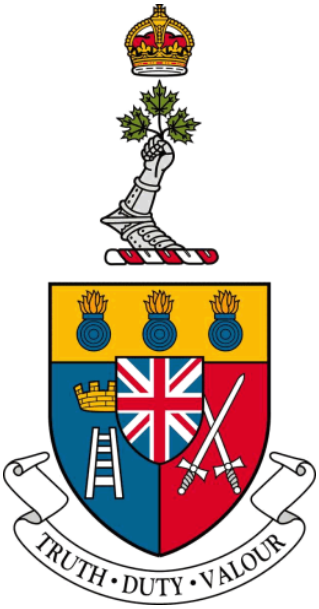
Royal Military College of Canada

Department of Electrical and Computer Engineering
EEE455/7 Electrical and Computer Engineering Design Project



DID-07 – Detailed Design Document
Image Transmission Using an SDR Based MIMO System

Presented by:
OCdt Thomas Jung
OCdt Eric Kim

Presented to:
Dr. Mostafa Hefnawi
Maj. John Lloyd

April 7, 2020

Contents

List of Figures

List of Tables

List of Design Codes

1 Introduction

1.1 Document Purpose

This document provides a detailed design description for the software-defined radio (SDR) based multiple-input-multiple-output (MIMO) system. The purpose of this document is as follows:

(a) to provide a description of the project's design methodology;

(b) to provide an explanation and justification for the changes and deviations from the originally planned project, in reference to the Statement of Requirements (SOR), the Preliminary Design Specification (PDS), and the Schedule Update;

(c) to present the details of the final project design;

(d) to present the testing and validation results of the project;

(e) to provide a summary of the degree of success of the project; and

(f) to provide suggestions and avenues for future project expansions.

1.2 Background

Multiple-input-multiple-output (MIMO) technology is a key component in both current and next generation communications systems. Achieving space diversity through numerous antennas enhances performance and efficiency when transferring data. Similarly, advances in digital electronics and increases in computing power have rendered practical the use of software-defined radios (SDR) for applications that have been historically implemented in hardware. Merging these two technologies results in adaptive communications systems. Known as cognitive radios, such systems can, depending on the state of the transmission medium, alter in real time the transmission and reception schemes in the digital signal processing (DSP) chains to make the most efficient use of the frequency spectrum while achieving the best possible performance. Software-defined radios have brought cognitive radios closer to reality, as many of the signal processing components can be realized and reprogrammed in software.

The motivation behind this project was to design and implement a 2x2 MIMO system to wirelessly transmit data through two SDRs as a starting point to the greater concept that is the cognitive radio. This project aims to explore the implementation of a functional system on the Universal Software Radio Peripheral (USRP) X310 using MATLAB's Simulink environment.

The scope of this project is bounded to using the Alamouti orthogonal space time block code (OSTBC) scheme along with two modulation schemes - quadrature and binary phase shift keying (Q/BPSK) - to transmit a Joint Photographic Experts Group (JPEG) image over a wireless channel. The designed communication protocol is unique to this project and is not restricted to any industry standard.

1.3 Design Methodology

In the requirement design phase described in the Statement of Requirement (SOR) [1], each step of the transmitter and receiver signal processing chains were listed as functional requirements, as all components require functionality for the overall system to be operational. The interface and implementation requirements were constrained by the SDR models that were readily available. The performance requirements such as the bit error rate (BER) and spectral efficiency were selected based on an examination of modern communication standards as outlined in the preliminary design specification (PDS) [2].

The preliminary design phase consisted of developing and understanding of the various Simulink tools and preexisting models for the SDR. As outlined in the PDS, the incremental model guided the project. The application of this structure saw the creation and testing of various subsystems in virtual simulations, after which each verified component would be integrated into an overall system to eventually be realized in hardware. This process was repeated to incrementally develop the final design, as seen in Figure 1.1.

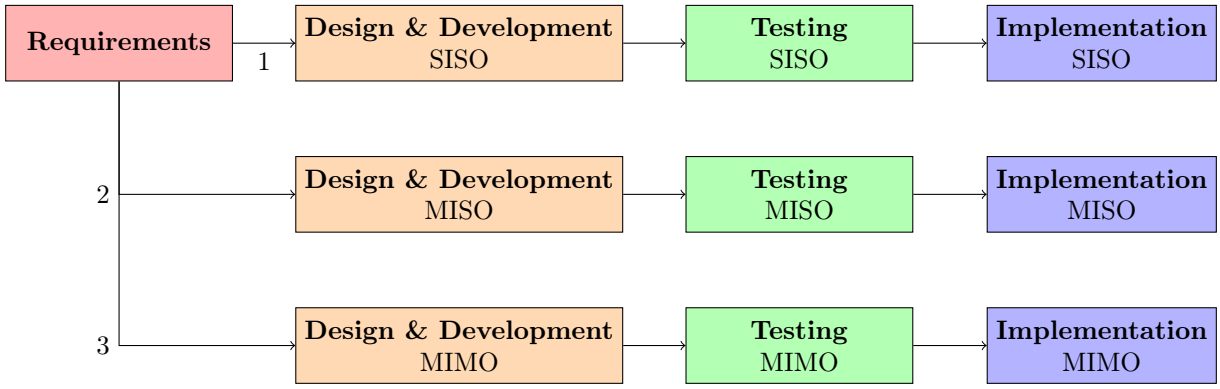


Figure 1.1: Design Method: Incremental Model

1.4 Hardware and Setup

To implement and test preliminary systems as well as the final design in hardware, two National Instrument USRP X310 SDRs were used [3]. These devices were connected via 10-Gigabit Ethernet cables to two separate Windows 10 desktop computers (Figure 1.2). One SDR was used for the transmitter and the other for the receiver, with each device connected to two Ettus Research VERT 2450 antennas. The separation of the transmitting and receiving SDRs allowed for the respective Simulink models to compile and execute relatively quickly, as opposed to running both signal processing chains on the same computer. The signal processing chains were created in the MATLAB 2019b Simulink environment.

Figure 1.2: Hardware

1.5 Approach

Simulink's pre-existing single-input-single-output (1x1 SISO) USRP QPSK model was used as a starting point to understanding the functioning of several key components of the transmitter and receiver modules. Testing this system provided confirmation of the correct functioning of various Simulink sub-modules and also presented a working realization of a complete signal processing chain, from which the more complex systems could be built.

After exploring the 1x1 system, the 2x1 MISO system provided a method of incorporating the Alamouti OSTBC scheme in an incremental fashion. Finally, the 2x2 MIMO system saw the addition of a second receiver antenna. However, it was found that despite the value of the 1x1 QPSK model as a foundation, the increased complexity and functionality from adding another transmitter antenna in a 2x1 system posed various new challenges.

1.6 Obstacles

The 2x1 MISO phase of the project introduced new components in the signal processing chains that had previously not been required in the 1x1 system. Specifically, the successful implementation of the Alamouti orthogonal space time block code (OSTBC) decoding and preamble detection blocks was a considerable obstacle, as these elements were integral to the proper recovery of the transmitted data. These blocks were tested separately in simulations that were created in the Simulink environment. The realized solutions to these difficulties will be detailed in their respective subsections in the design and simulation parts of this document.

2 MIMO Design Architecture Overview

The final design is comprised of two modules: the transmitter signal processing chain and the receiver signal processing chain. The transmitter module accepts a JPEG image, converts it to binary data, packages the information for transmission over a wireless channel, and sends the data through the transmitting SDR. The receiver module takes the signal obtained through the antenna, converts it to digital information, and reconstructs the transmitted image. The overview of the system architecture can be seen in Figure 2.1.

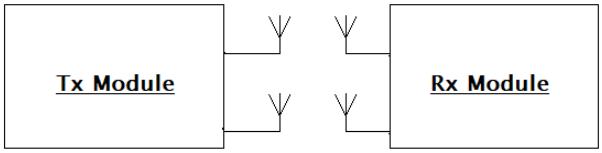


Figure 2.1: System Architecture Overview

3 MIMO Transmitter Module Design

3.1 Transmitter Module Overview

The overall design of the transmitter module can be seen in Figure 3.1 below. The module involves an image to bitstream converter, a Q/BPSK modulator, an Alamouti OSTBC encoder, a square root raised cosine filter, and the USRP transmitter interface block.

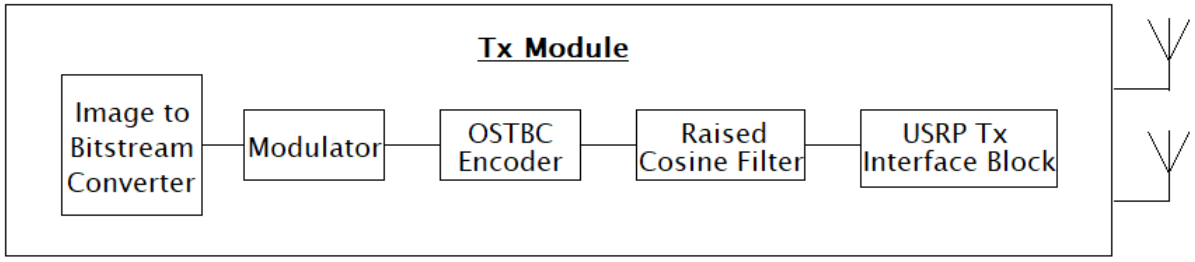


Figure 3.1: Transmitter Module Overview

3.2 Image to Bitstream Conversion

The design focus of the transmitter signal processing chain is the image to bitstream converter, as the other components in the module were available from Simulink's toolbox. The first design decision that required addressing was selecting the input image format to one that was suitable for data transmission. The JPEG format input requirement, as specified in the SOR, was decided on for various reasons.

A JPEG image is a raster image, which means that the image data is stored in a matrix, with each discrete value representing a pixel. This structure facilitates matrix manipulation of the image in MATLAB and Simulink, as opposed to modifying a vector image. In addition, the ability to access an image's data in matrix form was useful for debugging during the testing and verification phases. The JPEG format is well supported by the Windows operating system, as well as the MATLAB environment, which had pre-existing tools and functions that were used to package the image data for transmission. This ease of integration

removed the requirement to convert the input image to another data format before sending it through the transmitter chain. Finally, JPEG is ubiquitous. It is a common file type, which provides flexibility while keeping the overall system relevant to the standards of current technology.

Typically, any data that is to be wirelessly transmitted is packaged into a specific structure consisting of groups of information bits called frames. Frames are comprised of information bits and a series of extra bits called the preamble that is used to identify and distinguish each frame. The values of these extra bits are dependent on the communication standard known to both the transmitter and the receiver.

The initial design approach was to transmit the entire image in a single frame, with one preamble at the beginning. However, Simulink's preamble detector, one component in the receiver module, was proven to have an inconsistent output when receiving frames containing more than 100 000 bits. For perspective, a 480x480 8-bit JPEG image has over 5.5 million bits. This issue, shown in simulation and further detailed in Section 5.1.3, resulted in the design decision of selecting a frame length that could be managed by the receiver's preamble detector.

To transmit images of varying dimensions, it was realized that the transmitted frame size should stay constant. For instance, if the frame length was dependent on the width of the input image, the frame size may exceed the experimentally determined limits imposed by the preamble detector. Therefore, by experimenting with various frame lengths, it was decided to fix the data frame size to 40 000 bits. It should be noted that this value, when divided by 8, results in an integer. This is important, as it signifies that regardless of whether or not each transmission frame represents a row of the image, the frames will never contain a fraction of an 8-bit RGB value. This data organization structure means that the number of transmitted frames will differ depending on the size of the image. Lastly, the final data frame of an image will not necessarily be 40 000 bits long unless the image dimensions are exactly divisible by 40 000, which is unlikely. Therefore, the last frame will contain extra zero bits as padding, as Matlab can only manipulate rectangular matrices.

When transmitting data over a non-ideal channel, bit errors, that is receiving a '0' when a '1' was transmitted or vice versa, are expected. Knowing that not all pixel values of an image will be perfectly received, it was decided to continuously transmit the image to add system robustness. This way, even if one part of the image bit sequence experiences significant errors, there is a chance that the same sequence will not be as severely affected during the next transmission cycle. Although only one instance of the image is extracted from the total received bitstream, this module could be expanded on to best reconstruct the image using multiple samples. However, cyclical transmission introduced the problem of determining the starting and ending points of a cycle.

To resolve this issue, a preamble is appended to the beginning of each frame. This preamble is a 11-bit unipolar Barker sequence. Unipolar Barker sequences are unique combinations of ones and zeros that produce high correlation values when autocorrelated. In other words, Barker codes are useful for finding other instances of the same code in an overall bitstream. The preamble detector in the receiver module uses this known Barker code to find the locations in the bitstream at which the code appears. The sequence is a digital marker that identifies frames, which allows the receiver to gather the transmitted data and reconstruct the image.

Intuitively, longer Barker codes have a lower probability of appearing in a sequence of data bits. To provide a mathematical argument, the bits representing the image data will be modelled as independent random variables, with the probability of receiving a 0 or 1 being equal. The binomial distribution can then be used to calculate the probability of receiving a specific sequence of bits (3.1).

$$P(x) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x} \tag{3.1}$$

$$\text{(Unipolar) Barker code } 11 = 1\,1\,1\,0\,0\,0\,1\,0\,0\,1\,0 \tag{3.2}$$

Considering Barker sequence 11 (line 3.3), the probability that 5 ones and 6 zeros are received is shown below. This is the probability that ones and zeros will appear in the specific sequence of the length 11 Barker code.

$$\begin{aligned} P(11) &= \frac{11!}{11!(11-11)!} 0.5^{11} (1-0.5)^{11-11} \\ &= 0.0004883 \end{aligned} \tag{3.3}$$

Similarly, the probability of a shorter length Barker code (line 3.4) appearing in the image bitstream is shown below.

$$\text{(Unipolar) Barker code } 4 = 1\,0\,1\,1 \tag{3.4}$$

$$\begin{aligned} P(4) &= \frac{4!}{4!(4-4)!} 0.5^{4} (1-0.5)^{4-4} \\ &= 0.0625 \end{aligned} \tag{3.5}$$

The difference in the probabilities determined above justifies the use of longer Barker codes in the preamble. The use of longer preambles lowers the data rate of a system, but the amount is negligible and is worth the tradeoff if there is a much lower chance of accidentally detecting a Barker sequence in the data stream, where there is not meant to be a preamble.

Simulink's QPSK modulation and OSTBC encoding components require even-numbered input vectors. After QPSK modulation, the frame size is halved, as each symbol represents two bits. Similarly, OSTBC also decimates the frame size by 2, because one symbol is transmitted from each of the two antennas. Since using longer Barker sequences is advantageous, it was decided to use Barker code 11 and append an extra bit 0 to the end to force an even-lengthed sequence. The option to simply repeat the code twice was not possible, as the overall frame had to be of an even length after going through both the modulator and OSTBC encoder, which dictated that the quotient of the frame size when divided by 4 be an integer. It is important to note that the addition of this extra bit does not affect the efficacy of the preamble detector at the receiver, as the detector still searches for the length 11 Barker sequence. The extra bit is ignored, and only serves to meet Simulink's stringent array size requirements.

To reconstruct the image at the receiver, the receiver must know the input image's dimensions. Therefore, the image dimensions are appended to the start of the data frame in a location consistent to the unique communication standard that was designed. It was decided to place the image dimension values one after another, and appending extra zero bits to the end of the data to reach the standard frame size of 40 000 bits. This decision was made to accommodate any other image data that the user may wish to transmit. Although not directly implemented into the final design, this functionality can be further developed.

Figure 3.2 below illustrates the organization of the data bits and Barker sequences as detailed previously. It represents the communication standard designed for the final system. The input image is selected in an initialization Matlab script that has been programmed to run at the start of every Simulink run. This script converts the input image into a bitstream before importing the data to the Simulink environment, where it goes through the rest of the transmitter signal processing chain. The code can be found in the Appendix: Design Code 1.

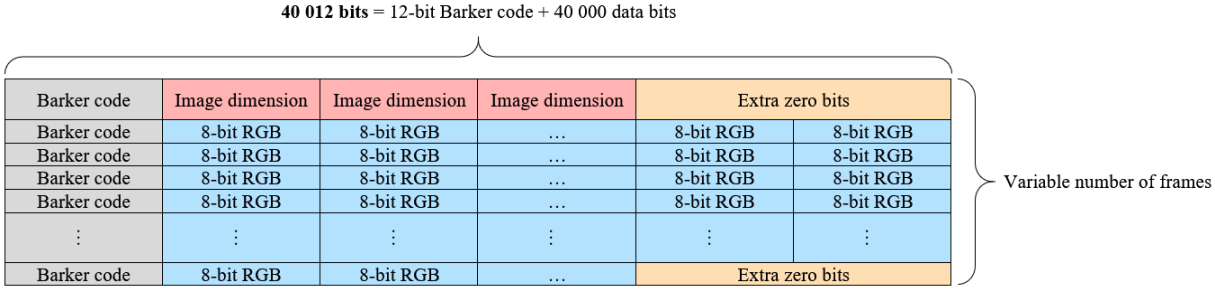


Figure 3.2: Transmitter Structure Bitmap

3.3 Modulation

After packaging the image data into a bitstream, the information must be modulated for over-the-air transmission. In QPSK modulation, every two bits are represented by one symbol, whereas in BPSK, every bit is represented by one symbol. Furthermore, for the QPSK modulator, it was decided to incorporate Gray coding. This type of symbol representation, where adjacent symbols only change by a single bit, reduces the number of bit errors. For instance, if a certain symbol is received, but the demodulator makes an error when identifying the received symbol, there is the greatest probability that an adjacent symbol is selected. If Gray coding is used, although the symbol neighbouring the correct one was selected, there is only a difference of one bit. In this case, there is one erroneous bit instead of possibly two, when Gray code is not used. As such, the number of incorrectly received bits can be reduced by half. The modulation constellations for both schemes are shown below (Figures 3.3 and 3.4). The preexisting Simulink modulation components were integrated without difficulty. Since the scope of the project involves using both BPSK and QPSK schemes, a virtual switch was integrated into the design to quickly change the modulation scheme.

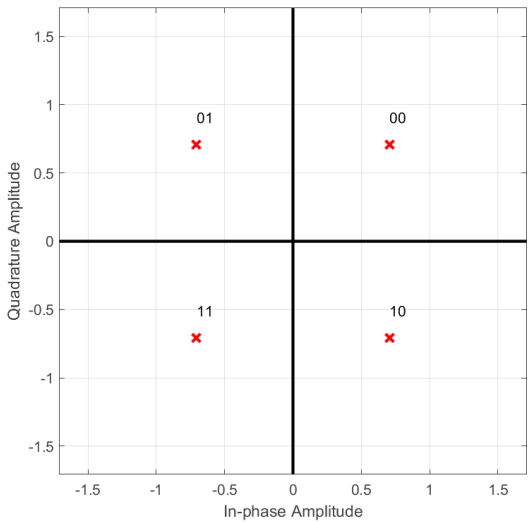


Figure 3.3: QPSK Symbol Constellation

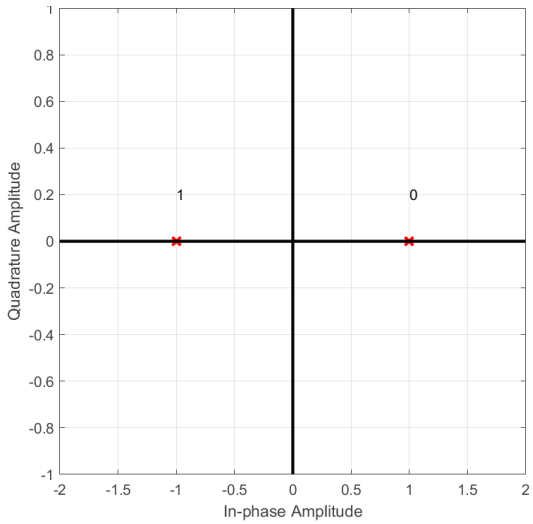


Figure 3.4: BPSK Symbol Constellation

3.4 OSTBC Encoding

After the modulator, the outgoing signals are encoded in accordance to Alamouti's OSTBC scheme. Multiple versions of the same signal are transmitted from two spatially-separated antennas. Since the paths taken by the signals from each antenna are affected differently by the channel, mathematically estimated values of the attenuation and phase delay introduced by the channel can be used to reconstruct the transmitted signal at the receiver. In Figure 3.5 below, the signals s_1 and s_2 are simultaneously transmitted at time t_1, after which mathematically altered versions of the signals $-s_2^*$ and s_1^* are transmitted at the next sampling instance t_2. This separation or orthogonalization is illustrated in Table 3.1. The received signals, denoted by variables r_1, r_2, r_3, and r_4 (Table 3.2), will be further explained in Section 4.3.

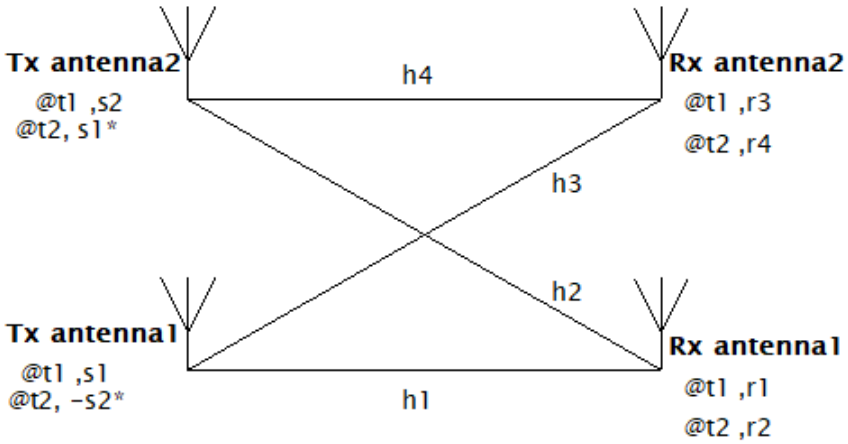


Figure 3.5: MIMO Antenna and Signal Configuration

Time	Antenna 1	Antenna 2
t_1	s_1	s_2
t_2	$-s_2^*$	s_1^*

Table 3.1: Transmitted Signals

Time	Antenna 1	Antenna 2
t_1	$r_1 = h_1 s_1 + h_2 s_2$	$r_3 = h_3 s_1 + h_4 s_2$
t_2	$r_2 = -h_1 s_2^* + h_2 s_1^*$	$r_4 = -h_3 s_2^* + h_4 s_1^*$

Table 3.2: Received Signals

Depending on the modulation scheme, the effect of the OSTBC encoder on each symbol will differ. Changing the sign of a QPSK-modulated symbol represents a 180° rotation on the constellation map, and taking the complex conjugate will simply reflect the symbol along the real axis. This is illustrated in Figure 3.3 below. For BPSK-modulated symbols, taking the complex conjugate will have no effect, and changing the sign will still represent a 180° rotation on the constellation map (Figure 3.4).

The mathematical operations of OSTBC encoding [4] is conducted automatically by the Simulink component. The received signals are further explained in Section 4.3.

3.5 Transmission Interfacing

The raised cosine filter is inserted at the end of the transmitter signal processing chain to reduce inter-symbol interference (ISR) [5]. This component is important for achieving a lower bit error rate. Finally, the USRP transmitter sub-module conducts the interfacing between Simulink and the SDR and transmits the processed waveforms through the antennas.

3.6 Transmitter Module Implementation

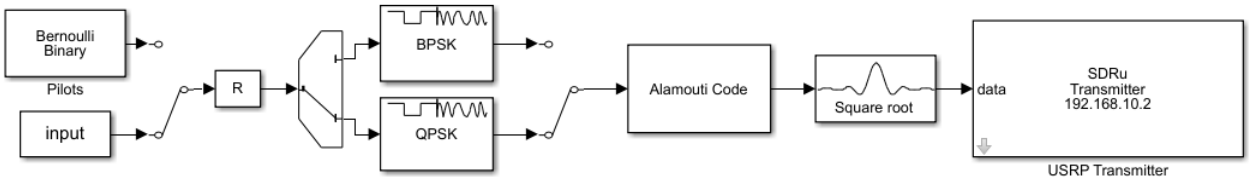


Figure 3.6: Transmitter Module Simulink Design

Figure 3.6 displays the full transmitter signal processing chain. The sequence and existence of each sub-module was designed to be the most logical while achieving the best performance. The input switch from the Bernoulli pilots to the image input in Figure 3.6 was implemented for ease of access between conducting offline channel estimation and transmitting the image, which is explained in Subsection 4.3. The reshaping (R) block between the input and modulator is required to organize the input image into an array with dimensions compatible with Simulink. Finally, the switches surrounding the modulation types allow the user to quickly change between QPSK and BPSK modulation schemes. The SDRu Transmitter is a module through which Simulink interfaces with the SDRu using the IP address.

4 MIMO Receiver Module Design

4.1 Receiver Module Overview

As seen in Figure 4.1 below, the receiver module consists of an automatic gain controller (AGC), a raised cosine filter, an OSTBC decoder, frequency and symbol synchronizers, a preamble detector, a frame synchronizer, an image converter, and finally a bit-error-rate (BER) calculator. The 1x1 USRP QPSK Simulink model introduced many of the components required for the implementation of a receiver capable of receiving and decoding spaced-time coded signals. Although some sub-modules did not present any issues when migrating from a 1x1 system to a 2x1 system, others required considerable alteration and tuning to be integrated into the overall design.

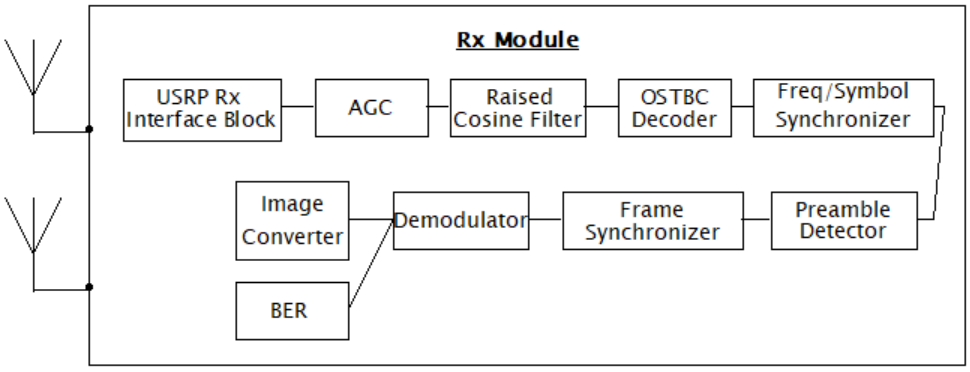


Figure 4.1: Receiver Module Overview

4.2 Minor Sub-Components

The USRP interface sub-module, automatic gain controller, raised cosine filter, and QPSK demodulator were integrated without difficulty. These blocks are independent of whether or not the received signals are encoded, and their functionality had been tested using the 1x1 USRP QPSK model. After receiving the signal from the USRP interface component, the automatic gain controller continuously adjusts the gain at the receiver to stabilize the power of the received signals, and the raised cosine filter reduces inter-symbol interference. Finally, demodulation occurs after the signals have been decoded. Therefore, the integration of the demodulator sub-module is identical in both SISO and MISO/MIMO systems. These systems are minor in the sense that their implementations presented few or no obstacles.

4.3 OSTBC Decoding

In order to decode the OSTBC signals at the receiver, an estimation of the channel must be conducted. Channel estimation is a mathematical operation that combines the received signals to extract information about the degree of interference, delay, and fading in the channel. These channel coefficients are required to recover the transmitted signals. The following description of the method used to conduct OSTBC decoding is a primary design component.

The approach used to estimate the channel was to transmit a sequence of pilot symbols, in this case a stream of 1s, and compare the sequence to what is received after the signal travels through the channel. This is shown in Table 4.1. The complex conjugate signals can be simplified, because all symbols lie on the real axis when using BPSK modulation, and for QPSK modulation, the phase offset can be adjusted so that the symbol representing a pair of 1s will lie on the real axis as well.

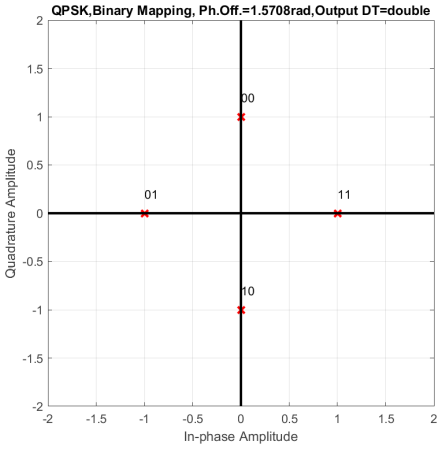


Figure 4.2: QPSK Constellation: Binary and $\frac{\pi}{2}$ Phase Offset

Time	Antenna 1	Antenna 2
t_1	$s_p = 11$	$s_p = 11$
t_2	$-s_p^* = -s_p$	$s_p^* = s_p$

Table 4.1: Transmitted Signals: Binary and $\frac{\pi}{2}$ Phase Offset

Time	Antenna 1	Antenna 2
t_1	$r_1 = h_1 s_p + h_2 s_p$	$r_3 = h_3 s_p + h_4 s_p$
t_2	$r_2 = -h_1 s_p + h_2 s_p$	$r_4 = -h_3 s_p + h_4 s_p$

Table 4.2: Received Signals: Binary and $\frac{\pi}{2}$ Phase Offset

In reference to Figure 3.5, the transmitted signals s_1 and s_2 will be affected by the channel parameters h_1, h_2, h_3, and h_4. At this point, the receiver is unable to distinguish which signal originated from which antenna, so the receiver only detects signals r_1, r_2, r_3, and r_4. However, the binary and $\frac{\pi}{2}$ phase offset in the QPSK modulation scheme setting simplifies the computation of the channel parameters from the received signals. The channel parameters, h_1, h_2 were retrieved by adding and subtracting signals r_1 and r_2 respectively (Equations 4.5, 4.6). Similarly, the channel parameters, h_3, h_4 were retrieved by adding and subtracting signals r_3 and r_4 respectively (Equations 4.3, 4.4). The parameter A is simply a gain factor to normalize the channel estimation parameters to 1.

$$h_1 = -A(-r_1 + r_2) \tag{4.1}$$
$$h_2 = A(r_1 + r_2) \tag{4.2}$$
$$h_3 = -A(-r_3 + r_4) \tag{4.3}$$
$$h_4 = A(r_3 + r_4) \tag{4.4}$$

Once computed, the offline channel estimation parameters are used to decode the incoming signals in accordance to Alamouti's decoding scheme, which exists inherently in the Simulink OSTBC decoder component. This process is shown below in lines 4.5 and 4.6.

$$\hat{s_1} = h_1^* r_1 + h_2 r_2^* + h_3^* r_3 + h_4 r_4^* \tag{4.5}$$

$$\hat{s_2} = h_2^* r_1 - h_1 r_2^* + h_4^* r_3 - h_3 r_4^* \tag{4.6}$$

Next, the maximum likelihood detector in the OSTBC decoder determines s_1 and s_2 by choosing the constellation that is closest in distance to the decoded signals, $\hat{s_1}$ and $\hat{s_2}$. For instance, if the decoded symbol is $0.9 + j0.9$, the maximum likelihood detector would chose the received symbol to be $1 + j1$. As long as the decoded symbol is close enough to the transmitted symbol, the maximum likelihood detector is able to correctly recover the symbol. For cases where the decoded symbol whose constellation is equal in distance to two different possible coordinates, or is closer to an incorrect coordinate, it is inevitable that the receiver will incorrectly recover the incoming signals. This highlights the importance of obtaining accurate channel estimation parameters to ensure that the decoded symbols whose constellations are as close to their expected locations, to minimize the bit-error rate.

Real-time channel estimation, which is a technique of continuously updating the channel estimation parameters as one of the inputs to the OSTBC decoder using the pilots in each frame, proved to be difficult to implement. Real-time channel estimation provides a degree of system robustness, as the system can adapt to abrupt changes in the channel, for instance due to the presence of an obstacle or atmospheric changes. Through testing and simulation, it was realized that this active estimation heavily depends on the consistent functionality of the preamble detector, which is usually placed before the OSTBC decoder in the receiver chain. However, further testing revealed that detecting modulated and OSTBC encoded pilot symbols was not a simple task that yielded consistent results. Moreover, such a responsive system using real-time channel estimation is beyond the project scope, which is detailed in the SOR. Since the final design is to be demonstrated in an indoor environment where the channel does not fluctuate significantly, it was decided to make a trade-off between system robustness and complexity by foregoing real-time channel estimation.

The alternative to real-time channel estimation is to obtain the channel estimation parameters prior to the transmission of the data. Known as offline channel estimation, this technique assumes that the channel will stay relatively constant between the transmission of the pilot symbols and the data. To implement such a system, the final design saw the placement of the preamble detector sub-module after the Alamouti OSTBC decoder. This way, the inconsistencies of the detector would not affect the signal decoding, and the data can progress through the rest of the processing chain. This deliberate arrangement of components and the writing of the decoding functions were one of the main design efforts in the receiver module. The channel estimation extraction functions can be found in the Appendix: Design Codes 3, 4, and 5. These functions were modelled based on the mathematical theory detailed earlier in this subsection while incorporating adjustments for integration into the unique structure of the project. Testing of offline channel estimation, detailed in Section 5.2, has shown that the constellations of the received signals converge at the expected locations. This indicated that using the offline channel estimation technique was suitable, resolving the difficulties of implementing real-time estimation.

The Alamouti OSTBC decoding sub-module is a critical component of the receiver module whose implementation proved to be nontrivial. As its functionality has a significant impact on the final design, it was decided to test the component in a simulation environment as well as within a separate SDR module before integration into the final system. The simulation results are presented in Section 5.2.

4.4 Synchronization

The carrier frequency and symbol synchronizers presented various issues when attempting to integrate them into a MIMO system. Initially, they were placed immediately after the AGC and raised cosine filter. However, it was discovered that these components are unable to correctly compensate for any frequency offsets and timing errors when the incoming signals are OSTBC encoded. To resolve this problem, it was decided to first decode the signals using the offline channel estimation parameters before feeding them through the synchronization components.

The design approach to the implementation of a MIMO system on the SDR is to rely on the offline channel estimation parameters to decode the incoming signals. The next component in the receiver chain is the preamble detector. Although the preamble detector was not used to distinguish the pilots from the data in each frame, it serves as a vital component in conjunction with the frame synchronizer as they organize the decoded signals into their respective frames, in the same structure as how the data was transmitted. The preamble detector and frame synchronizer are located just before the QPSK/BPSK demodulator because the threshold setting in the preamble detector is only available when the input is a symbol.

Once the incoming signals have been correctly organized into their respective frames, the data is demodulated using the QPSK/BPSK demodulator and the output is fed to the bitstream to image converter to reconstruct the image based on the received bits.

4.5 Preamble Detection

Much like channel estimation, preamble detection was another implementation and testing challenge. The preamble detector identifies the unique preamble located at the start of each data frame. In terms of this project's unique communication standard, the preamble detector must detect a 11-bit Barker sequence. The preamble detector sub-module takes the incoming bitstream and correlates it with the known Barker code. Since Barker codes are unique sequences that provide high autocorrelation, the detector identifies the locations in the bitstream where the autocorrelation peaks. This design's convention places Barker code in the preamble of each frame (see Subsection 3.2), so by locating the codes, the detector effectively identifies the start of each frame.

The autocorrelations of Barker sequences 11 and 4 are shown below (Figures 4.3 and 4.4). The extra 0 bit appended to the end of the code was proved through testing to have no effect. Therefore, along with the probability advantage that longer codes provide (see Subsection 3.2), they also result in higher autocorrelation values when compared to smaller codes.

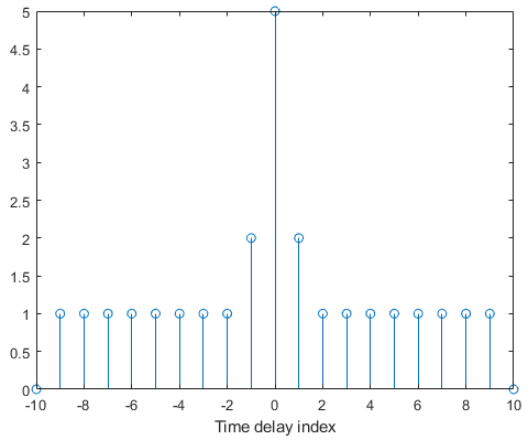


Figure 4.3: Barker Code 11 Autocorrelation

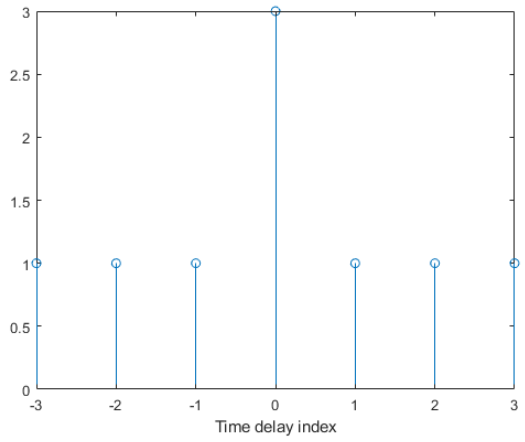


Figure 4.4: Barker Code 4 Autocorrelation

As shown in Figure 4.1, the preamble detector sub-module is followed by the frame synchronizer. The index of the location of the preamble is one of the preamble detector's outputs. This value acts as a control line that enables the frame synchronizer so that a given frame does not continue through the signal processing chain if it is not identified to be a data frame. The detection threshold of the preamble must be manually set. The only way to determine this value is through experimentation, and unfortunately, the detector was found to be extremely sensitive to noise and interference. As such, it was difficult to precisely find the threshold that would consistently locate the preambles in the expected locations. This was one of the main issues that inhibited successful hardware implementation of the MIMO system.

4.6 Bitstream to Image Conversion

After the preamble detector identifies and distinguishes each data frame, the information is exported from the Simulink environment to a Matlab script to be assembled, much like the initialization script. This code also runs automatically after each Simulink run. The design of this module was dependent on the design of the JPEG to bitstream converter detailed in Subsection 3.2 of the transmitter module section. The following steps summarize the events in the script built for this module (see Appendix: Design Code 2). First, for each frame, the first twelve bits that is the preamble are removed. This is under the assumption that the preamble detector correctly detected the Barker sequences and passed through only the frames in which the code was detected. Next, every received frame is appended to form one large matrix, whose rows represent the frames. The first frame that is passed through by the frame synchronizer is known to contain the image dimensions, so this data is extracted in order to reshape the data matrix into the dimensions corresponding to the original image. Next, the data matrix is restructured into a matrix of eight columns, each row of which corresponds to an 8-bit RGB value. Once the binary rows are converted into decimal values, the original image is reassembled using the previously extracted image dimension values. The final product is displayed using Matlab's image processing toolbox.

4.7 Receiver Module Implementation

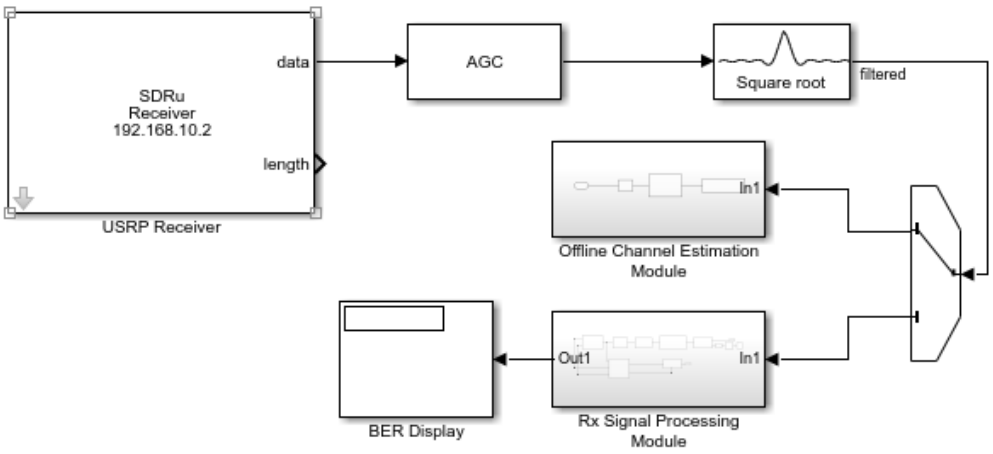


Figure 4.5: Receiver Module Simulink Design

The final Simulink receiver module shown in Figure 4.5 above contains a switch to quickly change between the offline channel estimation and image signal processing chains. The AGC simply stabilizes the received signals' amplitudes, which ensures the accuracy of the carrier and symbol synchronizers. Figures 4.6 and 4.7 below show the receiver signal processing chain and performance calculator in greater detail.

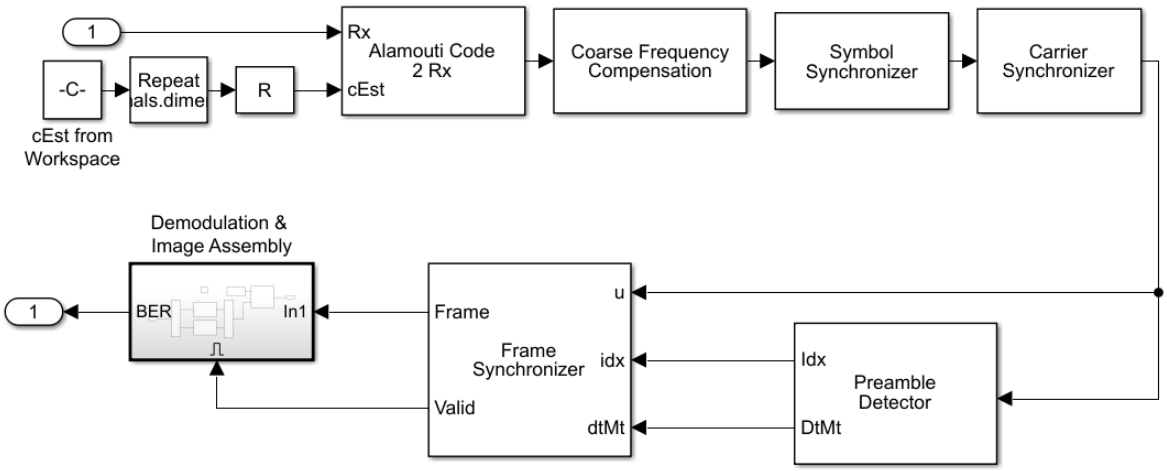


Figure 4.6: Receiver Signal Processing Simulink Design

The OSTBC decoding component is followed by carrier and symbol synchronizers, preamble detector, frame synchronizer, and finally demodulation and image assembly sub-module. Carrier and symbol synchronizers are two additionally required components for hardware implementation. This is due to the fact that the carrier frequencies of the two SDRs are not perfectly matched. Although the OSTBC decoding component equalizes the delay introduced by the channel, the symbol synchronizer acts as a secondary compensation to ensure that the received symbols are sampled at the correct instants of time. At those instants of time, the amplitudes of the symbols are at their peaks, which then allows easier decision making process for the preamble detector and the demodulator.

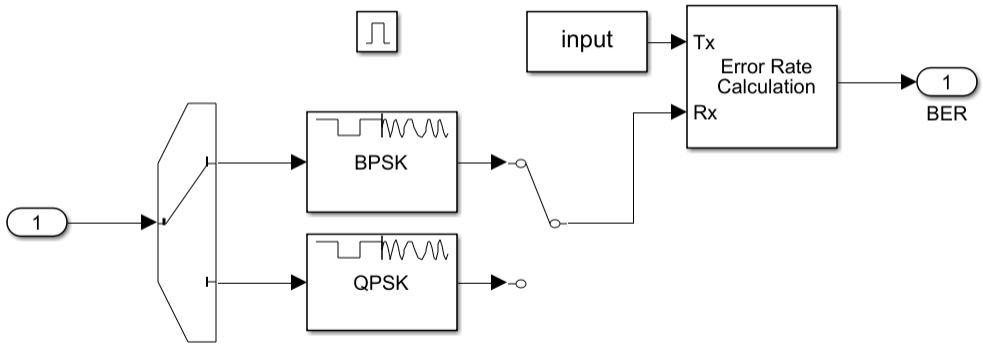


Figure 4.7: Demodulation and BER Calculation Simulink Design

Similar to the MIMO transmitter (Subsection 3.6), a manual switch was integrated to quickly change between the BPSK and QPSK demodulation schemes. The bit-rate rate is calculated using the Error Rate Calculation component by comparing the bits of the received frame with those of the input frame.

5 Simulation

This section details the simulations of the components most integral to the success of the final design. These simulations were conducted to test and verify the independent functioning of these components before integration into the overall system.

5.1 Preamble Detection Verification

The preamble detector is an important component in the receiver signal processing chain. It locates the Barker code that is appended to the beginning of each frame (see Subsection 3.2), and allows the frame to pass through if a Barker code is detected. Knowing the framing standard shared between the transmitter and receiver, the frame synchronizer ensures that the receiver complete captures each frame.

5.1.1 Preamble Detection Pattern

To properly detect the preambles in the incoming signals at the receiver, the two parameters of the preamble detector that must be adjusted are: the pattern to detect and the threshold. Both the preamble's bit sequence as well as the location of the detector relative to other components in the receiver chain dictate these parameters.

For instance, since the final design uses a Barker code of length 11 with an extra bit 0 at the end, and this preamble is to be detected before the QPSK demodulator at the receiver, the complex pattern that is to be detected is shown below in line 5.1.

$$\text{Barker Code Length 11 + '0'} = 1\,1\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0 \tag{5.1}$$

This Barker sequence is then QPSK-modulated to obtain the final preamble sequence for which the detector searches. The complex values are shown in the display of Figure 5.1. There are six values, due to the effect of QPSK modulation.

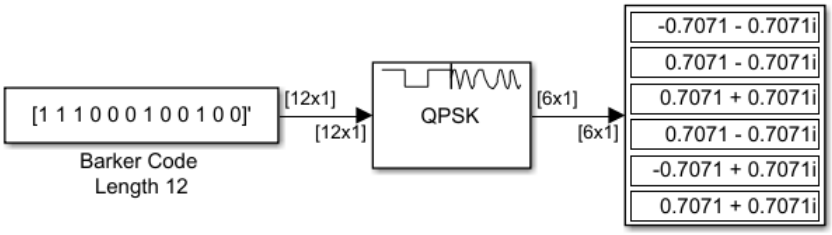


Figure 5.1: Preamble Pattern Simulink Computation

5.1.2 Preamble Detection with Small Frame Size

The output of the preamble detector is tested for consistency in Simunlink environment. The complex pattern to detect was determined as per Figure 5.1 and the detection threshold was adjusted through experimentation. In this simulation, the frame size was selected to be 100 bits long and consisted of the 12 bit-long Barker code appended to 88 randomly generated bits. This frame is repeatedly sent over a MIMO Rician and Gaussian noise channel, to simulate the slow, flat fading characteristics of a lab environment. The preamble detector is used both at the transmitter and receiver, and the indices at which the Barker code is detected from both detectors are compared. As seen in Figure 5.2, both preamble detectors were able to correctly locate the preambles at index 6, which is precisely the point at which the QPSK-modulated length 12 Barker code ends. The simulation demonstrates that the output of the preamble detector is accurate and consistent when the length of a frame is relatively short.

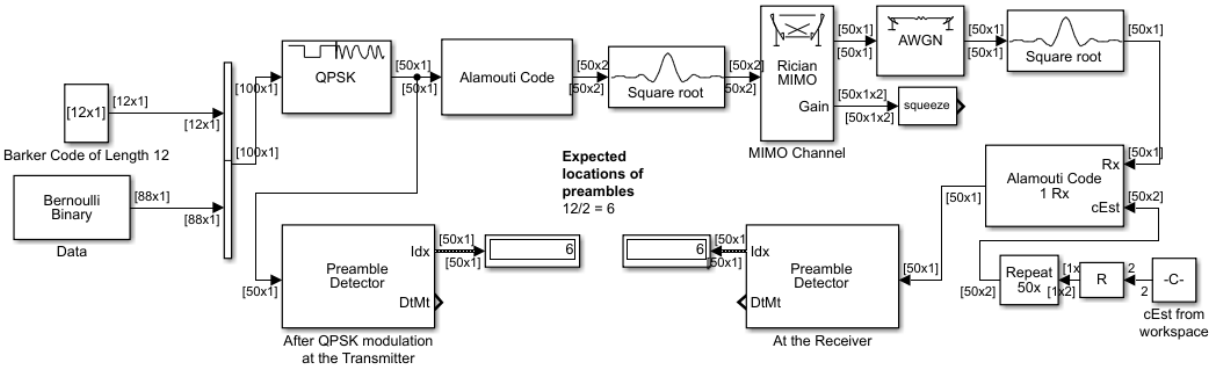


Figure 5.2: Preamble Detector Testing Schematic: Short Frame

5.1.3 Preamble Detection with Large Frame Size

The preamble detector no longer outputs indices with the same accuracy and consistency when the length of the data frame is longer than approximately one hundred thousand bits. The result of the simulation is shown in the display of Figure 5.3.

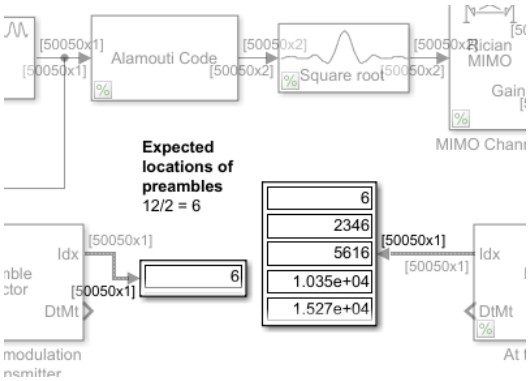


Figure 5.3: Preamble Detector Testing Schematic: Long Frame

At the transmitter, the preamble detector is able to detect the index 6, the point at which the Barker code ends in the outgoing QPSK modulated signal. However, at the receiver, the output of the preamble detector is inconsistent and outputs multiple indices. The output is heavily dependent on and is sensitive to the detection threshold. For instance, the detector's output will fluctuate when modifying up to the fourth decimal place of the threshold value. Contrarily, shorter frame sizes cause the output of the preamble detector to only be sensitive to changing a maximum of two decimal places. From the result of this simulation, it can be inferred that the probability that a sequence of symbols in the received data would randomly match the Barker code increases with the length of the data frame. As a result of the simulation, the frame size for image transmission was designed be slightly less than half of this upper limit of one hundred thousand bits to ensure a reliable preamble detector output.

5.2 Channel Estimation Verification

The designed simulation schematic in Figure 5.4 creates a frame comprised 100 pilot bits (all 1s). This frame is then modulated and transmitted through a MIMO Rician channel to simulate multi-path fading, as well as a Gaussian noise channel to simulate attenuation from interference. These fading sub-modules serve as an approximation of the final design's real-world channel environment, where there is a direct line of sight between the transmitter and receiver.

In a simulation environment, the carrier frequencies of the transmitter and receiver are perfectly matched; therefore, there is no frequency offset in the received signals and no need for a frequency synchronizer. It should also be noted that the MIMO Rician and Gaussian noise channels introduce insignificant levels of delay as opposed to that of a real-world channel, and since the transmitter and receiver modules are built in one Simulink file, the receiver can be made to sample the incoming signals at the correct time instants.

Knowing that the pilot symbols are all 1s, the decoding scheme is simplified (Table 4.1). The channel estimation parameters are determined by adding and subtracting two consecutive samples of the received signals, and applying a gain term of 1/2 (Equation 4.5). The samples of the channel estimation parameters are verified for consistency, and the averages of the samples are taken as a measure to reduce the effect of random fluctuations. The accuracy of the channel estimations parameters are then tested by switching the input signal from the pilot symbols to sequences of randomly generated bits and observing the decoded signal's constellation.

Regardless of the ideal assumptions, the primary focus of this simulation is on obtaining stable channel estimation parameters, using them to decode the incoming signals, and verifying the proper functioning of the OSTBC decoding sub-module. This ensures that a proper equalization for the channel is taken into consideration, and hence a lower bit-error rate once the decoded signals are demodulated. The code for the MISO channel estimation, the MIMO channel estimation, and the averaging functions can be found in the Appendix: Design Codes 3, 4, and 5, respectively.

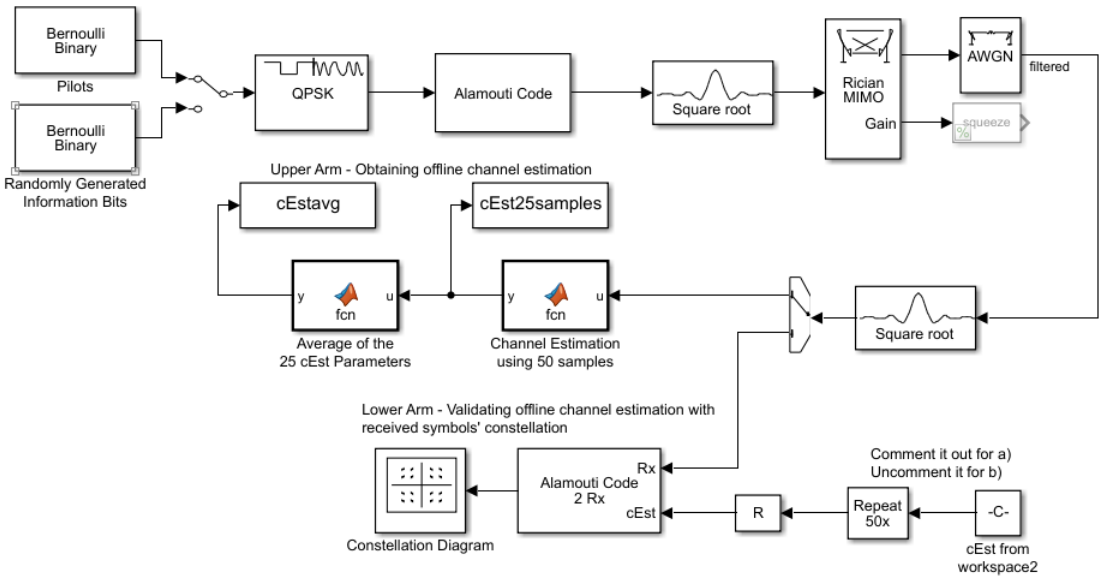


Figure 5.4: Offline Channel Estimation Schematic

The results of the simulation (Figure 5.5) indicate that the channel estimation was correctly performed. The QPSK-modulated randomly generated bits can be seen in distinct, tight clusters in the constellation map. Contrarily, the constellation of the unsuccessful channel estimation (Figure 5.6) shows how the QPSK modulated data would be poorly recovered.

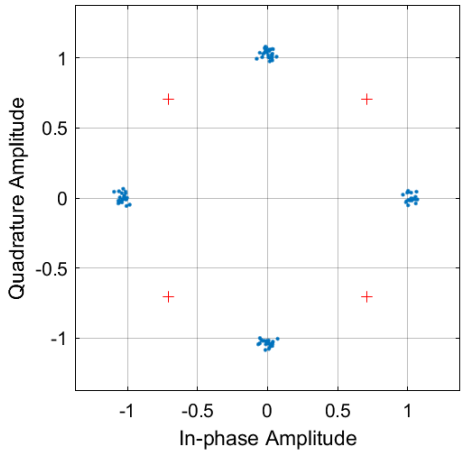


Figure 5.5: Successful Estimation

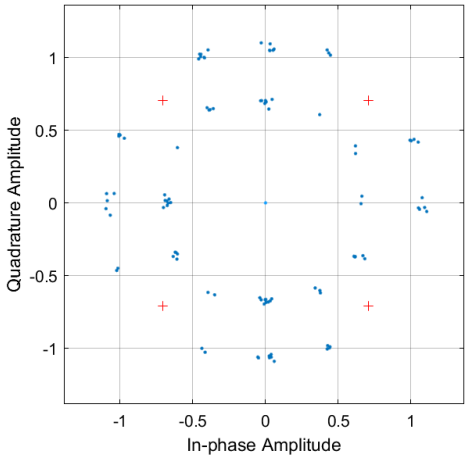


Figure 5.6: Unsuccessful Estimation

6 Hardware Results

In this project, hardware implementation refers to the creation of Simulink models that can run on the software-defined-radios. With the SISO design completed and preamble detector verified, the Simulink models for the hardware implementation of the SISO configurations were built. Through testing, it was discovered that the preamble detector posed challenges with the hardware implementation. Unfortunately, due to the project deadlines, it was decided to forego achieving complete functionality of the SISO system to advance to the MISO and MIMO systems. It was rationalized that if the difficulties with the preamble detector could not be resolved in time, the final design's implementation, testing, and analysis would be resorted to a simulated environment. The pursuit and exploration of the more complex MISO and MIMO configurations were deemed to be more valuable than perfecting a SISO system. This section details the results of testing the SISO, MISO, and MIMO hardware systems, as well as analyzing the difficulties encountered in each stage.

6.1 SISO

The SISO hardware implementation results did not align with the simulation results of Subsection 5.1. In simulation, the preamble detector was able to correctly locate the Barker codes in the received bitstream and properly extract each data frame. However, the hardware results show that the preamble detector performs poorly in a real channel, which prevents the rest of the receiver signal processing chain from functioning as intended.

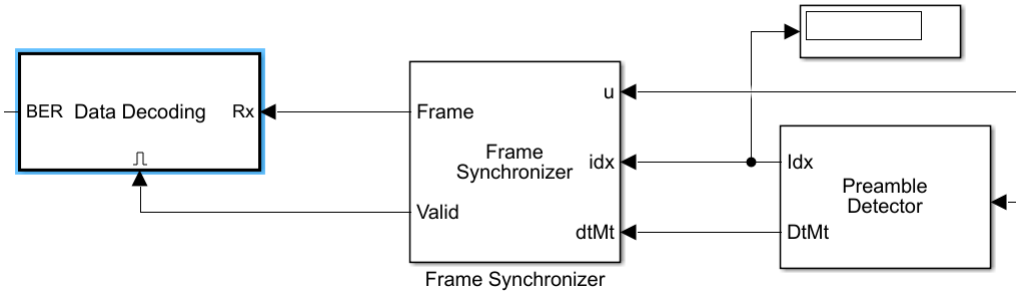


Figure 6.1: Disabled BER Data Decoder

Regardless of the detection threshold, hardware testing showed that the index output of the preamble detector would be either empty or variable. When the index output is empty, the enable signal to the BER Data Decoder sub-module is inactive. As a result, the receiver is not able to collect any frames. From Figure 6.1, it is shown that the BER Data Decoder sub-module can only be activated by a valid signal from the frame synchronizer, which only occurs when there is a stable index output from the preamble detector. Considering the other case, a variable index output implies that the received symbols are incorrectly organized into frames. In this scenario, tests showed that less than half of the transmitted frames would be received, and among the received frames, none of the frames would begin with the expected Barker code preamble. This is shown in Figure 6.2, where the first 12 bits of the received frames do not reflect the length 12 Barker code that was appended at the beginning of each frame before transmission. In this hardware system, the preamble detector did not function as intended.

It is suspected that the preamble detector's threshold setting is particularly sensitive in real-world environments. Similar to the degradation in the detector's performance when frame sizes are extremely large, the changing, non-ideal nature of a real channel is thought to create instability in the output indices. Although more tests were conducted involving the changing of numerous settings, no solution was realized.

	1	2	3	4	5	6	7	8	9	10	11	12
19	1	1	1	1	1	1	1	1	1	1	1	1
20	0	1	0	1	0	1	0	1	0	1	0	1
21	1	1	1	1	1	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1	1	1	1	1	1
23	0	1	0	1	0	1	0	1	0	1	0	1
24	0	1	0	1	0	1	0	1	0	1	0	1
25	0	1	0	1	0	1	0	1	0	1	0	1
26	0	0	1	0	0	1	1	0	0	1	0	0
27	0	1	0	1	0	1	0	1	0	1	0	1
28	1	1	1	1	1	1	1	1	1	1	1	1
29	1	1	1	1	1	1	1	1	1	1	1	1
30	0	0	1	1	0	1	1	0	0	1	1	0
31	1	1	1	1	1	1	1	1	1	1	1	1
32	0	1	1	0	1	1	0	0	1	1	0	1
33	1	1	1	1	1	1	1	1	1	1	1	1
34	0	1	0	1	0	1	0	1	0	1	0	1
35	0	1	0	1	0	1	0	1	0	1	0	1
36	1	1	1	1	1	1	1	1	1	1	1	1
37	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0
39	1	1	1	1	1	1	1	1	1	1	1	1

Figure 6.2: Received Frames

6.2 MISO/MIMO

When implementing multiple antenna configurations, channel estimation parameters are required to decode the received signals. Without these, the subsequent components in the receiver signal processing chain simply cannot function. The successful obtaining of the channel estimation parameters presented the greatest challenge in this project while having the most significant impact on the success of the final design, as it is the first step in the receiver chain.

Hardware testing of both MISO and MIMO systems revealed that the results of offline channel estimation were inconsistent between tests and between SDR devices. When using one SDR as both the transmitter and receiver, successful offline channel estimation parameters were obtained for a single test (see Figure 6.3). Analyzing the total 500 parameters that were estimated, it was found that the channel estimation values h_1 and h_2 were consistent (only 20 samples are shown in Figure 6.3). When these these offline channel estimation parameters were used to decode a randomly generated bitstream, the received signals' constellations converged to their expected QPSK-modulated coordinates. This is illustrated in Figure 6.4. Although the clusters are not as tight and distinct as those in simulation, this is an expected result of a non-ideal real-world channel.

Contrarily, when offline channel estimation was conducted using two separate SDRs as transmitter and receiver or when it is conducted again using one single SDR at a different instant of time, the obtained channel estimation parameters were no longer consistent and significantly fluctuated from one sample to another (Figure 6.5). As a result, the constellation of the received signals did not converge to their expected locations. Figure 6.6 shows that the constellation forms a diamond, indicating the presence of attenuation and phase errors.

The inability to achieve consistent channel estimation parameters is believed to be a result of an underrun error in the USRP X310 SDRs, specifically caused by buffer mismatches. Underruns are a phenomenon where the receiver samples faster than the rate at which the transmitter produces bits. This would cause the receiver to sample at time instants where data is not being transmitted. If the transmitter is sending only 1s and the receiver is oversampling, it is not guaranteed that the receiver will only receive 1s due to noise and interference in the channel. Consequently, channel estimation would fail in this case in which it is expected that the pilot symbols are constant. This is supported by the fact that channel estimation using a single SDR is also unsuccessful.

0.4155 + 0.3958i	-0.0009 - 0.0019i
0.4156 + 0.3966i	-0.0005 - 0.0010i
0.4158 + 0.3959i	-0.0004 - 0.0006i
0.4158 + 0.3962i	-0.0003 - 0.0002i
0.4155 + 0.3965i	0.0001 + 0.0002i
0.4154 + 0.3975i	0.0004 + 0.0010i
0.4154 + 0.3985i	0.0012 + 0.0017i
0.4156 + 0.3997i	0.0018 + 0.0026i
0.4156 + 0.4007i	0.0013 + 0.0035i
0.4151 + 0.4009i	0.0007 + 0.0036i
0.4151 + 0.4010i	0.0006 + 0.0039i
0.4156 + 0.4012i	0.0004 + 0.0038i
0.4153 + 0.4010i	-0.0003 + 0.0037i
0.4154 + 0.4002i	-0.0009 + 0.0031i
0.4156 + 0.3988i	-0.0011 + 0.0021i
0.4157 + 0.3977i	-0.0011 + 0.0004i
0.4163 + 0.3975i	-0.0010 - 0.0020i
0.4159 + 0.3958i	-0.0004 - 0.0029i
0.4160 + 0.3953i	0.0006 - 0.0021i
0.4162 + 0.3960i	0.0010 - 0.0015i

Figure 6.3: Successful Channel Estimation

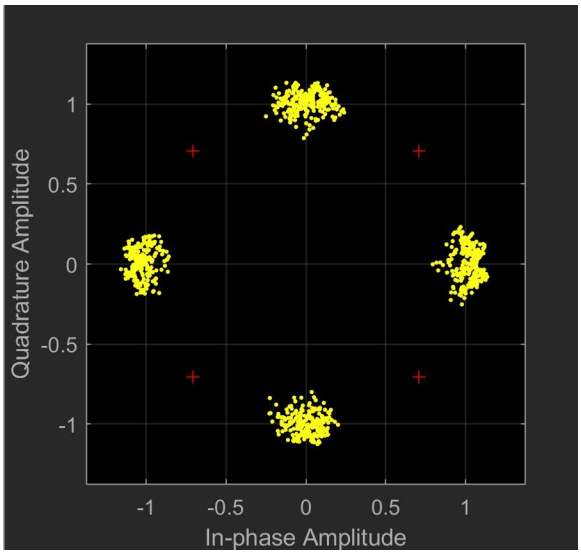


Figure 6.4: Constellation of Correctly Decoded Signals

0.0039 - 0.0002i	2.0787 - 2.1259i
0.0035 - 0.0030i	2.0638 - 2.1209i
0.0030 - 0.0029i	2.0508 - 2.1097i
0.0026 - 0.0025i	2.0395 - 2.0989i
0.0023 - 0.0022i	2.0295 - 2.0894i
0.0020 - 0.0019i	2.0208 - 2.0813i
0.0018 - 0.0016i	2.0132 - 2.0742i
0.0015 - 0.0014i	2.0066 - 2.0682i
0.0013 - 0.0012i	2.0009 - 2.0630i
0.0011 - 0.0010i	1.9959 - 2.0588i
9.7730e-04 - 8.1468e-04i	1.9917 - 2.0552i
8.5006e-04 - 6.8363e-04i	1.9881 - 2.0522i
7.3598e-04 - 5.4011e-04i	1.9849 - 2.0497i
6.4009e-04 - 4.7560e-04i	1.9821 - 2.0477i
5.5857e-04 - 3.9724e-04i	1.9797 - 2.0460i
4.8968e-04 - 2.8716e-04i	1.9777 - 2.0446i
4.2934e-04 - 2.4982e-04i	1.9758 - 2.0435i
3.8245e-04 - 2.1521e-04i	1.9742 - 2.0426i
3.3851e-04 - 1.4723e-04i	1.9728 - 2.0419i
3.0266e-04 - 1.1780e-04i	1.9715 - 2.0413i

Figure 6.5: Unsuccessful Channel Estimation

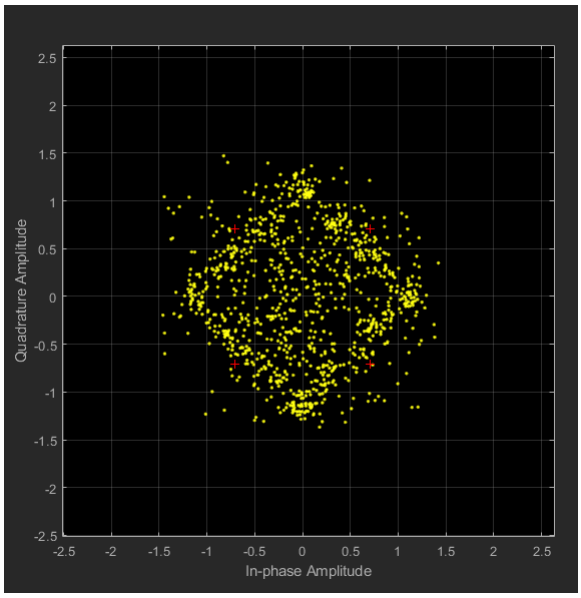


Figure 6.6: Constellation of Incorrectly Decoded Signals

In addition to the characteristics of the SDR hardware, the high memory requirements of Simulink may be a contributing factor to the channel estimation errors. Underruns may arise when the interfacing between the two hardware devices that is the computer and the SDR is imperfect, which can prevent the computer from keeping up with the sampling rate of the SDR. It was also observed that there is a mismatch in the sampling frequencies between the SDR hardware and the Simulink software. This is supported by the significant degradation in system performance that is present when the master clock frequencies of the SDRs are changed to other values supposedly supported by the X310 model. Modifying the master clock frequencies had a drastic effect on the consistency of channel estimation and subsequent sub-modules in the chain. Due to the inconsistent nature of the underrun errors and the possibility that they are inherent to the SDR devices, it was difficult to accurately identify and correct them. If it is proven to be a hardware fault, a possible solution would be to use a newer model SDR that supports buffer integrity. A solution to a hardware to hardware interfacing problem may be to conduct most of the signal processing on the SDR FPGAs to relieve the CPU workload. However, this solution is beyond the scope of this project.

Near the end of the project cycle, it was realized that solutions or workarounds to the preamble detection threshold sensitivity and the channel estimation inconsistency would not be produced in time. Therefore, the decision was made to shift the focus of the project to a simulation-based approach, and conduct testing and analysis by transmitting an image through a virtual Rician additive white Gaussian noise (AWGN) channel.

7 Performance Testing

Bit Error Rate

This section contains the results and analysis of the bit error performance parameter for all antenna configurations in simulation. The theoretical expectation is that the BPSK systems should perform the same as the Gray-coded QPSK systems in terms of BER [6]. Assuming coherent reception, that is there is no phase error in the received signal, the probability of bit error for BPSK is given by Equation 7.1.

$$P(e)_{BPSK} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{7.1}$$

When Gray coding is used, as is the case in the final designs of all QPSK systems in this project, a QPSK symbol error corresponds to one erroneous bit, and the probability of error between BPSK and QPSK are identical (Equations 7.2, 7.3, and 7.4).

$$P_b = \frac{1 \text{ (erroneous bit/symbol)}}{2 \text{ (bits/symbol)}} \tag{7.2}$$

$$P(e)_{QPSK} = Q\left(\sqrt{\frac{E_s}{N_0}}\right) \tag{7.3}$$

$$E_s = 2\ E_b$$

$$P(e)_{QPSK} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{7.4}$$

To analyze the BER, the energy per bit to noise power spectral density ratio $\frac{E_b}{N_0}$ was varied between 0 and 20 dB. The plots for the BPSK systems, QPSK systems, and the modulation comparison are shown in Figures 7.1, 7.2, and 7.3 respectively.

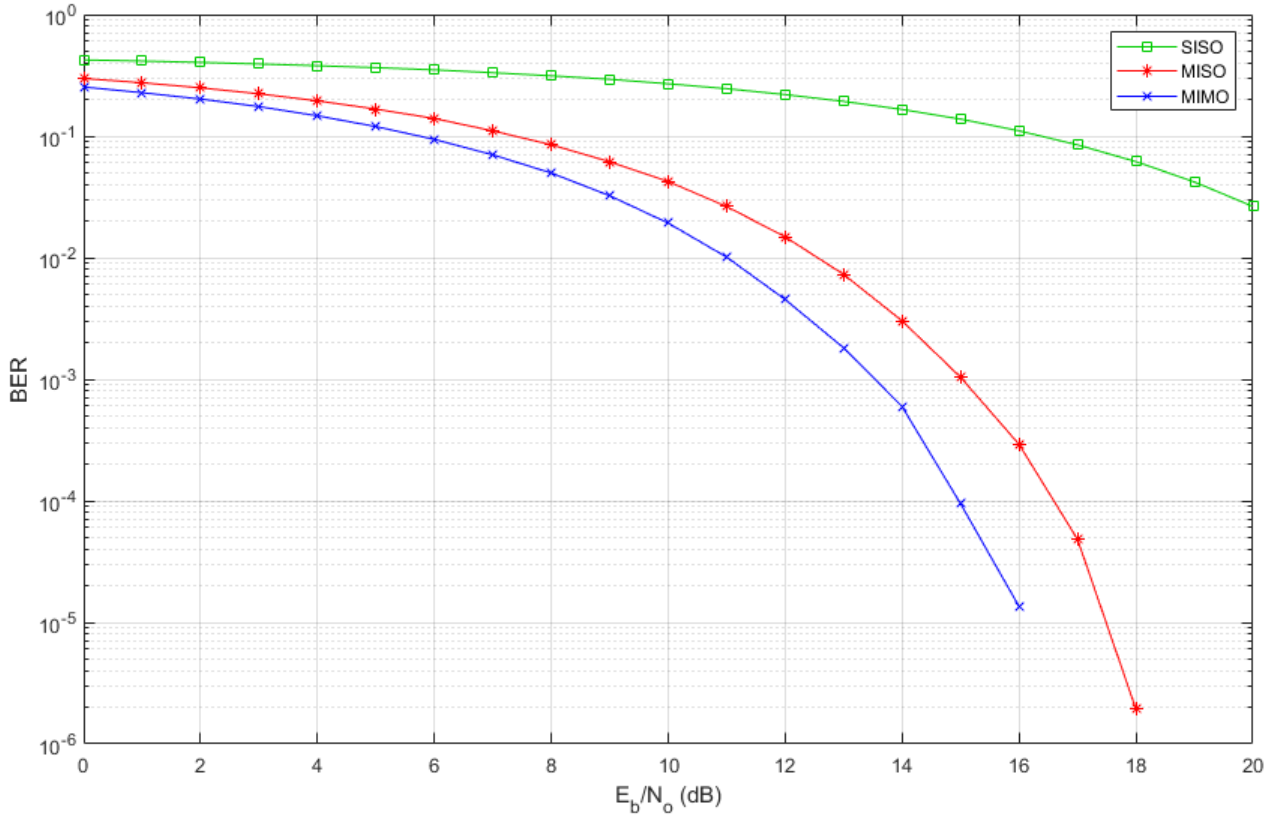


Figure 7.1: BPSK Systems BER Analysis

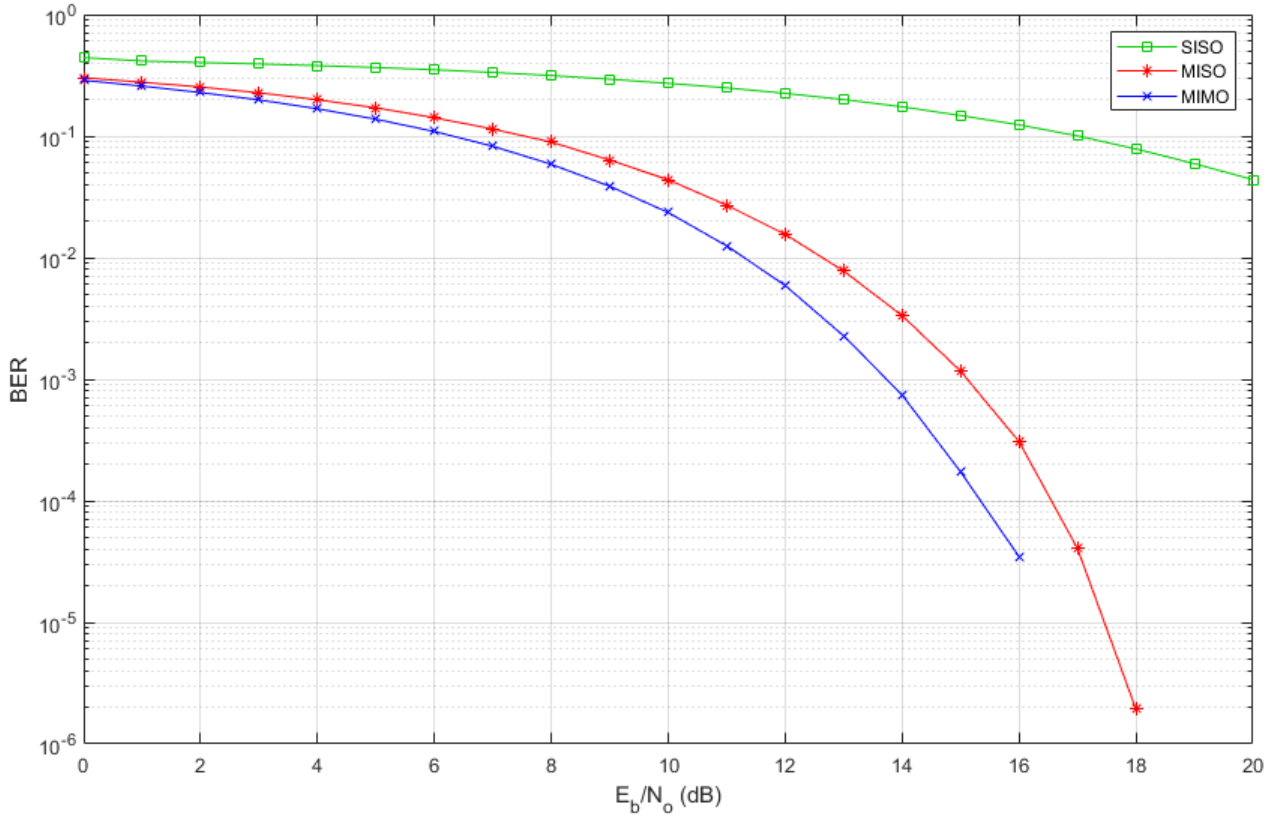


Figure 7.2: QPSK Systems BER Analysis

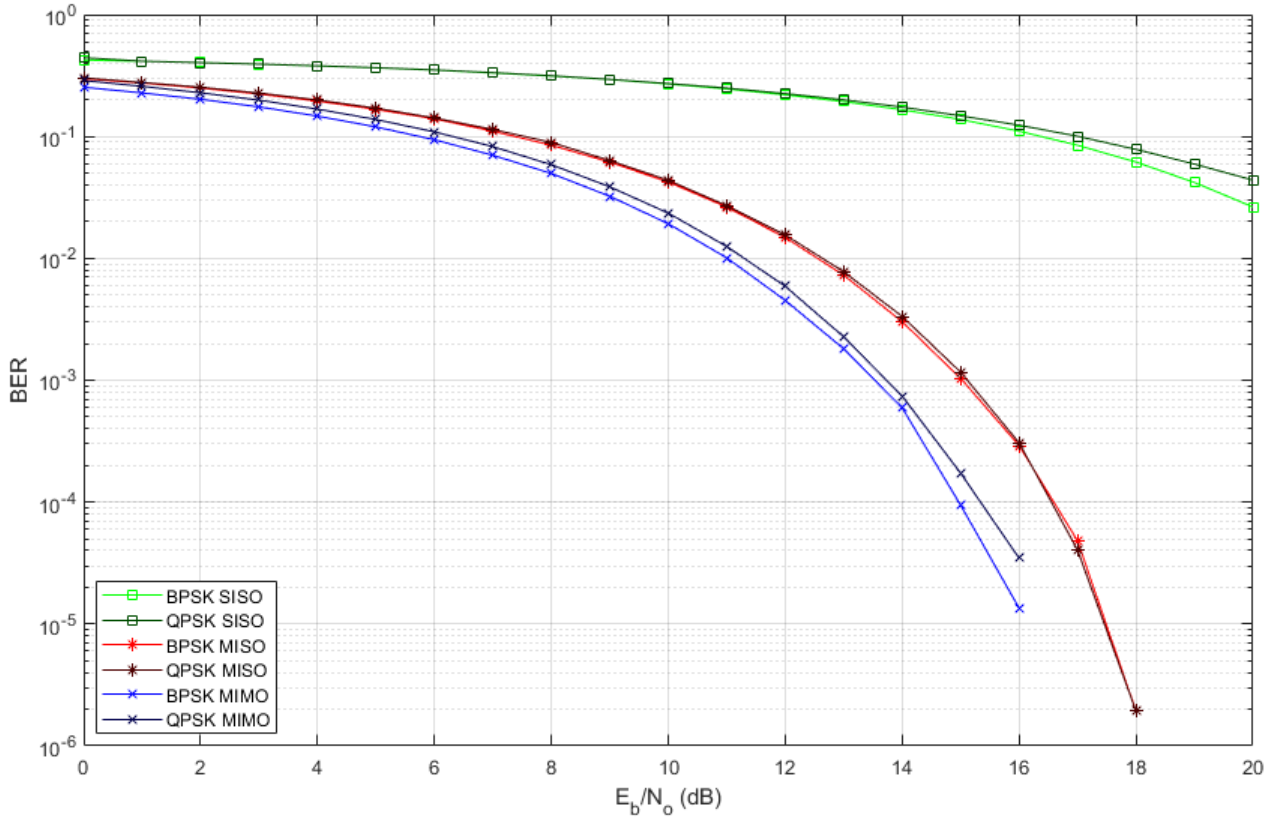


Figure 7.3: Q/BPSK BER Comparison

The BPSK and QPSK modulation schemes' BER results reflect theoretical expectations. The bit error rates for the MIMO and MISO systems stop at 16 and 18 dB respectively, because at higher E_b/N_0 ratios, the rates go to zero in the simulation. For the SISO and MIMO systems, the bit error rates of both modulation schemes are nearly identical. In the MIMO systems, there is a small discrepancy, as the BER of the QPSK system is higher, or worse, than that of the BPSK system when they should be identical. The small differences can be attributed to the relatively small number of bits that were simulated to calculate these values. This decision was made due to the unrealistic time consuming nature of running the simulations for numerous E_b/N_0 ratios with extremely high bit counts. The significance of these BER results will be analyzed in conjunction with the image transmission simulation results found in Section 8 below.

8 Simulation Results

The following simulation results show each of the SISO, MISO, and MIMO systems' outputs resulting from a 480x720 pixel input image. This input image will be referred to as "Input 1," since some systems were tested using other inputs. All systems were simulated through a Rician, additive-white-Gaussian-noise channel, with constant Doppler and energy per bit to noise power spectral density ratios.

8.1 SISO

Figure 8.1: Input 1 BPSK SISO Result

Figure 8.2: Input 1 QPSK SISO Result

From Figures 8.1 and 8.2, it is clear that the QPSK system performs better. According to theory outlined in Section 7 and the subsequent BER testing results, there should not be any difference in the two modulation schemes' performance. However, in actuality, the threshold value in the BPSK system's preamble detector could not be fine tuned accurately enough to result in a performance similar to that of the QPSK chain. The deficiency in the preamble detector was verified by observing the index values of the detector output in real time during the simulation.

8.2 MISO

Figure 8.3: Input 1 BPSK MISO Result

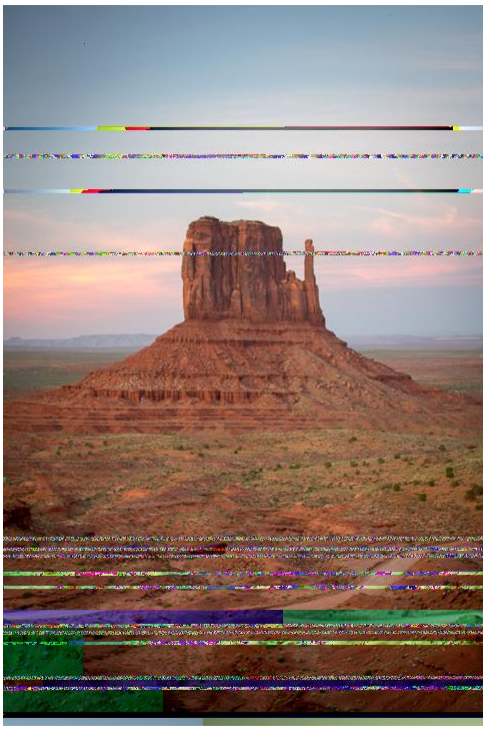


Figure 8.4: Input 1 QPSK MISO Result

From Figures 8.3 and 8.4, it is clear that the BPSK system performs better. Once again, there should not be any difference in the two modulation schemes' performance. Similar to the previous simulation, the preamble detector is failing to identify the Barker code in every transmitted frame. Close examination of the visual errors shows that the contours of the image exist, and it is simply the colours that are distorted. Since JPEG images have a 3-bit colour depth, the distortion indicates that one or more of the three frames corresponding to a single segment of the image are not received.

The BER plot of Figure 7.3 shows that the simulation performance of the BPSK and QPSK MISO systems should be identical. It can therefore be inferred that the majority of the errors seen in Figure 8.4 are the result of an inconsistent preamble detector.

The most important result from the MISO system simulation is that the performance is much improved from the SISO system in unchanging channel conditions. If the errors in the QPSK chain can be attributed to the preamble detector, it is evident from observing the BPSK result that the bit-error-rate is much lower in the MISO system. This finding illustrates the advantage of incorporating space-diverse systems.

8.3 MIMO

Figure 8.5: Input 1 BPSK MIMO Result

Figure 8.6: Input 1 QPSK MIMO Result

Figures 8.5 and 8.6 show that there is a slight discrepancy between the BPSK and QPSK systems. This is expected and can be ignored, as previously indicated by the BER plot of Figure 7.3. Since the final design is the MIMO system, more simulations were conducted to better test the robustness of the design.

Figures 8.7 and 8.8 are the outputs of a 960x1249 pixel image (Input 2), and Figures 8.9 and 8.10 are the outputs of a 1080x1620 pixel image (Input 3). None of the settings in the MIMO system's signal processing chains were altered between inputs. This demonstrates the success of the final design in terms of its adaptability to variable input sizes. Like the results of the Input 1 image, the Inputs 2 and 3 provided the same expected result, with extremely low BER values across both modulation schemes.

After simulating all three SISO, MISO, and MIMO systems and observing the relative performance improvements, another important benefit of space-diverse systems was realized. In a communication system, increasing the modulation order independently of other settings results in lower system performance in a non-ideal environment. This is because as the modulation order increases, the Euclidean distance between symbols decreases, which causes the system to be more sensitive to noise and interference. The lowest modulation order that can be implemented in communications systems is of order 2, achieved through BPSK modulation. This signifies that to achieve a low BER value in SISO systems, the energy per bit to noise power spectral density ratio must be extremely high (see Figure 7.3). In mobile systems, the transmission power cannot be indefinitely increased, which implies a performance limit that such a system can maximally achieve. However, by incorporating space diversity such as in MISO or MIMO systems, it is possible to achieve low BER values at much lower E_b/N_0 ratios. This indicates that such systems would be advantageous in mobile platforms, where transmission power is limited.

Figure 8.7: Input 2 BPSK MIMO Result

Figure 8.8: Input 2 QPSK MIMO Result

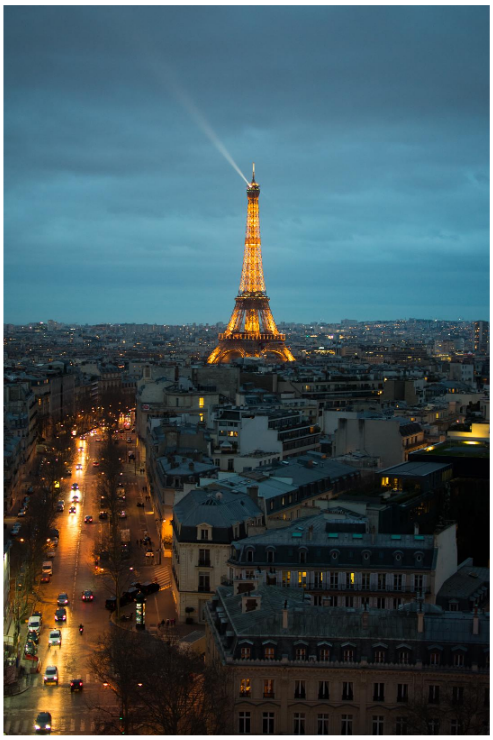


Figure 8.9: Input 3 BPSK MIMO Result

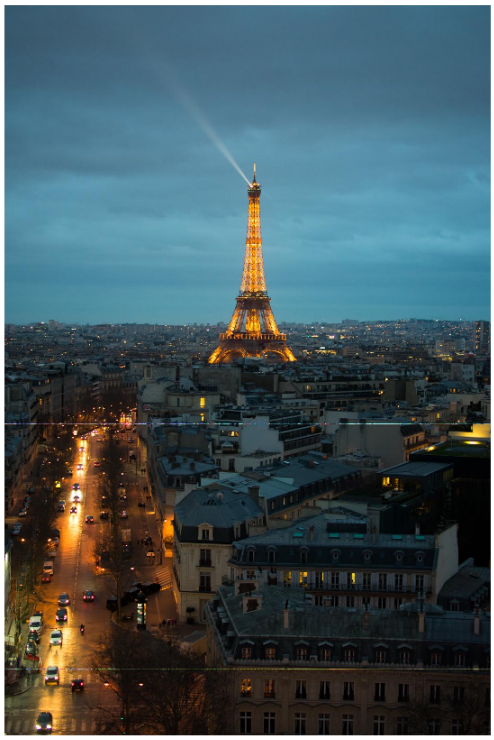


Figure 8.10: Input 3 QPSK MIMO Result

9 Discussion

This section will provide a detailed analysis of the final design's performance in relation to the requirements listed in the Statement of Requirement [1]. Since the final design was the SDR hardware implementation of a 2x2 MIMO system, the analysis of the requirements will consider only the final design.

Table 9.1 below examines all the functional results.

Index	**Requirement**	**Description**	**Result**
1	Data Generation	convert JPEG image to binary bitstream	Met
2.a	Modulation	modulate signal using BPSK	Met
2.b	Modulation	modulate signal using QPSK	Met
3	Coding	conduct Alamouti OSTBC coding	Met
4	Transmission	transmit signals on SDR	Met
5	Reception	receive signals on SDR	Met
6	Recovery	recover the transmitted signals	Unclear
7	Decoding	conduct Alamouti OSTBC decoding	Not Met
8	Demodulation	demodulate signal	Not Met
9.a	Display	display transmitted and received images	Not Met
9.b	Display	display BER	Met
9.c	Display	display spectral efficiency	Not Met
9.d	Display	display modulation scheme	Met

Table 9.1: Summary of Functional Results

Functional requirements 1 to 4 were successfully met. The transmitter signal processing chain in its entirety was successfully implemented on the USRP X310 SDR. After going through the channel, the signal was able to be picked up by the receiving SDR. It was realized during this analysis that requirement 6 was poorly defined in the SOR. It does not represent a distinct step in the signal processing chain and therefore has no significance. It will be ignored in this analysis. Requirement 7 is not met in the final design. OSTBC decoding could not be conducted, as the channel estimation parameters could not be retrieved (see Subsection 6.2). The failure to perform this step had a cascading result, leading to the inability to meet requirements 8 and 9a. Requirements 9b and 9d were still met, as the BER display and modulation scheme toggle are independent of the OSTBC decoding. Requirement 9c was not met, as it was not possible to incorporate a spectral efficiency calculation sub-module in time. Although these requirements were not met in the final hardware system, all factors with the exception of requirement 9c were successfully met in the final simulation system, which serves as a replacement to the hardware design.

Table 9.2 below examines the performance, interface, and implementation results.

Index	Requirement	Description	Result
PR	Performance Requirements		
PR-1	Bit Error Rate	transfer a 480x480 pixel image with BER 1E-4	Unclear
PR-2	Spectral Efficiency	achieve a spectral efficiency of 5.0 bits/s/Hz	Unclear
IR	Interface Requirements		
IR-1	Connections	Connect SDRs via Ethernet	Met
IR-2	Software	Use MATLAB and Simulink USRP Communications Toolbox to interface SDRs	Met
ImpR	Implementation Requirements		
ImpR-1	SDR platform	Use USRP SDRs for the hardware implementation platform	Met

Table 9.2: Summary of Performance, Interface, and Implementation Results

Through this analysis, it was realized that the originally defined bit error rate performance requirement was too vague to analyze. Both practicum and theory were learned throughout the project cycle, which resulted in a much stronger understanding of the functioning of a general communications system. However, when PR-1 was conceived, the requirement did not take into account the energy per bit to spectral noise density ratio. The requirement of achieving a certain BER value is arbitrary without specifying the E_b/N_0 ratio, so the result of this analysis is unclear.

As previously mentioned the spectral efficiency calculator module could not be created in time for the analysis, so it is unknown whether the numerical specification was met. All the interface and implementation requirements were satisfied.

10 Conclusion

The purpose of this project was to design and implement a MIMO communications system using software defined radios to wireless transmit an image. The objectives were to transmit an image, receive the image, and assess the system's performance. The incremental project model was used to develop the final design. The design and development, testing, and implementation phases were conducted sequentially from the least complex SISO system to the more complex MISO and MIMO systems. Before hardware implementation, the most important components of the signal processing chains were verified in simulation. However, during the final testing stage of the MIMO design, several of the key components in the receiver signal processing chain presented implementation challenges.

One of these components was the offline channel estimator. Unlike in simulation, obtaining channel estimation parameters was inconsistent and heavily dependent on the SDR. Due to the underrun errors present in the USRP X310 SDRs, any calculated channel estimation parameters were unusable in decoding. As a result, the rest of the receiver signal processing chain in the context of the overall design could not be tested on hardware.

The other component that presented implementation challenges was the preamble detector. Although the capabilities and limitations of this component were rigorously studied and tested in simulation before integrating it into the final hardware design, the preamble detector could not be made to consistently locate the Barker codes in each frame. As a result, the detector could not be integrated in the receiver signal processing chain.

The unexpected behaviour of the offline channel estimation and preamble detector led to the decision to implement the final design and assess its performance in simulation. Without any changes in the overall design, Simulink models were used to transmit, receive, and successfully recover images. The only difference between the hardware and simulation implementation was the substitution of the real-world channel by a Rician, AWGN channel model. It is suspected that because of the absence of the hardware synchronization error and the use of a virtual channel, the channel estimation and preamble detection were fully functional.

The implementation of the final design in simulation validated and supported the theory and engineering decisions that were applied to the project. The specific data packaging standard that was designed was proven to be responsive enough to transmit images of varying sizes. Additionally, the design decision to integrate offline channel estimation as a replacement to real-time channel estimation was proven to be a viable solution for this project. This approach in obtaining the channel parameters was found to be a simple but accurate method of integrating Alamouti's OSTBC scheme. With the successfully creation of simulation models, it was possible to test and validate the theory behind space-diversity, as well as modulation theory, since two different modulation schemes were integrated. The resultant bit-error-rate plots and sets of recovered images corresponded to both the theoretical expectations and the characteristics unique to the designed system.

Aside from the eventual inability to implement the final design in hardware, the simulation-based product successfully met all requirements, with the exception of calculating the spectral efficiency parameter. In the end, the transmitter and receiver signal processing chains were proved functional and the system displays the received image.

Unfortunately, the limitations of the preamble detector and the hardware synchronization errors are unresolved. The preamble detector threshold's extreme sensitivity is unclear, even when using a Barker code with high autocorrelation. Perhaps either an adaptive threshold or a more robust system architecture would be required to address this issue. Further testing of the USRP X310 SDRs is required to more comprehensively understand the synchronization issues, which are present even when the motherboard clock frequency and the transmission carrier frequencies are identical. If these two issues can be resolved, it is believed that successful hardware implementation on the SDRs is possible.

Through this project, various subsets of communications theory were explored and learned in detail. By designing and implementing the final system on hardware and in simulation, the difficulties of implementing a project for real-world use were realized. Based on the successful simulation results, the advantages of using a multi-antenna configuration along with space-time coding were validated both visually with the images and numerically with the bit-error-rate plots. In the future, this project can be expanded to incorporate other space-time coding schemes and modulation techniques to accommodate more complex antenna configurations. This is useful for the application of massive MIMO, truly adaptive modulation scheme systems, and real-time channel estimation systems, which are necessary for variable channel environments.

11 Appendix

Matlab Code

Design Code 1: jpeg2bitstream.m

```
%% JPEG to bitstream converter code

% This script converts a user inputted JPEG image into a bitstream
% according to the designed data structure standard.

% Insert JPEG file here:
jpg_img = double(imread('test_small','jpg'));

% Memory initialization
rx_bitstream_qpsk = [];
rx_bitstream_bpsk = [];

if exist('rx_img.jpg', 'file')==2
  delete('rx_img.jpg');
end

% Preamble initialization
my_barker_code = [1 1 1 0 0 0 1 0 0 1 0 0];
bc_len = length(my_barker_code);

img_rows = size(jpg_img,1);
img_cols = size(jpg_img,2);
img_depth = size(jpg_img,3);

% 12 bits for image dimension data: this means maximum image dimension is
% 2^12 = 4096 pixels
bin_rows = de2bi(img_rows,12); %binary
bin_cols = de2bi(img_cols,12); %binary
bin_depth = de2bi(img_depth,12); %binary

% Image manipulation
decimalstream = zeros(img_rows,img_cols*img_depth);
decimalstream = reshape(jpg_img,[img_rows,img_cols*img_depth]);
dec_col = reshape(decimalstream',[],1); % each row of the image after one another
bit8_col = de2bi(dec_col);

% Pad data so it is divisible by 40k
N = 40e3; % number of data bits per frame
R = rem(size(bit8_col,1)*size(bit8_col,2),N);
fr_pad_size = (N-R)/8;
fr_pad = zeros(fr_pad_size,8);
padded_mtrx = vertcat(bit8_col,fr_pad);
bit1_col = reshape(padded_mtrx', [], 1);

bit40k_mtrx = reshape(bit1_col,N,[])';

% Add dimension data row
dim_data = horzcat(bin_rows,bin_cols,bin_depth);
dim_row = horzcat(dim_data,zeros(1,N-length(dim_data)));
data_mtrx = vertcat(dim_row,bit40k_mtrx);

% Add Barker codes to each frame
bc_pad = repelem(my_barker_code, size(data_mtrx,1), [1]);
tx_mtrx = horzcat(bc_pad, data_mtrx);

% Create structure for Simulink access
input = struct;
input.signals = struct;
input.time = [];
input.signals.values = tx_mtrx;
input.signals.dimensions = size(tx_mtrx,2);
```

Design Code 2: bitstream2jpeg.m

```
%% Image Reconstruction Script

% This script takes the decoded and demodulated frames and reassembles the
% received image.

my_barker_code = [1 1 1 0 0 0 1 0 0 1 0 0];
bc_len = length(my_barker_code);

if ~isempty(rx_bitstream_qpsk)
    rx_mtrx = squeeze(rx_bitstream_qpsk)';
elseif ~isempty(rx_bitstream_bpsk)
    rx_mtrx = squeeze(rx_bitstream_bpsk)';
end

rx_mtrx = rx_mtrx(:,bc_len+1:end); % remove each frame's Barker code

% Extract dimension data row
dim_data = rx_mtrx(1,:);
img_rows = bi2de(dim_data(1:bc_len));
img_cols = bi2de(dim_data(bc_len+1:2*bc_len));
img_depth = bi2de(dim_data(2*bc_len+1:3*bc_len));

% Remove dimension data row
rx_mtrx(1,:) = [];

% Cut off extra/repeated data
N = size(rx_mtrx,2);
rx_mtrx = rx_mtrx(1:ceil(img_rows*img_cols*img_depth*8/N),:);

% Binary to decimal conversion
rx1_col = reshape(rx_mtrx',[],1); % making a column vector of data
rx8_col = reshape(rx1_col,8,[]);
rxde_col = bi2de(rx8_col');

% Remove zero padding
R = rem(img_rows*img_cols*img_depth*8,N);
fr_pad_size = (N-R)/8;

data_col = rxde_col(1:end-fr_pad_size);

% Create image according to extracted dimensions
pre_img_mtrx = reshape(data_col, img_cols*img_depth, img_rows)';
img_mtrx = reshape(pre_img_mtrx, img_rows, img_cols, img_depth);

img_mtrx = uint8(img_mtrx);
imwrite(img_mtrx,'rx_img.jpg','jpg');
imshow(imread('rx_img.jpg','jpg'));
```

Design Code 3: MISO_chEst.m

```
%% MISO Channel Estimation Calculation

% This function takes the pilots symbols after going through the channel
% and calculates the channel estimation parameters for a MISO system.

function y = fcn(u)

    u1 = complex(zeros (25,1));
    u2 = complex(zeros (25,1));
    j = 1;
    for i = 1:25
        u1(i) = u(j);
        u2(i) = u(j+1);
        j = j + 2;
    end

    u_resized = [u1' ; u2'];

    h1=u_resized(1,:)+u_resized(2,:);
    h2=-u_resized(1,:)+u_resized(2,:);
    k1=.5;
    k2=.5;

    y=[ -k2*h2.' k1*h1.'];
end
```

Design Code 4: MIMO_chEst.m

```
%% MIMO Channel Estimation Calculation

% This function takes the pilots symbols after going through the channel
% and calculates the channel estimation parameters for a MIMO system.

function y = fcn(u)
    u1 = u(:,1);
    u2 = u(:,2);

    u1 = reshape(u1,[2,25]);
    u2 = reshape(u2,[2,25]);

    v = vertcat(u1,u2);

    h1 = v(1,:)+v(2,:);
    h2 = -v(1,:)+v(2,:);
    k = .5;

    h3 = v(3,:)+v(4,:);
    h4 = -v(3,:)+v(4,:);

    y=[-k*h4.' k*h3.' -k*h2.' k*h1.'];
end
```

Design Code 5: chEst_avg.m

```
%% This function takes the average of the multiple channel estimation parameter samples

function y = fcn(u)
   y = mean (u,1);
end
```

References

[1] T. Jung and E. J. Kim, "Did-03 statement of requirement," Oct. 2019.

[2] ——, "Did-04 preliminary design specification," Nov. 2019.

[3] E. Research, "Usrp x310 high performance software defined radio." [Online]. Available: https://www.ettus.com/all-products/X310-KIT/

[4] S. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, p. 1451–1458, 1998.

[5] B. P. B. P. Lathi and Z. Ding, *Modern Digital and Analog Communication Systems*. Oxford Univ Press, 2010.

[6] S. S. Haykin and M. Moher, *Communication systems*. Wiley, 2010.