to increased latency [44].

Additionally, since samples are processed in batches, it generally prevents logical loops when building flow graphs because the data must follow a continuous stream without feedback [39]. GNU Radio does have a work around in the form of asynchronous messages. The scheduler does not process these messages along with the sample stream but instead they are sent to a different queue and read in by subscribing blocks [46], [47]. This work around allows some feedback but it is not efficient for a sample-by-sample

feedback operation due to delays in the message queue. Ref [48] does note that a designer may be able to force some alignment and timing policy in their implementation.

4. Out-of-Tree Blocks

Out-of-Tree (OOT) blocks refer to GNU Radio components that are not part of the pre-installed source tree. These blocks give developers the ability to create custom DSP blocks which expand the GNU Radio functions [49]. These are of utmost importance for realizing and testing new modulation and coding schemes as well as implementing non-standard control structures or DSP algorithms. These blocks can be implemented in C++ or Python but there are tradeoffs between the two programming languages. Ref [48] notes that Python is generally easier for newer developers while on the other hand C++ will usually have a higher throughput for a given algorithm. Interestingly, the OOT Block design allows an entire flow graph to be captured as a single block, known as a hierarchical block, and implemented as a component within a separate GNU Radio flow graph [49]. Chapter V will demonstrate ways in which OOT blocks were utilized to enable DSP modules and aspects of medium access control as discussed in Chapter III.

III. WIRELESS MEDIUM ACCESS CONTROL

A. OVERVIEW

Multiple access techniques employ resource partitioning and protocols in order to prevent collisions that may occur when several terminals are competing for usage of a communications channel. Usage deconfliction of the shared channel, also known as a medium, is vital to modern day communications systems and networks which typically have numerous simultaneous users. These methodologies reside in the Medium Access Control (MAC) sublayer within layer 2 of the OSI model [14] and are implemented in order to allow efficient and orderly use of channel capacity [16]. MAC protocols and techniques are typically classified according to two main categories, conflict-free and contention based [14]. An overview of various practices is provided in Figure 11 with brief explanations in the ensuing sections.

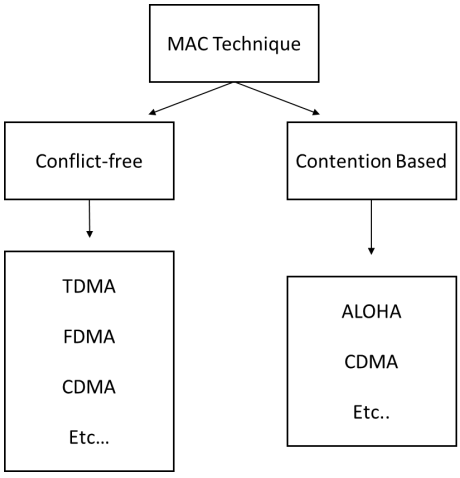

Figure 11. MAC Sublayer Techniques. Adapted from [14].

B. CONFLICT FREE MAC PROTOCOLS

Conflict-free MAC protocols avoid wireless transmission collision by partitioning the available communication channels and assigning separate sections to individual users. Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA),

and Code Division Multiple Access (CDMA) are traditional methods by which the communication medium is divided [18]. While not covered in the following sections, it should be noted that other techniques exist including token based turn taking and polling systems [16].

1. TDMA

TDMA splits the communication system transmission time into multiple time slots which are allocated amongst the radio terminals [14]. These time slots are typically separated by a guard interval, which is a space of null time, to prevent accidental overlap due to synchronization issues [16]. Since different users are only authorized to communicate over non-overlapping time slots, the problem of collision is avoided as shown in Figure 12.

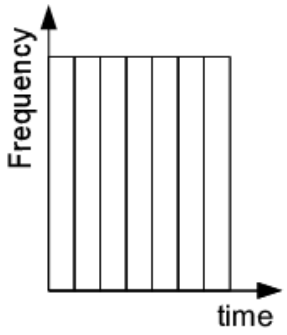

Figure 12. Depiction of TDMA. Adapted from [50].

2. FDMA

FDMA addresses the problem of transmission collision by employing several carrier frequencies, subchannels, to allow transmissions from multiple users at the same time [14]. Similar to TDMA guard intervals, adjacent subchannels are usually separated by guard bands of unused frequency bandwidth to prevent interference [16]. A representation of the FDMA technique is shown in Figure 12.

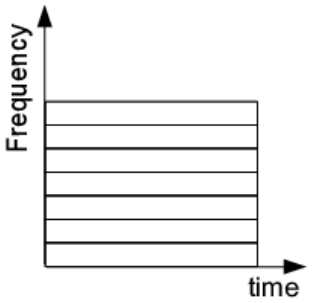

Figure 13. Depiction of FDMA. Adapted from [50].

3. CDMA

CDMA works via a similar process to that described by MCM techniques in II.C. Orthogonal code bases are employed to create subchannels which utilize the overlapping frequency bands and timeslots. It should be noted that this MAC technique typically requires more complex receivers to decode the transmitted signal [14]. The CDMA technique can be seen in Figure 14.

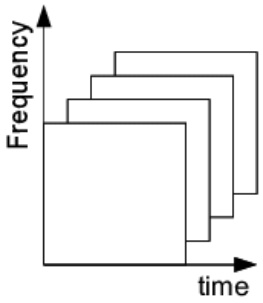

Figure 14. Depiction of CDMA. Adapted from [50].

C. CONTENTION BASED MAC PROTOCOLS

Contention Based techniques, also called random access techniques, allow a terminal to access the shared medium at nearly any time to transmit a message within the bounds of the protocol. Some protocols require certain conditions be met like having the channel sensed prior to transmission or they have specific actions in the event of a suspected collision. The methodologies tend to be most efficient for bursty traffic in light to moderate load conditions [16].

1. Aloha

One of the earliest Contention Based techniques was called ALOHA. Under the Pure Aloha protocol, if a terminal had data to send it would transmit immediately and if an acknowledgement was not received within a certain timeframe then the message would be retransmitted. After a specified number of reattempts, the transmitter would give up [16]. An evolution of this technique was called Slotted Aloha. In this method, stations could only transmit at the beginning of a synchronized message timeslot; thereby lowering the number of collisions that could occur while in the middle of a message transmission. Under this implementation a collision would occur over the entire transmission frame or not occur at all, increasing reliability [18].

2. CSMA

The Aloha variants were found to have relatively low efficiencies [16]. A major leap forward in random access techniques was the idea of carrier sense multiple access (CSMA). Under this system, a transmitting station must listen to the channel medium and determine if it is in use. If so, the station must wait until the transmission has ended before transmitting [14]. This lowers the number of collisions since collision will only occur under two conditions[16]:

- Simultaneous transmission from multiple terminals
- Sensing an idle medium due to propagation delay while a message from another station is already being transmitted. That is to say, a terminal thinks the medium is clear because the transmission from the other station has not reached it yet.

In evolved versions of CSMA, various features are added such as timeslots similar to Slotted Aloha, random waits before transmitting, and random backoff (i.e., waiting) periods before retransmission after a collision has occurred [14], [16]. Examples of these conditions implemented as Non-persistent, 1-persistent, and p-persistent CSMA are shown in Figure 15. Note that more advanced implementations, such as CSMA with Collision

Avoidance (CSMA/CA) further define timeslots in the transmission window and in some cases require acknowledgements from the destination station [14].

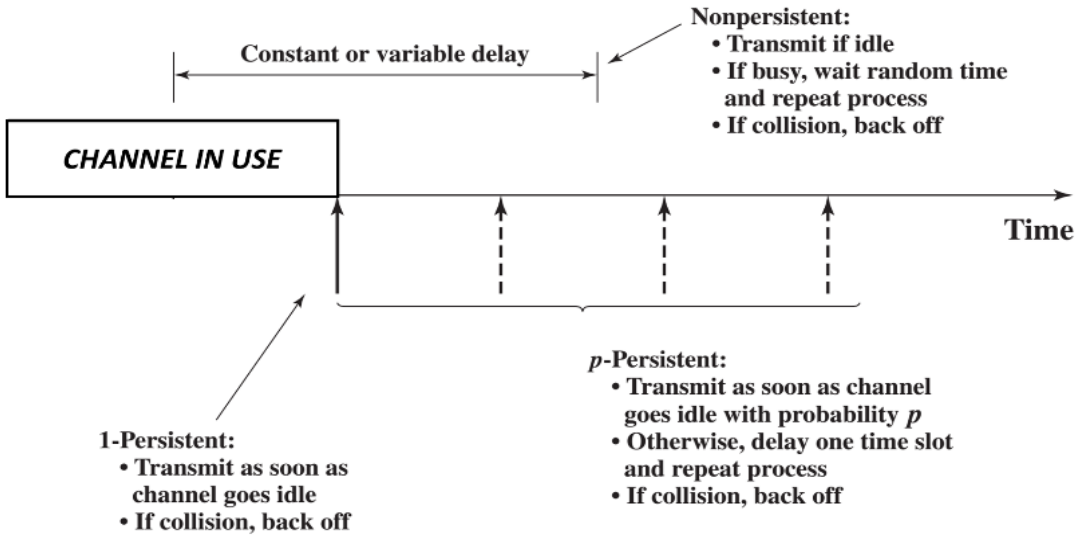

Figure 15. CSMA Variations in Transmit Rules and Backoff Times. Adapted from [14].

D. GNU RADIO MAC AND FLOW CONTROL

Over the past few years there has been increasing interest in MAC and communication flow control methodologies in GNU Radio. Some of these newer implementations involve full duplex communications, allowing transmission and reception simultaneously on the same terminal, while others are half duplex, allowing transmission and reception but not at the same time [16].

1. Full Duplex Methodologies

While a simple example of full duplex could be the use of FDMA as described previously, there are more intricate and bandwidth efficient designs available. More recent innovations have looked at solving the problem of both transmitting and receiving simultaneously on the same frequency and same antenna. Reference [51] points out that the major inhibitor to this approach is self-interference, where the transmit power dwarfs the receive power on a given terminal. The author describes their implementation and

showed that it was entirely feasible to support this level of Full Duplex in GNU Radio via digital cancellation blocks and advanced multithreading.

Another implementation is noted in [52] where the designers took a more hardware-centric approach. They describe implementing an RF canceller on a printed circuit board (PCB). This PCB facilitates integration of their self-interference cancellation hardware with a SDR RF front-end and note that it shows improved performance over integrated circuit based solution in their test bed.

2. GNU Radio Based MAC

Software repositories, like GitHub or CGRAN, and various forums and SDR wikis currently house a multitude of software based, GNU Radio MAC implementations that range from TDMA and CSMA to full implementations of IEEE compliant 802 series protocols [53], [54]. Notably, quite a few of the available designs operate at the OSI model network layer or above. Refs [48], [55]–[59], are just a few examples of the various open source MAC layer protocols available on the internet.

The "Simple MAC" GNU Radio block implemented in [48] operates at the network and transport layers for addressing, network layer framing, and employment of Stop-And-Wait Automatic Repeat Request (ARQ). ARQ is an error handling flow control technique in which the destination provides feedback to the source transmitter about whether or not a message was received correctly [14]. In the Stop-And-Wait variation, the transmitter will wait for each package to be acknowledged before transmitting the next data sequence [18]. The Simple MAC design appears to be dependent upon additional asynchronous message passing features utilized on specific varieties of SDR RF Front-end, specifically those utilizing the UHD library [48]. It is able to utilize any physical layer transceiver configuration through the use of OOT hierarchical blocks and is configurable for multiple MAC protocols including CSMA and TDMA [48].

The GNU Radio Wireless Network (GWN) is another open source methodology for packet based MAC protocol usage in GNU Radio. Per [57], GWN acts as a toolkit extension for GNU Radio that adds useful functions for MAC implementations such as message orientation, events, time handling, and finite state machines (FSMs). In GNU

Radio, the message is normally stream based but the message orientation handles data in discrete byte groupings. The events and time handling implementations allow GWN to react to internal signaling and perform actions based on an underlying timing structure. Lastly, as [57] points out, most modern communications protocols rely on states and transitions which can be mathematically modeled as FSMs. In aggregate these functions allow for a versatile MAC implementation.

3. SDR Challenges

While SDR provides a convenient environment for implementing flexible, adaptable radios, it is not without its drawbacks, especially in the realm of MAC protocols and flow control. Ref [57] points out that while there exists a collection of GNU Radio tutorials, the documentation can be scare in certain areas. The author also notes that GNU Radio version updates can be frequent, breaking existing flow graphs since backwards compatibility is not guaranteed.

Latency is another major issue that makes MAC and flow control difficult. As evidenced by the brief discussion earlier in this chapter, several techniques including TDMA, Slotted Aloha, and variants of CSMA require strict timing compliance. Ref [48] noted that the Universal Software Radio Peripheral (USRP), an RF front-end, in use during testing resulted in poor performance of the Stop-And-Wait ARQ protocol. This idea is echoed in [59], for SDRs in general, where it was noted that multiple interconnecting buses and non-homogenous processing units induce large delays and jitter. Their test results revealed round trip times between GNU Radio and the SDR FPGA of up to 1 millisecond. Their testing also showed that the timing jitter prevented precise scheduling of CSMA backoff. Backoff times between 1ms and 100ms were found to be unpredictable, requiring the host computer to utilize a large minimum backoff time.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PREVIOUS WORK

The work presented in this thesis is primarily based on the work accomplished as part of two previous Naval Postgraduate School thesis projects. These works will be briefly reviewed in order to facilitate discussion of the implementation in Chapter V and the results in Chapter VI.

A. WPM MIMO SDR

The information contained in this section is derived solely from [11], the first of the two previous theses to be examined, unless otherwise specified. In [11], the author delineates the design, implementation, and test of a GNU Radio based SDR transmitter and receiver. The SDR implemented MCM using WPM and applied Alamouti coding with a 2x2 MIMO antenna configuration.

1. SDR Transmitter

The transmitter sequence of operations is depicted in Figure 16.

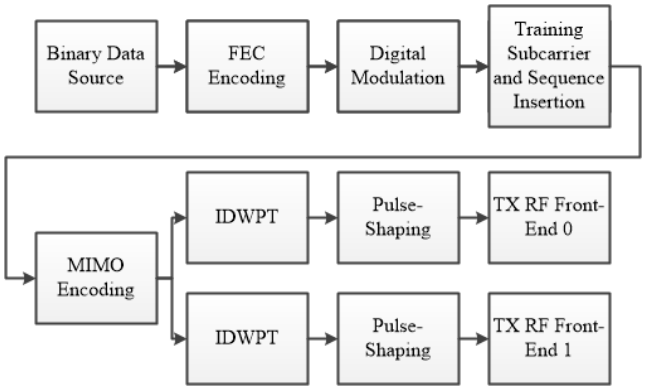

Figure 16. Block Diagram Representation of Reference [11] Transmitter. Source: [36].

While similar to the Digital Communications Model of Figure 2, there are several important additions to note. The transmitter implements channel encoding, as described in Chapter II.A.2.c, via a forward error correction (FEC) algorithm utilizing a half rate, non-

symmetric, constraint length seven convolutional code. The digital modulation scheme was QPSK and the pulse shape utilized was a square-root raised cosine with a 0.35 roll-off factor.

2. SDR Receiver

The receiver sequence of operations is depicted in Figure 17.

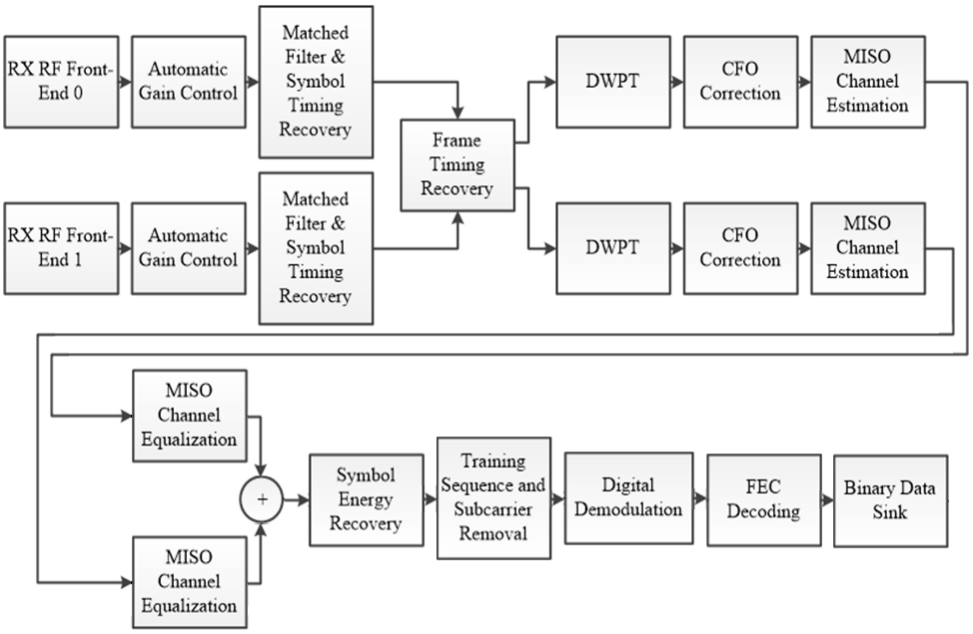

Figure 17. Block Diagram Representation of Reference [11] Transmitter. Adapted from [36].

Functions not discussed previously in II.A.2 or in this thesis, will be noted. The Automatic Gain Control (AGC) block sets the average incoming sample energy to a value of two since the receiver antennas acquire signals from the two transmitter antennas, both of which are set to transmit symbols at unit energy. Matched filtering is implemented to maximize the signal-to-noise ratio (SNR) during sampling and the function of frame timing recovery is implemented via a GNU Radio OOT block called the Frame Synchronizer. This block is especially important since it provides synchronization between incoming data streams at each receive antenna. If the Frame Synchronizer cannot detect an adequate peak correlation between the two received streams, then the associated frame is dropped from

the received data stream. The Carrier Frequency Offset (CFO) block accounts for differences between transmitter and receiver carrier frequency due to Doppler shift and local oscillator drift. Lastly, it should be noted that the each antenna is treated as a separate 2x1 MISO system as depicted in Figure 6. The resultant equalized streams are added together and then the symbol energy is recovered.

B. ARTIFICIAL NOISE MASK TECHNIQUE

The information contained in this section is derived solely from [3], the second foundational thesis, unless otherwise specified. In [3], the author describes a method to extend the work described in [60] from a LOS to a BLOS PLS technique. Reference [3] posits that purposeful destructive interference, akin to beamforming, can be used to apply an additive mask that effectively increases the SNR at an eavesdropper location while canceling out at the intended receiver position. This mask utilizes the CSI of the intended receiver as a key to unlock the physical layer encryption provided by the noise mask.

1. Masking Vector

In order to derive the masking equations, the author added generic mask variables (M_0, M_1, M_2, and M_3) to the received symbols of an Alamouti Encoded MIMO transmission as portrayed in Figure 18.

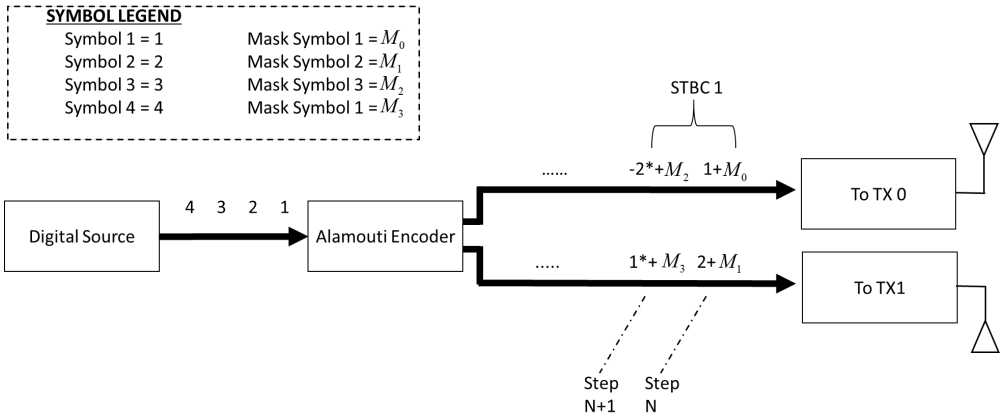

Figure 18. Visual Representation of the Alamouti STBC Encoding Scheme with Masking Vector Applied. Adapted from [3], [19].

Here the first STBC of an Alamouti Encoding scheme, similar to that of Figure 8, is shown with the masking vector, $\mathbf{M} = [M_0\ M_1\ M_2\ M_3]$, added to the applicable symbols. It is assumed that the channel state is constant during the two time steps within a single block of the STBC. A new masking vector must be calculated whenever the communications channel changes significantly. The mask addition transforms Table 2, from II.D.2, into Table 4 and changes the received STBC transmissions of (5) into (8).

Table 4. Transmitted Masked Symbols. Adapted from [3], [11].

Time Step	TX Antenna 0	TX Antena 1
N	$s_N + M_0$	$s_{N+1} + M_1$
$N+1$	$-s^*_{N+1} + M_2$	$s^*_N + M_3$

$$\begin{aligned} r_0 &= h_0(s_N + M_0) + h_1(s_{N+1} + M_1) \\ r_1 &= -h_0(s^*{}_{N+1} + M_2) + h_1(s^*{}_N + M_3) \\ r_2 &= h_2(s_N + M_0) + h_3(s_{N+1} + M_1) \\ r_3 &= -h_2(s^*{}_{N+1} + M_2) + h_3(s^*{}_N + M_3) \end{aligned} \tag{8}$$

The decision statistics that result from mask addition when applied to a system with Maximal Ratio Combining (MRC) Equalization are shown in (9) and (10).

$$x_N = \frac{h_0^* r_0 + h_1 r_1^* + h_2^* r_2 + h_3 r_3^*}{|h_0|^2 + |h_1|^2 + |h_2|^2 + |h_3|^2} - \frac{(|h_0|^2 + |h_2|^2)M_0 + (h_0^* h_1 + h_2^* h_3)(M_1 + M_2^*) + (|h_1|^2 + |h_3|^2)M_3^*}{|h_0|^2 + |h_1|^2 + |h_2|^2 + |h_3|^2} . \tag{9}$$

$$x_{N+1} = \frac{h_1^* r_0 - h_0 r_1^* + h_3^* r_2 - h_2 r_3^*}{|h_0|^2 + |h_1|^2 + |h_2|^2 + |h_3|^2} - \frac{(h_0 h_1^* + h_2 h_3^*)(M_0 - M_3) + (|h_1|^2 + |h_3|^2)M_2^* - (|h_0|^2 + |h_2|^2)M_3^*}{|h_0|^2 + |h_1|^2 + |h_2|^2 + |h_3|^2} \tag{10}$$

Without the mask applied, the decision static would be represented solely by the first terms of (9) and (10). This led to the conclusion that if the second term of each equation equaled zero (or was insignificant relative to the first term), it would achieve the effect of mask cancellation at the intended receiver (i.e., the mask would be transparent if tuned to the receivers specific channel state). The condition of mask cancellation just described is represented by (11) and (12).

$$(|h_0|^2 + |h_2|^2)M_0 + (h_0^* h_1 + h_2^* h_3)M_1 + (h_0^* h_1 + h_2^* h_3)M_2^* + (|h_1|^2 + |h_3|^2)M_3^* = 0 \quad (11)$$

$$(h_0 h_1^* + h_2 h_3^*)M_0 + (|h_1|^2 + |h_3|^2)M_1 - (|h_0|^2 + |h_2|^2)M_2^* - (h_0 h_1^* + h_2 h_3^*)M_3^* = 0 \quad (12)$$

Reference [3] makes it clear that the mask cancellation conditions are dependent on the specific method of equalization utilized. For the SDR described in the previous subsection, the mask cancellation conditions are those shown in (13) and (14).

$$\frac{|h_0|^2 M_0 + h_0^* h_1 (M_1 + M_2^*) + |h_1|^2 M_3^*}{|h_0|^2 + |h_1|^2} + \frac{|h_2|^2 M_0 + h_2^* h_3 (M_1 + M_2^*) + |h_3|^2 M_3^*}{|h_2|^2 + |h_3|^2} = 0 \quad (13)$$

$$\frac{h_0 h_1^* M_0 + |h_1|^2 M_1 - |h_0|^2 M_2^* - h_0^* h_1 M_3^*}{|h_0|^2 + |h_1|^2} + \frac{h_2 h_3^* M_0 + |h_3|^2 M_1 - |h_2|^2 M_2^* - h_2 h_3^* M_3^*}{|h_2|^2 + |h_3|^2} = 0 \quad (14)$$

2. Additional Constraints

In order to bound mask power, the magnitude of the mask vector $\mathbf{M}$ is considered. For a unit energy mask vector, the magnitude would be defined by (15). The constant on the right side of (15) may be altered in order to change the value of the mask relative to the signal energy. This quantity is called the signal-to-mask ratio (SMR) and is defined as shown in (16). The numerator contains the expectation of the magnitude of the transmitted signal energy while the denominator contains the user-defined mask vector magnitude.

$$\sum_{i=0}^{3} |M_i|^2 = 1 \quad (15)$$

$$SMR = \frac{E\left\{|s|^2\right\}}{|\mathbf{M}|^2} \tag{16}$$

For the remainder of this subsection, the case of a unity energy mask will be considered. Equations(13), (14), and (15) contain the four complex unknown mask vector values within two linear equations and one non-linear equation. While there are potential methods to solve this system of equations, [3] proposes a simple solution. Since many masks solve the three equations, the author proposed adding (17) and (18) to form a fully bound set of four linear equations that can be solved via basic matrix manipulation. Note that a trivial solution to (13) and (14) would have been to set the mask values to zero, but in that case the transmission data stream would remain unmasked.

$$\sum_{i=0}^{3} M_i = 0 \tag{17}$$

$$\frac{1}{4}M_0 + \frac{1}{4}M_1 + \frac{1}{4}M_2 - \frac{1}{4}M_3 = 1 \tag{18}$$

Equations (13), (14), (17), and (18) form a solvable set of linear equations with a distinct solution, notated as $\mathbf{B}$. In order to apply the energy constraint imposed by (15), $\mathbf{B}$ is normalized via (19) resulting in the desired masking vector, which solves (13), (14), and (15).

$$\mathbf{M} = \frac{\mathbf{B}}{|\mathbf{B}|} \tag{19}$$

With the baseline foundational thesis work having been discussed in this chapter, Chapter V will detail the realization of the noise mask algorithm and CSI acquisition method as well as implementation of custom GNU Radio flow control modules.

V. IMPLEMENTATION

This chapter describes the method by which the WPM MIMO SDR communications system of [11] was enhanced with the artificial noise mask that was simulated in [3] and [10]. Additionally, the rudimentary flow control features that were implemented in order to support early-stage testing and future communication system development will also be explored.

A. NOISE MASK APPLICATION

1. Mask Calculation Modules

The first step in implementation was creation of the noise mask calculator as an OOT block within GNU Radio. Reference [48] noted that this could be accomplished in either C++ or Python. While C++ would have offered a faster runtime for mask calculation, Python was chosen for ease of use and more rapid prototyping due to requiring fewer file changes with each iteration of code implementation. The custom C++ modules in [11] required 3 supporting scripts whereas Python only required two. The custom mask calculation block created used the GNU Radio Sync Block as its base because Sync Blocks generally produce the same number of output samples as input samples. This is as opposed to utilizing an Interpolator or Decimator Block as the base class.

a. Static and Random Masks

Two separate mask calculation blocks were created. The first applied a "static mask," strictly following the calculation schema proposed by [3], while the latter applied a "random mask" which introduced random elements into one of the arbitrary equations needed to find the masking vector. The reasoning behind this additional implementation was that a random mask could make eavesdropper attempts at "guessing" [10] the mask more difficult. Other than the additional processing required to introduce random elements at the transmitter, the random nature does not have to be accounted for at the receiver due to the mask transparency at the receiver when implemented properly.

The static mask implemented (13), (14), (17), and (18) as described previously, whereas the random mask changed (18) into (20). Here the static $\frac{1}{4}$ real-valued multiplicative constants were replaced by complex numbers (R_0 , R_1, and R_2) with random Gaussian distributed real and imaginary parts. Equation (17) could have implemented random elements as well but for a simple proof of concept, and to ensure linear independence, only one equation was altered for this thesis.

$$R_0 M_0 + R_1 M_1 + R_2 M_2 - R_2 M_3 = 1 \tag{20}$$

b. SMR

In addition to having two separate mask calculation modules, the mask-enhanced SDR implementation allowed for varying the SMR of the transmitted signal. The GNU Radio symbols were constructed with unit energy, thus in order to alter SMR, the magnitude of the mask vector was parameterized instead of being hard coded to unity as per (15). This allowed SMR to function as a variable input to the mask calculation blocks in GNU Radio and solved for via (16).

c. CSI Acquisition

The original implementation of the mask-enhanced SDR was designed as a full duplex, frequency division multiple access system. The intent was to have separate uplink and downlink frequencies in order to pass the mask values continuously as part of a message header. This potentially would have allowed updating of the mask at the transmitter in near real time. The radio hardware appeared to work in transmit and receive mode simultaneously, however, the self-interference issue described in III.D.1, and further discussed in [51], proved to be prohibitive for full duplex operations. Since the focus of this thesis was real world mask application, rather than full duplex operations, a different method was investigated.

The next course of action was to attempt to utilize channel reciprocity, as described in II.B.3, in order to obtain the distant end CSI. In order for this to work, an unmasked setup message would initially need to be transmitted from one station to the other. The

receiving station in this first exchange would be able to gather the first set of CSI for mask production. Afterwards, masked communication data messages could continue between the two terminals. This idea is depicted in in Figure 19. Masked communication can continue from steps 2 through N if the mask is properly applied and messages are transmitted with sufficient regularity in order to accurately characterize the channel state.

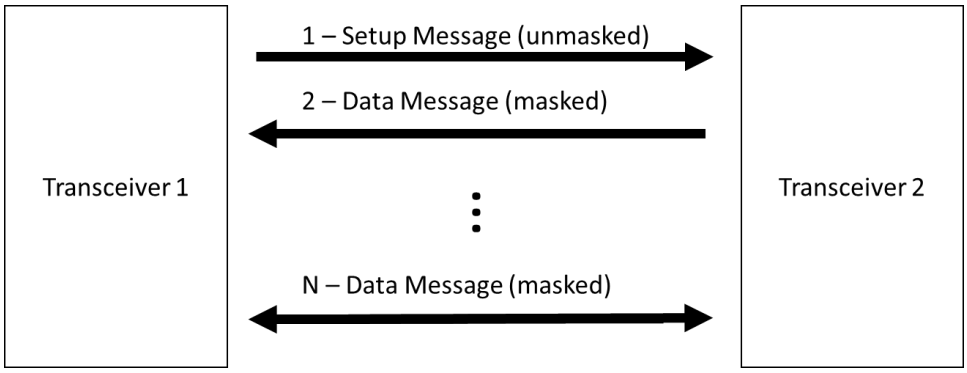

Figure 19. Masked SDR Communication Flow

The foundational WPM MIMO SDR of [11] provided the CSI directly from the MISO channel estimation block shown in Figure 17. The data was offloaded via GNU Radio ZMQ message source and sink blocks, a set of asynchronous socketed message ports [61]. The use of the ZMQ blocks was necessitated for the planned full duplex version of the mask-enhanced SDR in order to work around the fact that GNU cannot natively support a feedback loop from receiver to transmitter. While not mandatory for proper operation in the final configuration, the ZMQ blocks remained in place for future usage. Once transferred via the ZMQ block sockets, the CSI values were used in the applicable mask calculator blocks and the resultant mask vector values (M_0, M_1, M_2, and M_3) were each written to individual files to be used in the transmitter. An example flow graph featuring the ZMQ socket source coming from the receiver, static mask calculation, and mask vector file writing is shown in Figure 20. Several implementations of this flow graph were created to support tests of various conditions such as static vs. random masking and intended receiver vs. eavesdropper configurations.

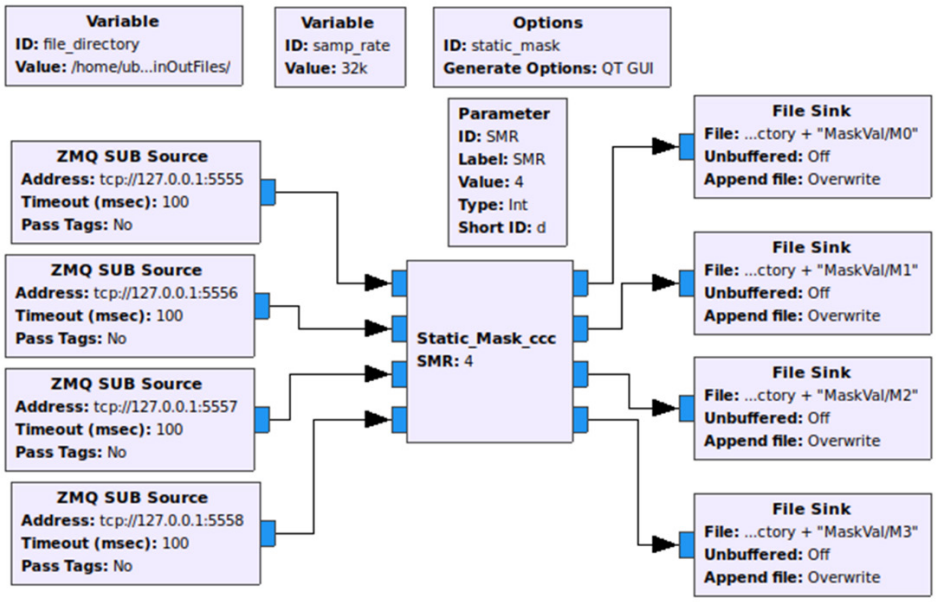

Figure 20. Example Mask Calculation Implementation

2. Mask Adder Block

In addition to the mask calculator block, a custom OOT Mask Adder Block was created in order to add the masking symbols which were previously written to text files. The Mask Adder Block applies the masking symbols to the transmit data stream immediately following the MIMO Encoding operation of the transmitter. This OOT block was built as a hierarchical block (i.e., a collection of resident GNU Radio block modules combined together to create a single block). Figure 21 shows how the hierarchical Mask Adder Block would be implemented as a collection of separate GNU Radio blocks. Referring to Figure 18, two Mask Adder Blocks were required, one for the TX 0 data stream and one for the TX 1 data stream. The STBC data is de-interleaved, the applicable mask symbol is added (for TX 0 in Figure 18, Mask A is M_0 and Mask B is M_2), and then the data stream is interleaved to produce the masked output data stream. For a more complete view, Figure 22 portrays how the Mask Adder Block is implemented between the transmitter Alamouti Encoder and IDWT blocks. The output of the Alamouti Encoder is sent to two separate Mask Adders along with the applicable masking symbols from file source blocks. The file sources for the mask values are set to repeat in order to account for different sized message.

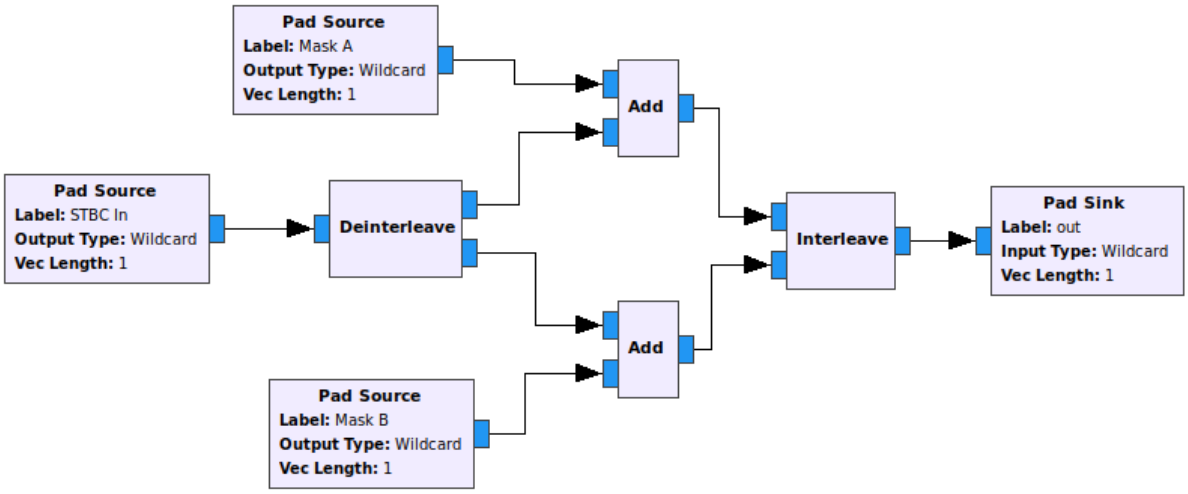

Figure 21. Mask Adder Module Internal Block Implementation

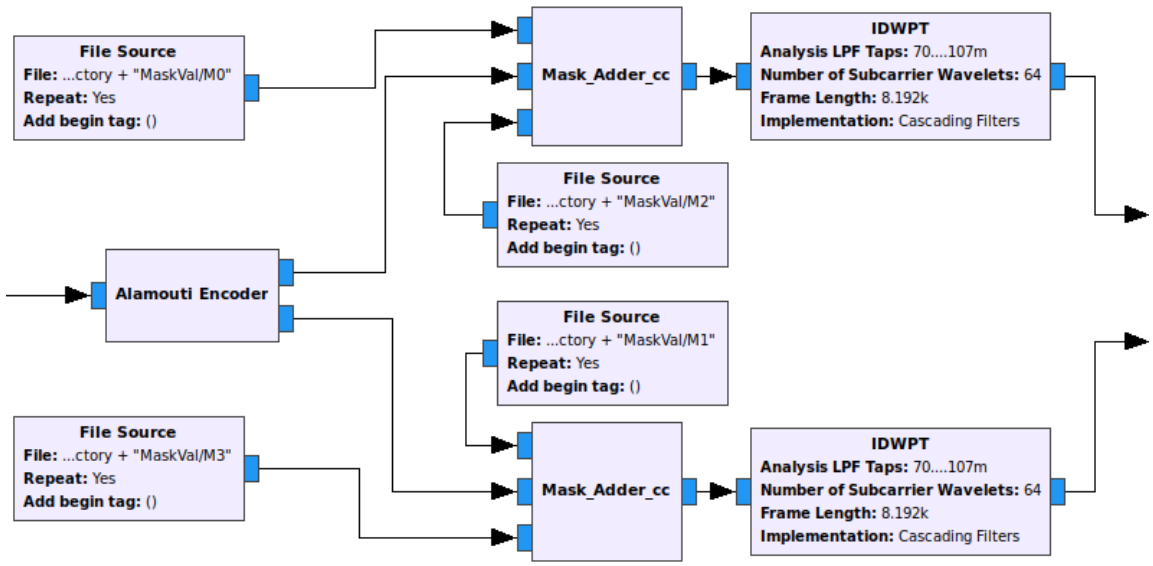

Figure 22. Flow Graph Implementation of the Mask Adder Block

3. Additional Considerations

Upon initial mask implementation it was found that the beginning of the setup message, shown as step one in Figure 19, was being received distorted. This resulted erroneous CSI, which in turn caused the mask calculation to be incorrect causing increased bit error rate (BER) at the intended receiver. To solve this problem, a header was added to

the transmitted test file and the AGC maximum gain values were lowered. These steps allowed the AGC to tune correctly and the mask calculation to proceed without errors. Additionally, it was noted that the end of data messages would sometimes be distorted or missing from the received file. While the root cause of this was not discovered, a simple workaround was the use of trailer bits to fully encapsulate the message. The header and trailer are completely removed prior to the received file being compared to the reference file. This increased message overhead but allowed any lost trailer bits at the end of the message to not affect BER calculation.

B. FLOW CONTROL

The primary focus of this thesis was testing the additive mask with OTA communications, however, flow control was also considered in order to create a fully functional transceiver and support future design efforts. Additionally, it was determined that basic flow control would aid preliminary test operations by allowing the radio to automatically cycle between transmit and receive modes without user intervention.

The primary method of flow control was polling; one transceiver would start in a waiting state and when a transmission was received it would collect the CSI of the receiver and send back a masked message. The system was designed to be message length agnostic (i.e., the polled transceiver would wait a set amount of time after the transmission ended before it stopped "listening" and responded back). In order to determine when the distant end had finished transmitting and when all samples were processed, spectrum sensing and sample sensing capabilities were implemented.

1. Spectrum Sensing

In Section III.C.2 spectrum sensing was discussed as a key enabler for CSMA, allowing a terminal to determine when the channel is free of transmissions and the medium can be accessed for usage. In the polling implementation used in this research, CSMA was included in the original design configuration for a similar purpose. Once a transceiver had started to receive data, it would keep sensing the channel and writing the message to a file until the spectrum no longer indicated that a transmission was taking place. At this point the transceiver, which had just collected fresh CSI, would respond with a masked data

message. In support of this, the receiver flow graph of [11] was modified with the addition of the GNU Radio Function Probe, Probe Avg Mag^2, and Adder blocks as shown in Figure 23. Here the signal is received from the USRP Source, level controlled by the AGCs, and then added together. Due to the Alamouti coding scheme, if a non-zero signal is transmitted, both steps of its STBC cannot result in a zero magnitude value at the output of the adder. The adder output is passed to the Probe Avg Mag^2 which will compute a running average of the square of the input magnitude [62]. The Function Probe block was formatted to read the value produced by the Probe Avg Mag^2. Additionally, code was written for the Function Probe function definition within the receiver flow graph python script.

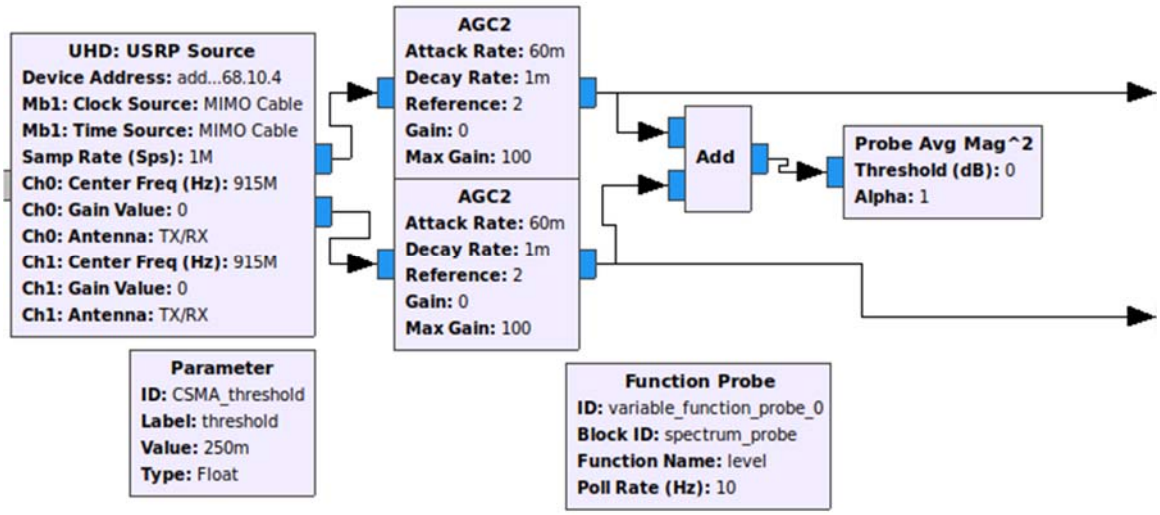

Figure 23. Receiver Spectrum Sensing GNU Radio Blocks

In order to sense the channel state and then decide when transmission from a distant terminal had completed, the custom code in the python script for the Function Probe followed the Flowchart shown in Figure 24.

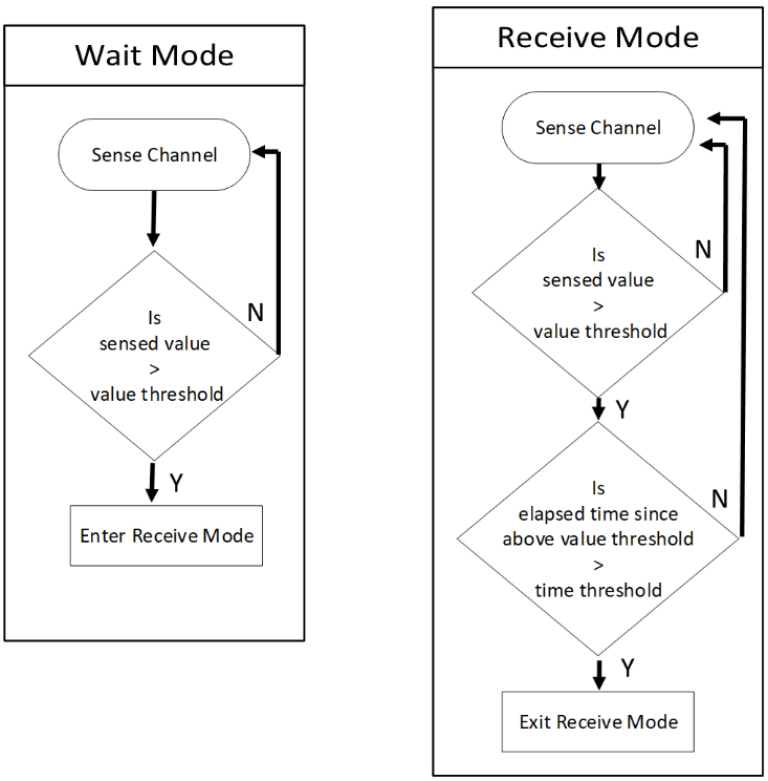

Figure 24. Receiver Spectrum Sensing Flow Chart

The transceiver starts in Wait Mode and only shifts into Receive Mode once the sensed value of the function probe is above the parameterized threshold. This value will be environmentally dependent but, for this thesis the CSMA Threshold parameter block was set to 0.25. It was originally lower but it was noted that unknown signals would spuriously set the transceiver into Receive Mode. Once in Receive Mode, the channel would continue to be sensed until the received signal level dropped below the designated threshold. At this point the system checks against a time threshold to ensure that the transceiver remains in Receive Mode for a minimum amount of time in case the distant end temporarily ceased transmitting due to a processing lag or other anomaly. The time threshold value is equipment dependent but in this thesis implementation it was generally on the order of seconds.

During preliminary testing it was discovered that the time threshold required varied due to inconsistencies with host computer processing time. In some cases the system would not start writing the received data to the output file until upwards of 10 to 15 seconds after

the other terminal had stopped transmitting. In this case, if the time threshold was set too short the whole message could be missed. This led to the decision that, in order to enable a polling type of flow control, the system would need to start sensing the samples as they moved through the flow graph rather than just the spectrum itself.

2. Sample Sensing

While sample sensing would not prove to be completely lag free, it could better account for processing delays in the system. In order to apply sample sensing, the Function Probe and Probe Avg Mag^2 blocks were utilized in addition to a custom OOT Counter Block as shown in Figure 25 and Figure 26. The custom Counter Block simply counted the number of samples that passed through it providing a steadily increasing number while a given flow graph was in operation. In Figure 25, the sample sensing system was implemented in the receiver of [11] right before the data stream was converted into bits and written to the output file. This location helped to deal with some of the processing lag encountered. Figure 26 shows the same sample sensing system implemented into the transmitter in order to keep track of when the transmit file completes. In the transmitter version, the samples are sensed right before they are sent over the air via the USRP Sink block. When the samples stopped, indicating the source file was fully processed, the transmitting system could switch to a receive flow graph in order to acquire the response to its polling request.

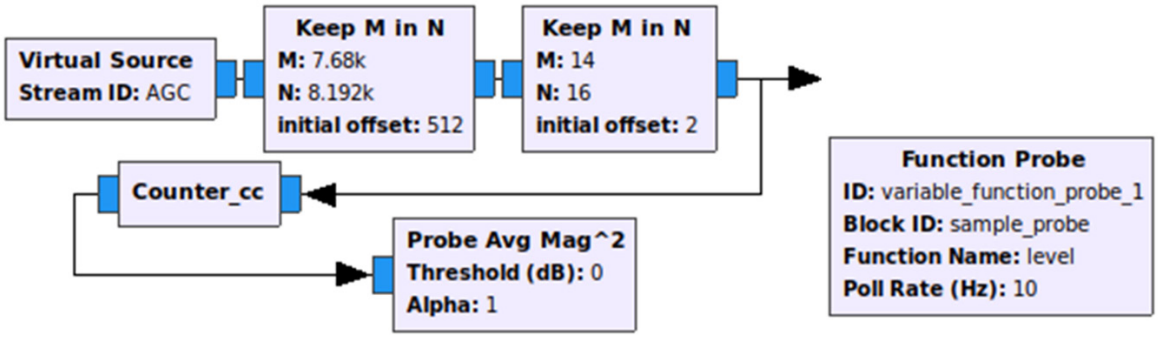

Figure 25. Receiver Sample Sensing GNU Radio Blocks

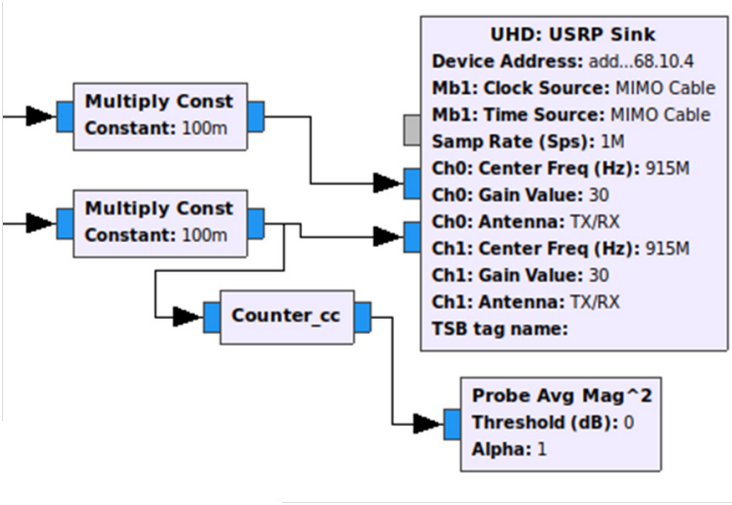

Figure 26. Transmitter Sample Sensing GNU Radio Blocks

Both transmit and receive versions of the sample sensor implementation had identical code added to the Function Probe function definition in their respective python scripts. The operation of the function probe is as shown in Figure 27.

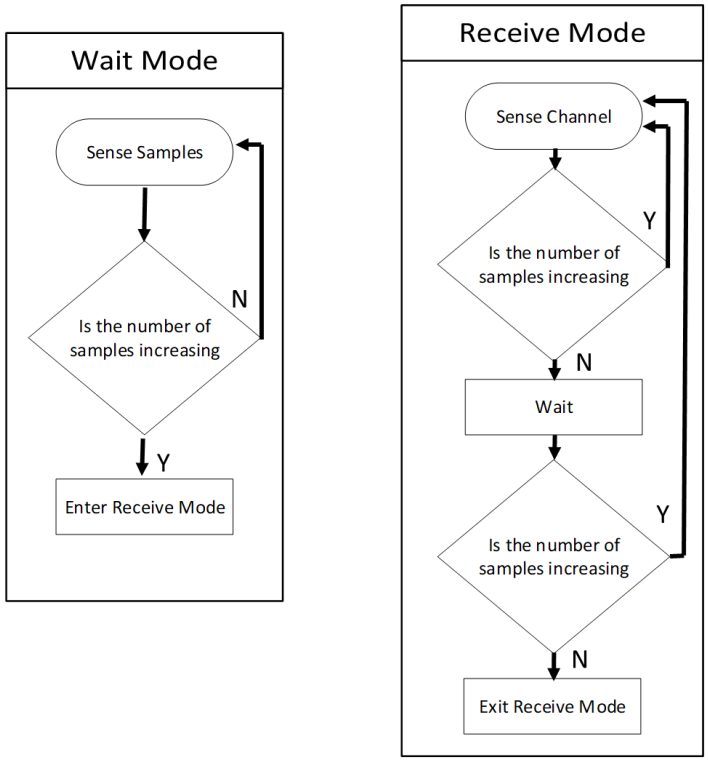

Figure 27. Receiver Sample Sensing Flow Chart

The sample sensing flow chart shows that the system transitions from Wait Mode to Receive Mode whenever it evaluates that the value of the counter block is increasing (i.e., there are samples being processed by the flow graph). Note that in this context Receive Mode indicates that samples are being received at either a transmitter or receiver. Once in Receive Mode, the sample sensing system looks for a state when the number of samples stops increasing. If this state is encountered twice, after a sufficient waiting period in between the two checks, then the system will exit Receive Mode. The waiting period was added because of early termination of flow graphs occurring due to system lags where samples would temporarily stop flowing for a couple of seconds. The exact length of wait time was host-computer-hardware dependent, but setting a wait time of 10 seconds appeared to prevent premature termination issues.

3. Supervisory Program

With the spectrum sensing and sample sensing systems in place, the final step for rudimentary flow control was enabling a method to shift between receiver and transmitter flow graphs. While [39] indicates that there are methods to pause a flow graph and reconfigure it on the fly, potentially allowing a GNU Radio flow graph to shift between transmit and receive functions, this proved ineffective during initial testing. It is believed that the issues experienced were due to transparent background messages that shutdown the USRP when the source file reached its end file marker [63].

In order to work around this limitation, a supervisory program was created. This program would utilize the python subprocess module, allowing creation and control of new processes from within a running script [64]. In effect the main supervisory program could be launched and then various GNU Radio flow graph processes could be called or terminated from within the higher level program. When a transceiver had finished its current mode of operation as indicated by lack of new samples, the flow graph could self terminates and an "exit file" would be written into the directory as part of the "Exit Receive Mode" process indicated in Figure 27. The supervisory program searches the directory for these files continuously and when one is found it will terminate all previous processes (if not already closed) and then start the alternate mode of operation. The "exit file" is simply

a blank text document but has a unique name that tells the supervisory program which mode of operation should be running. When launched on two terminals, the supervisory program could allow the two computers to automatically communicate back and forth, over the radio link, in support of early stage testing.

VI. RESULTS

A. EXPERIMENT SETUP

The hardware setup for this thesis is shown below in Figure 28. Each transceiver laptop was connected to a set of USRP N210 RF front-end devices via a gigabit Ethernet cable. Additionally, each set of N210s were connected together via a MIMO expansion cable for timing and synchronization [11]. During data collection, the transceiver sending the data message was configured with an SBX daughterboard in each N210 while the receiver had a WBX daughterboard in one N210 and an SBX daughterboard in the other. These two daughterboard are similar in frequency range, operation, and application with each having a noise figure of 5dB [65]. The only reason two different daughterboards were used was due to parts availability. The data rate during operations was set to 51.3 kB per second, the same value as OTA testing conducted in [11].

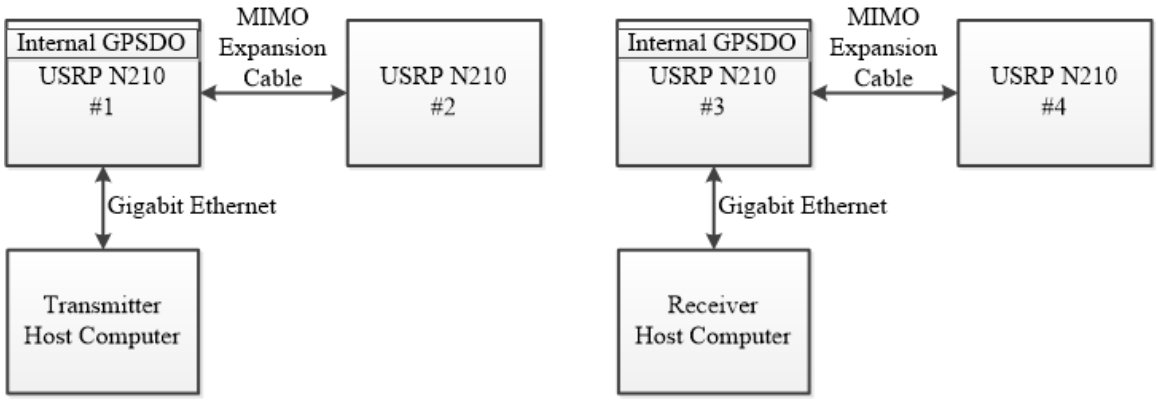

Figure 28. RF Front-End Hardware. Source: [36].

During testing, the supervisory programs of Section V.B.3 were only utilized in a limited manner in order to have better control over the test scenario rather than having both transceivers communicate automatically. One transceiver was started with a receiver flow graph and the applicable masking program, static or random, running in tandem. A small setup file of roughly 27 kB and consisting of an ASCII image was transmitted in order to record the CSI information and calculate the mask (Step 1 from Figure 19). The received

setup file was manually checked for any anomalies in size or content and then, if there were no issues present, the masked data message was transmitted for BER calculation (Step 2 from Figure 19). Setup file anomalies can be passed on to the receiver in a masked condition so this verification is important.

Custom python code was scripted in order to perform basic test file operations such as creation, formatting, and comparison. The test data message contained 80 million bits in the form of a randomly generated sequence of ASCII characters. This message size was selected to be the same as the one used in [11]. The test data message, once generated, was wrapped in header and trailer bits as described in Section V.A.3. At the transceiver responsible for receiving the masked signal, the acquired data message was stripped of its header and trailer bits, converted into binary digits, and then compared to a reference copy of the test data message. In order to test at an eavesdropper location, randomly generated channel state information was provided to the static and random masks for mask vector calculation.

To ensure fairness in comparison between multiple runs, the only BER attempts counted were those in which the header and trailer wrapper were successfully removed at the predetermined indices unless it was evident that the failure was due to the mask application. This allowed the selected attempts to be free of errors due to processing lag or other irregularities. The inability of the BER calculation script to find the predetermined start and stop indices associated with the wrapper, help show the file may have not been written correctly. If the two files were not aligned then the BER test script would not conduct an accurate bit by bit comparison.

It should be noted that a sufficiently masked signal will typically result in failure to record a file due to the samples not synchronizing at the MIMO Frame Synchronizer as described in Section IV.A.2 and [11]. In the event of a correctly working mask, the file size generally does not reach the expected 10MB size at an eavesdropper. As such, it was important to observe all indications, such as file size, sample counter block output, IQ plots, etc. in order to determine if an anomaly occurred during testing.

Example amplitude and in-phase and quadrature (IQ) graphs for unmasked, masked at intended receiver, and masked at an eavesdropper location are shown in Figure 29, Figure 30, and Figure 31.

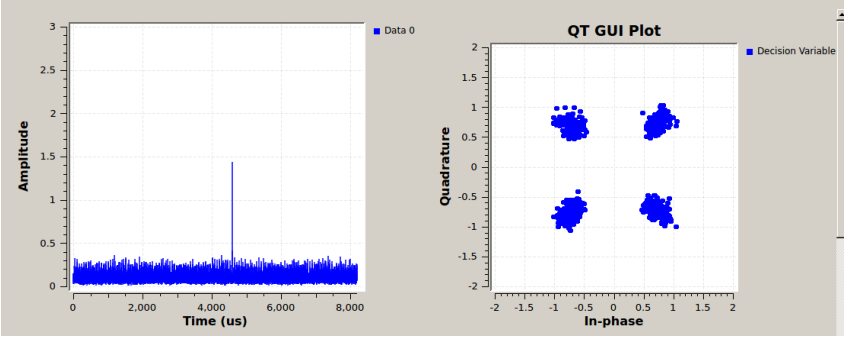

Figure 29. Example Amplitude and IQ Graph at Intended Receiver (unmasked, -4dB SMR)

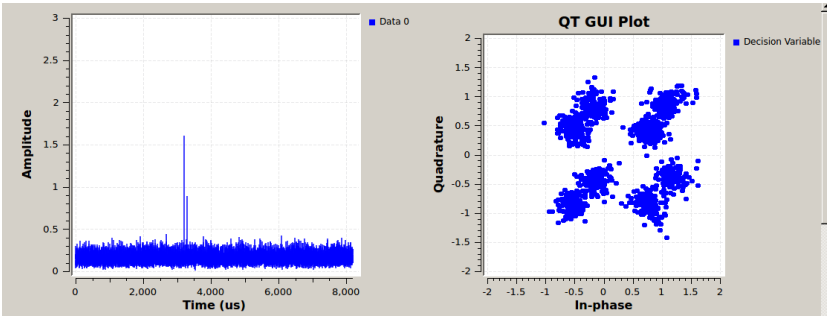

Figure 30. Example Amplitude and IQ Graph at Intended Receiver (masked, -4dB SMR)

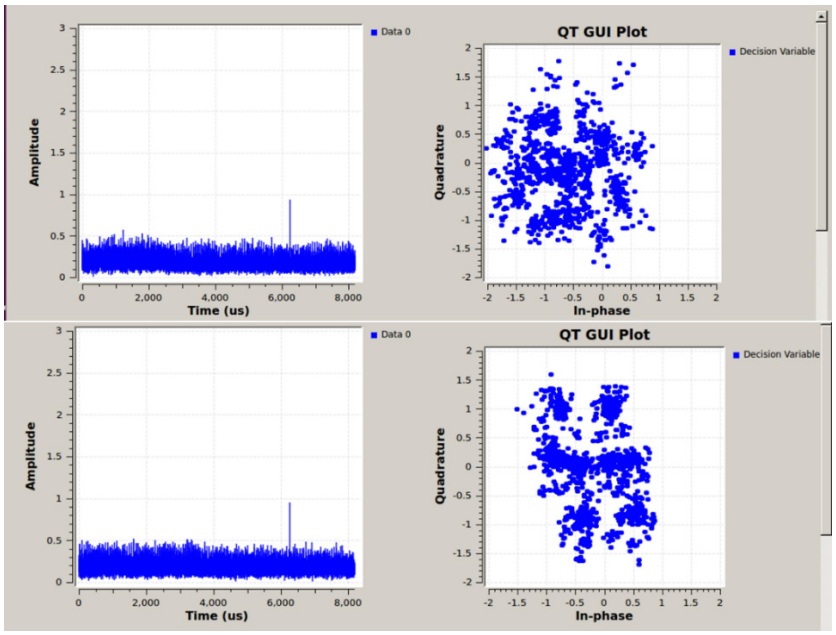

Figure 31. Examples of Amplitude and IQ Graph at Eavesdropper (masked, -4dB SMR)

Examining these amplitude and IQ graphs for anomalies during testing helped to determine whether transmission is occurring and if the mask appeared to be applied correctly. For example, if the transmitter terminal is emitting power, the amplitude box should show an appropriate response at the receiver. In Figure 29, the unmasked example shows that the received IQ constellation point for each QPSK symbol is tightly grouped, enabling data stream demodulation. The next example, Figure 30 shows that when the mask is applied, the spread of each constellation point increases. If the CSI were perfectly measured and the mask applied, then this example is expected to look more like the unmasked case of Figure 29. The system in this research calculates the CSI based on an average estimate of the real life time-varying and space-varying channel state [11]. Additionally, OTA propagation and computer processing delays contribute to the fact that the CSI is not perfectly applied to the mask calculation. Lastly, in Figure 31, we can see that if a sufficiently strong artificial noise mask is applied, the spread of the constellation points increases to a level whereby the eavesdropper will have great difficulty in demodulating the signal.

Some of the anomalies encountered during testing are illustrated below in Table 5. The table shows that it took 15 attempts in order to get two successful test runs with issues ranging from computer freezing to files not recording.

Table 5. SMR = +4dB Testing Attempts at the Eavesdropper

Attempt Number	Anomaly Description	File Size	Run Acceptance
1	Only header decoded, trailer position incorrect	10 MB	No
2	Incomplete File	2.5 kB	No
3	Only header decoded, trailer position incorrect	10 MB	No
4	Only header decoded, trailer position incorrect	10 MB	No
5	Incomplete File	4 MB	No
6	Only header decoded, trailer position incorrect	10 MB	No
7	Incomplete File	108 kB	No
8	File size slightly to small	9.9 MB	No
9	Header and Trailer indices correct	10MB	Yes
10	Incomplete File, did not appear to receive anything from transmitter via amplitude plot	10.1 kB	No
11	Receiver computer crash, restart required	N/A	No
12	Incomplete File	5 kB	No
13	Only Header index correct	10 MB	No
14	Unable to process file contents	10 MB	No
15	Transmit USRP frozen for ~20 seconds but Header and Trailer indices correct	10 MB	Yes

The particular conditions of the test in Table 5 were a random mask at an eavesdropper location with an SMR of +4dB. Note that the simulations in [3] indicate that this level of SMR should not result in a complete loss of transmission reception at the eavesdropper, just a higher BER. As such, one would expect to be able to decode the header and trailer of most transmissions rather than just two out of fifteen. Note that anomalies would occur in all cases, masked and unmasked as well as intended receiver and eavesdropper conditions.

B. OVER-THE-AIR TESTING

1. LOS Testing

The results of LOS testing at +/- 4 dB SMR are shown below in Table 6 and Table 7. The two SMR levels chosen mark the boundaries of the testing conducted in [10].

Table 6. LOS Over-the-Air Test Conditions and Resulting BER Performance Measurement for SMR = +4 dB

	Intended Receiver		
	No Mask	Static Mask	Random Mask
Run 1	5.04179e-4	2.77950e-4	9.74713e-4
Run 2	5.58213e-4	7.42875e-4	9.49275e-4
Average	5.31196e-4	5.10413e-4	9.61994e-4
	Eavesdropper		
Run 1	N/A	8.83813e-4	8.66975e-4
Run 2	N/A	8.76513e-4	7.68713e-3
Average	5.31196e-4	8.80163e-4	4.27705e-3

Table 7. LOS Over-the-Air Test Conditions and Resulting BER Performance Measurement for SMR = -4dB

	Intended Receiver		
	No Mask	Static Mask	Random Mask
Run 1	9.14125e-4	5.70625e-4	4.49112e-4
Run 2	5.60573e-4	6.26429e-4	6.57889e-4
Average	7.37349e-4	5.98527e-4	5.53501e-4
	Eavesdropper		
Run 1	N/A	Failure	Failure
Run 2	N/A	Failure	Failure

From the data presented above it can be observed that in both SMR conditions the mask appeared to have functioned as expected. When taken as a whole, the artificial noise appears to have little to no effect on the intended receiver since all conditions appeared to have resulted in a BER on the order of 10-4. It does seem that the random mask skewed the BER higher at the intended receiver in the +4dB case but this trend was absent during the -4 dB testing. This is counter-intuitive, since a higher SMR should result in a lower BER per [10]. This result may have been a product of the real world testing environment. Also, it is worthwhile to note that, in the +4 dB case, the Eavesdropper performed poorer than the intended receiver with both the static and random mask implementations. In the +4 dB case, the intended receiver required approximately 2-4 attempts to achieve two acceptable runs while the eavesdropper could require up to 15. The mask was given 5 chances to pass during the -4 dB SMR run for both static and random cases but only encountered failures, indicating a working mask. With the same setup but at a later time, the Static Mask was re-attempted for 15 runs at -4dB to verify if the header and trailer could demodulate correctly and all attempts failed again. This was done to revalidate the acceptance of a failure state for the eavesdropper under a -4dB SMR masking signal. Since the random mask performance appeared to be comparable, if not worse, than the static

mask performance at the eavesdropper, additional revalidation of those results were deemed unnecessary.

2. BLOS Testing

Reference [3] improved on previous work in the field of PLS by including BLOS capabilities in an artificial noise mask implementation. This improvement allows the mask to "pinpoint" the specific location of the receiver for cancellation via CSI rather than beamforming in their direction. As such, it is prudent to include testing data for the intended receiver in a BLOS condition. Ideally, the static and random mask should perform comparably to the unmasked case. The results are displayed in Table 8.

Note that for the BLOS test, the hardware for one transceiver appeared to experience extreme memory issues and, as such, the masking blocks were added directly to the receiver flow graph in an effort to deal with the performance issues. This was instead of the previous method of launching the receiver and mask calculations flow graphs in tandem via supervisor program. The memory issues appeared consistent with memory allocation problems noted in [11], however, the solution presented in that research did not resolve the symptoms. This change has no bearing on mask performance but was noted for completeness.

Table 8. BLOS Over-the-Air Test Conditions and Resulting BER Performance Measurement for SMR = +4dB

SMR: +4 dB	No Mask	Static Mask	Random Mask
Intended Receiver	6.00575e-4	6.38925e-4	3.71638e-4

From the data in Table 8, it appears as though the mask calculation in a BLOS communication channel is still adequate. The three cases of unmasked, static masked, and random masked achieved BERs on the order of 10^{-4}. Continued hardware issues precluded additional testing of the eavesdropper in a BLOS condition, however, the main goal of Table 8 was to illustrate that the mask cancelled at the intended receiver in both the LOS

and BLOS condition. Since the static and random masked results were comparable to the non-masked results at the intended receiver in both Table 6 and Table 8, the premise appears to been validated.

C. SUMMARY

In this chapter, the basic experimental hardware setup and testing operations were discussed and outlined. The methodology for testing and criteria for an acceptable test run were described along with examples of some of the anomalies encountered during testing. Rationale was provided for selecting specific OTA testing attempts based on the idea of equity between the attempts to be compared.

Examples of IQ patterns were shown along with a brief explanation of their meaning. OTA testing was performed to evaluate mask performance. Both LOS and BLOS cases were considered and the testing conditions covered the unmasked case as well as the masked at an intended receiver and an eavesdropper.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS

Through the application of OTA testing, this research shows that it is indeed possible to apply the artificial noise masking algorithm of [3] to the radio communication system of [11]. While there were issues encountered with the radio, to be discussed in Section A below, it was proven that under the right circumstances the masking signal will severely degrade the signal at an eavesdropper to the point at which it cannot be deciphered.

A. CHALLENGES

Overall, the testing successfully proved that it is possible to implement a BLOS radio with WPM and MIMO features such that only the intended recipient has sufficient SMR to decipher the transmitted message. That being said, the digital radio communication system that was created for testing is not without its faults.

As shown by anomalies discussed in Chapter VI, there were many issues preventing the designed radio from being a fully operational communications system. In its current state, the radio is not suitable for low rate production without further troubleshooting. While some of the issues appear to be memory related, as evidenced by processing delay, computers freezing, and the occasional memory segmentation error; the root cause of all the failed tests remains uncertain.

The radio clearly works under the correct conditions and effectively applies the artificial noise mask, however, future efforts would be required to smooth out the design and fix the bugs which are currently being experienced.

B. RECOMMENDATIONS

1. Improved Stabilization and Flow Control

First and foremost, future works should focus on stabilization of the current design or salvaging the usable code in a more stable environment. The anomalies inhibit further testing and prevent usage of the designed radio as a field-able communications solution. If the issues are solely memory related, a more powerful calculation library, such as machine learning tools like TensorFlow, could prove useful [66], [67]. The scripting used for the

python subprocess modules and the supervisor program may have caused some of the issues on one terminal. This indicated that a different methodology of controlling how various flow graphs are launched may be required as well. If the runtime issues are fixed, further development could build upon the spectrum sensing code to create an implementation of CSMA with Acknowledgements or build the initial setup message used to acquire CSI into a 3-way handshake protocol [14].

2. Full Duplex Testing

Full duplex testing was one of the initial driving factors for design considerations. Rather than wait for message exchange in order to calculate CSI from a received transmission, a full duplex system, akin to [52] and [51], could allow faster updates to the mask calculation. Enabling mask calculation in near real-time would potentially support communication on the move since the space-varying aspects of the channel could be better accounted for with sufficient CSI updates. This testing could also be used to show how accurate the CSI measurement must be in order to reach lower levels of SMR while still maintaining an acceptable BER at the intended receiver.

3. Carrier Frequency Offset

As discussed in Section II.B.3, channel reciprocity allows the assumption that the channel states between receiver and transmitter are identical when operating at the same frequency. As such, the effect of carrier frequency offset due to Doppler effect or local oscillator drift [11] would be important to test and characterize.

4. Deceptive Noise Mask

A further logical step in the noise mask calculation would be the application of a deceptive noise mask. The current mask contains two arbitrary equations that determine the mask received by the eavesdropper. It has currently been postulated that it may be possible to select those equations such that the resultant mask would decode as a distinct message to the eavesdropper while not interfering with the intended receiver. This would require alterations to the mask calculation code and, potentially, a low SMR operating point.

LIST OF REFERENCES

[1] H. Tao, *Sun Tzu's Art of War: The Modern Chinese Interpretation*. New York: Sterling PubCo, 1990.

[2] J. Mattis, *Summary of the 2018 National Defense Strategy*, Arlington, VA, USA, U.S. Department of Defense.

[3] R. C. Sellin, "A noise-masking physical layer security method for wavelet packet modulated multiple-input, multiple-out communication systems," M.S. thesis Dept. of Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, USA, 2018.

[4] M. Wills, *The Official (ISC)2 SSCP CBK Reference*. Hoboken, NJ: John Wiley & Sons, 2019.

[5] "Chapter 7: The role of cryptography in information security," Infosec Resources, 11 Jun 2012. [Online]. Available: https://resources.infosecinstitute.com/role-of-cryptography/. [Accessed: 04Dec2019]

[6] P. A. Regalia, A. Khisti, Y. Liang, and S. Tomasin, "Secure communications via physical-layer and information-theoretic techniques [Scanning the Issue]," *Proceedings of the IEEE*, vol. 103, no. 10, pp. 1698–1701, Oct. 2015.

[7] "Best readings in physical-layer security | IEEE Communications Society." [Online]. Available: https://www.comsoc.org/publications/best-readings/physical-layer-security. [Accessed: 04Dec2019]

[8] "What is defense in depth?," *Forcepoint*, 09 Aug 2018. [Online]. Available: https://www.forcepoint.com/cyber-edu/defense-depth. [Accessed: 04Dec2019]

[9] K. Ryland, M. Lichtman, and T. C. Clancy, "Implementation of two physical layer security techniques in an OTA system," *Proceedings of the GNU Radio Conference*, vol. 2, no. 1, pp. 9–9, Sep. 2017.

[10] R. C. Sellin and F. Kragh, "A noise-masking physical layer security technique for MIMO communications systems," in *MILCOM 2018*, 2018, pp. 536–540.

[11] M. R. Cribbs, "Multiple-input multiple-output wavelet packet modulation based software-defined transceiver design," M.S. thesis Dept. of Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, USA, 2015.

[12] B. S. Blanchard and W. J. Fabrycky, *Systems Engineering and Analysis*, 5th ed. Upper Saddle River, NJ: Prentice Hall.

[13] R. L. Freeman, *Telecommunication System Engineering*, 2nd ed. New York: Wiley, 1989.

[14] D. P. Agrawal, *Introduction to Wireless and Mobile Systems*, 3rd ed. Stamford, CT: Cengage Learning, 2011.

[15] S. Lin, *Error Control Coding: Fundamentals and Applications*, 2nd ed. Englewood Cliffs, N.J: Prentice-Hall, 2017.

[16] W. Stallings, *Data and Computer Communications*, 10th ed. Boston: Prentice Hall, 2014.

[17] E. Grayver, *Implementing Software Defined Radio*, 1st ed. 2013. New York, NY: Springer New York, 2013.

[18] A. B. Carlson and P. B. Crilly, *Communication Systems: An Introduction to Signals and Noise in Electrical Communication*, 5th ed. New York: McGraw-Hill, 2010.

[19] T. T. Ha, *Theory and Design of Digital Communication Systems*. New York: Cambridge University Press, 2011.

[20] A. Ghosh, J. Zhang, J. G. Andrews, and R. Muhamed, *Fundamentals of LTE*. Pearson Education, 2010.

[21] V. Savaux and Y. Louët, *MMSE-Based Algorithm for Joint Signal Detection, Channel and Noise Variance Estimation for OFDM Systems*, vol. 9781848216976. Wiley-ISTE, 2014 [Online]. Available: https://hal-supelec.archives-ouvertes.fr/hal-01073118. [Accessed: 14Oct2019]

[22] M. Guillaud, D. T. M. Slock, and R. Knopp, "A practical method for wireless channel reciprocity exploitation through relative calibration," in *Proceedings of the Eighth International Symposium on Signal Processing and Its Applications, 2005,* 2005, vol. 1, pp. 403–406.

[23] F. Kaltenberger, Haiyong Jiang, M. Guillaud, and R. Knopp, "Relative channel reciprocity calibration in MIMO/TDD systems," in *2010 Future Network & Mobile Summit*, 2010, pp. 1–10.

[24] J. A. C. Bingham, "Multicarrier modulation for data transmission: An idea whose time has come," *IEEE Communications Magazine*, vol. 28, no. 5, pp. 5–14, 1990.

[25] F. Farrukh, S. Baig, and M. J. Mughal, "Performance comparison of DFT-OFDM and Wavelet-OFDM with zero-forcing equalizer for FIR channel equalization," in *2007 International Conference on Electrical Engineering,* 2007, pp. 1–5.

[26] U. Khan, S. Baig, and M. J. Mughal, “Performance comparison of wavelet packet modulation and OFDM for multipath wireless channel,” in *2009 2nd International Conference on Computer, Control and Communication*, 2009, pp. 1–4.

[27] A. R. Lindsey and J. C. Dill, “Wavelet packet modulation: A generalized method for orthogonally multiplexed communications,” in *Proceedings of the Twenty-Seventh Southeastern Symposium on System Theory*, 1995, pp. 392–396.

[28] A. R. Lindsey, “Wavelet packet modulation for orthogonally multiplexed communication,” *IEEE Transactions on Signal Processing*, vol. 45, no. 5, pp. 1336–1339, 1997.

[29] International Conference on Wavelet Analysis and Its Applications, *Wavelet Analysis and Applications*, 1st ed. 2007. Basel: Birkhäuser Basel, 2007.

[30] Juan Fang, Zihao You, I-Ta Lu, Jialing Li, and Rui Yang, “Comparisons of filter bank multicarrier systems,” in *2013 IEEE Long Island Systems, Applications and Technology Conference (LISAT),* 2013, pp. 1–6.

[31] H. Nikookar, *Wavelet Radio: Adaptive and Reconfigurable Wireless Systems Based on Wavelets*. Cambridge: University Press, 2013.

[32] M. K. Lakshmanan and H. Nikookar, “Construction of optimal bases for wavelet packet modulation under phase offset errors,” in *3rd European Wireless Technology Conference*, 2010, pp. 293–296.

[33] E. Kjeldsen and A. Lindsey, “Receiver timing recovery for adaptive wavelet packet modulated signals,” in *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, 2002, vol. 1, pp. 92–96 vol.1.

[34] Y. Sun and J. Hao, “Decision-aided extended kalman filter based adaptive timing recovery for wavelet packet modulated signals,” *Tsinghua Science & Technology*, vol. 15, no. 3, pp. 329–334, 2010.

[35] J. S. Walker, *A Primer on Wavelets and Their Scientific Applications*. Boca Raton, Fla: Chapman & Hall/CRC, 1999.

[36] M. R. Cribbs, “Multiple-input multiple-output wavelet packet modulation based software-defined radio transceiver design,” LTS, 03-Sep-2015.

[37] G. D. Durgin, *Space-Time Wireless Channels*. Upper Saddle River, NJ: Prentice Hall PTR, 2003.

[38] E. R. Brand a National Instruments, “SDR software,” *Ettus Research*. [Online]. Available: https://www.ettus.com/sdr-software/. [Accessed: 18Oct2019]

[39] “FAQ – GNU Radio.” [Online]. Available: https://wiki.gnuradio.org/index.php/FAQ. [Accessed: 18Oct2019]

[40] L. Zhang, “Implementation of wireless communication based on software defined radio,” M.S. Thesis Department of Audiovisual Engineering and Communications, Polytechnic University Of Madrid, Madrid, Spain, 2013.

[41] “Guided tutorial introduction – GNU Radio.” [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorial_Introduction. [Accessed: 18Oct2019]

[42] “Simplified wrapper and interface generator.” [Online]. Available: http://swig.org/. [Accessed: 18Oct2019]

[43] D. Shen, “Tutorial 10: Writing a signal processing block for GNU Radio – Part I.” 11Jun2005 [Online]. Available: http://www.snowymtn.ca/GNURadio/GNURAdioDoc-10.pdf

[44] “GNU Radio manual and C++ API reference: Main page.” [Online]. Available: https://www.gnuradio.org/doc/doxygen/. [Accessed: 18Oct2019]

[45] J. Corgan, “GNU Radio runtime operation,” Aug2015 [Online]. Available: https://static1.squarespace.com/static/543ae9afe4b0c3b808d72acd/t/55de1259e4b01e5c160764cf/1440617049937/5.+corgan_johnathan-scheduler+2015-08-25.pdf

[46] “Message passing - GNU Radio.” [Online]. Available: https://wiki.gnuradio.org/index.php/Message_Passing. [Accessed: 18Oct2019]

[47] T. Rondeau, “Scheduler details,” 26Sep2013 [Online]. Available: http://www.trondeau.com/blog/2013/9/15/explaining-the-gnu-radio-scheduler.html

[48] J. Malsbury, “Modular, open-source software transceiver for PHY/MAC research,” in *Proceedings of the second workshop on software radio implementation forum*, 2013, pp. 31–36.

[49] “OutOfTreeModules – GNU radio.” [Online]. Available: https://wiki.gnuradio.org/index.php/OutOfTreeModules. [Accessed: 18Oct2019]

[50] M. Belleschi, “Cross-layer optimization protocols in Ad-hoc networks: analysis and practical implementation of transport and network layers,” M.S. thesis Dept. of Elect. Eng., KTH, Stockholm, Sweden, Mar. 2008.

[51] A. Parower, “A GNU radio-based full duplex radio system,” GRCON17, 13Sep2017.

[52] T. Chen, M. B. Dastjerdi, J. Zhou, H. Krishnaswamy, and G. Zussman, "Wideband full-duplex wireless via frequency-domain equalization: Design and experimentation," 20181203 [Online]. Available: http://arxiv.org/abs/1812.01126. [Accessed: 20Oct2019]

[53] "GitHub," *GitHub*. [Online]. Available: https://github.com. [Accessed: 20Oct2019]

[54] "CGRAN." [Online]. Available: http://cgran.org/. [Accessed: 20Oct2019]

[55] A. F. Y. Mohammed, "Studying media access and control protocols," 2010 [Online]. Available: http://kth.diva-portal.org/smash/get/diva2:508792/FULLTEXT01.pdf. [Accessed: 20Oct2019]

[56] J. Malsbury, *jmalsbury/gr-mac*. 2019 [Online]. Available: https://github.com/jmalsbury/gr-mac. [Accessed: 20Oct2019]

[57] V. González-Barbone, P. Belzarena, and F. Larroca, "Software defined radio: From theory to real world communications," in *2018 XIII Technologies Applied to Electronics Teaching Conference (TAEE)*, 2018, pp. 1–7.

[58] "vagonbar/gr-gwn," *GitHub*. [Online]. Available: https://github.com/vagonbar/gr-gwn. [Accessed: 20Oct2019]

[59] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste, "Enabling MAC Protocol implementations on software-defined radios," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2009, pp. 91–105 [Online]. Available: http://dl.acm.org/citation.cfm?id=1558977.1558984. [Accessed: 20Oct2019]

[60] E. Tollefson, B. R. Jordan, and J. D. Gaeddert, "Out-phased array linearized signaling (OPALS): A practical approach to physical layer encryption," in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, 2015, vol. 2015-, pp. 294–299.

[61] "gnuradio.zeromq – GNU radio 3.7.9.1 documentation." [Online]. Available: https://www.gnuradio.org/doc/sphinx-v3.7.9.1/zeromq_blocks.html. [Accessed: 02Dec2019]

[62] "Probe Avg Mag^2 – GNU radio." [Online]. Available: https://wiki.gnuradio.org/index.php/Probe_Avg_Mag%5E2. [Accessed: 02Dec2019]

[63] "GNU Radio manual and C++ API reference: gr::blocks::file_source Class Reference." [Online]. Available: https://www.gnuradio.org/doc/doxygen/classgr_1_1blocks_1_1file__source.html#a3efcf0d0acb9038eb7e2f242564561a6. [Accessed: 02Dec2019]

[64] “17.1. subprocess — Subprocess management — Python 2.7.17 documentation.” [Online]. Available: https://docs.python.org/2/library/subprocess.html. [Accessed: 03Dec2019]

[65] “Selecting a RF daughterboard – Ettus knowledge base.” [Online]. Available: https://kb.ettus.com/Selecting_a_RF_Daughterboard. [Accessed: 03Dec2019]

[66] R. Python, “Pure Python vs NumPy vs TensorFlow performance comparison – Real Python.” [Online]. Available: https://realpython.com/numpy-tensorflow-performance/. [Accessed: 05Dec2019]

[67] “TensorFlow,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/. [Accessed: 05Dec2019]

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California