ECE4305: Software-Defined Radio Systems and Analysis

Laboratory 2: Basic SDR Implementation
of a Transmitter and a Receiver

C-Term 2011

Objective

This laboratory will provide a theoretical foundation for various modulation schemes and their robustness to error. You will also be introduced to Simulink as a development tool for communications systems. This laboratory will implement a "bare bones" communication system in Simulink. This laboratory assumes a knowledge of MATLAB but little or no knowledge of Simulink.

Contents

1 Theoretical Preparation

This section will provide an understanding of several basic digital modulation schemes that form the basis of modern communications. You will learn to analyze the power efficiency of various schemes and analyze mathematically the error robustness of these modulation schemes.

1.1 Digital Modulation Schemes

In analog modulation schemes, the message signal modulates a continuous wave. Conversely, digital modulation involves having the message signal modulated against a pattern of bits and transforming them into symbols. As we will see, this can be accomplished by uniquely manipulating the amplitude and phase information of a signal during each symbol period T.

1.1.1 Pulse Amplitude Modulation

Pulse amplitude modulation (PAM) is a form of signal modulation where the message information is encoded in the amplitude of a series of signal pulses. Demodulation is performed by detecting the amplitude level of the carrier at every symbol period.

Pulse amplitude modulation (PAM) constructs symbols with varying amplitudes. The most basic form of PAM is a series of rectangular pulses with varying amplitudes.

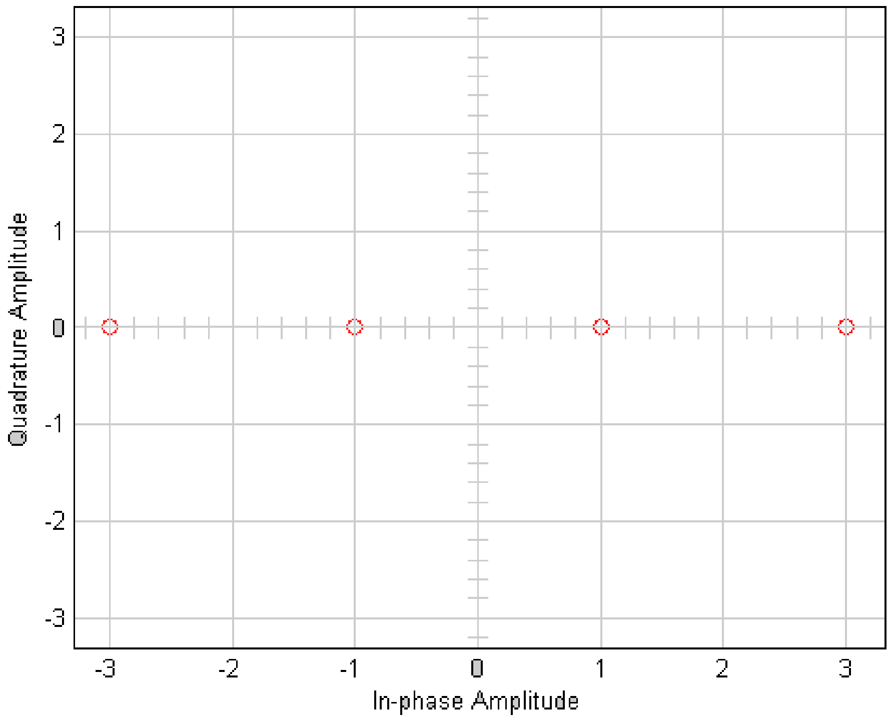


Figure 1: A two bit modulator (PAM-4) will take two bits at a time and will map the signal amplitude to one of four possible levels, for example 3 volts, 1 volt, 1 volt, and 3 volts.

For more information about pulse amplitude modulation, please refer to Section 5.2 of the course textbook [6].

1.1.2 Quadrature Amplitude Modulation

Similar to PAM, quadrature amplitude modulation (QAM) implies some sort of amplitude modulation. However, QAM modulation is a two-dimensional scheme as apposed to PAM modulation. For instance, recall the discussion of complex baseband and how 4-QAM is represented:

$$S_i(t) = I_i(t)\cos(2\pi f_c t) + Q_i(t)\sin(2\pi f_c t), \tag{1}$$

where $I_i(t)$ is in-phase amplitude, $Q_i(t)$ is quadrature amplitude, and f_c is carrier frequency.

The two dimensions of the QAM modulation, namely the in-phase and quadrature components, are orthogonal to each other, which implies you can essentially double your data rate "for free." Rectangular QAM can be thought of as two orthogonal PAM signals being transmitted simultaneously. Mathematically, this may be represented as $\sqrt{M}$PAM + $\sqrt{M}$PAM = M-QAM. QAM constellations could also take the form of nested circles (called circular QAM), or any other geometric pattern that involves orthogonal modulators. Which geometric pattern you choose is the result of how easy the signal constellation is to transmit and how robust the constellation is to noise.

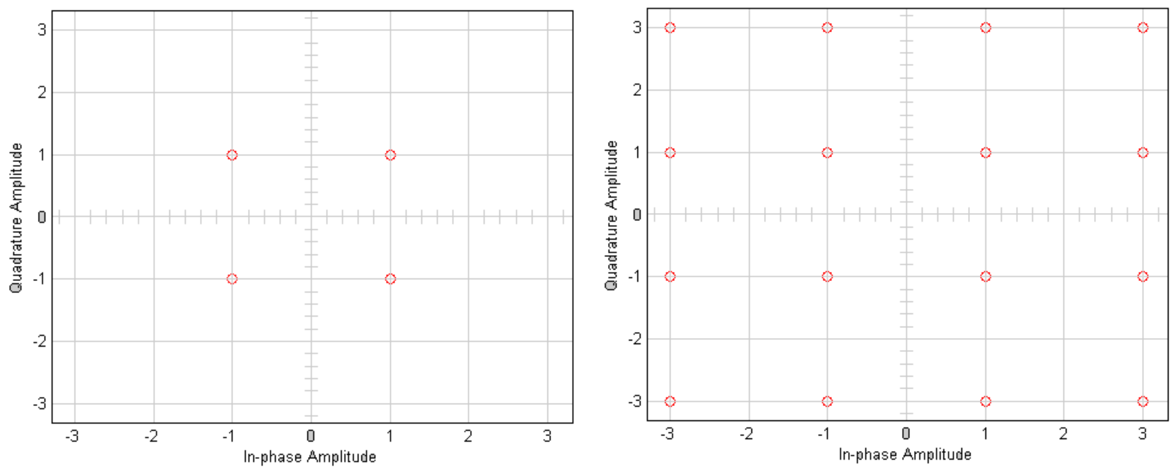


Figure 2: 4-QAM and 16-QAM captured on a scope. Since QAM is usually square, the most common forms are 16-QAM, 64-QAM, 128-QAM and 256-QAM. By moving to a higher-order constellation, it is possible to transmit more bits per symbol.

For more information about quadrature amplitude modulation, please refer to Section 5.3 of the course textbook [6].

1.1.3 Phase Shift Keying

Phase shift keying (PSK) is a digital modulation scheme that conveys data by changing, or modulating, the phase of a reference signal (the carrier wave).

Any digital modulation scheme uses a finite number of distinct signals to represent digital data. PSK uses a finite number of phases, each assigned a unique pattern of binary digits. Usually, each phase encodes an equal number of bits. Each pattern of bits forms the symbol that is represented by the particular phase. The demodulator, which is designed specifically for the symbol-set used by the modulator, determines the phase of the received signal and maps it back to the symbol it represents, thus recovering the original data. This requires the receiver to be able to compare the phase of the received signal to a reference signal such a system is termed coherent.

Phase shift keying (PSK) characterizes symbols by their phase. Mathematically it is represented by:

$$S_i(t) = A\cos(2\pi f_c t + (2i-1)\frac{\pi}{m}), \quad \text{for} \quad i = 1, ..., \log_2 \mathrm{m}, \tag{2}$$

where A is the amplitude, f_c is carrier frequency, and $(2i-1)\frac{\pi}{m}$ is the phase offset of each symbol.

PSK presents an interesting set of trade-offs with PAM and QAM. In amplitude modulation schemes, channel equalization is an important part of decoding the correct symbols. In PSK schemes, the phase of the received signal is much more important than the amplitude information.

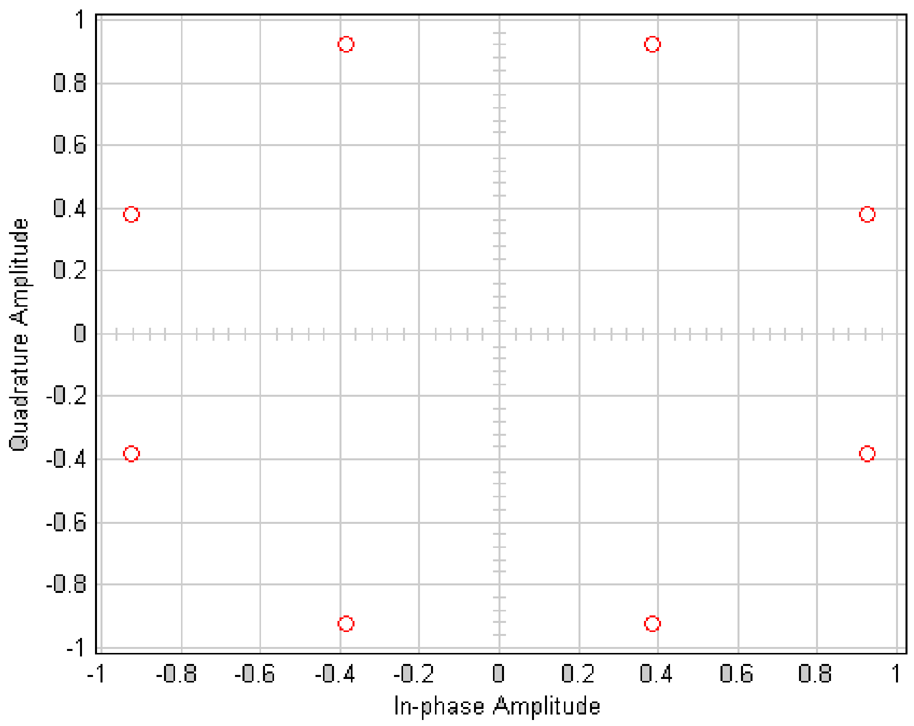


Figure 3: 8-PSK captured on a scope. In PSK, the constellation points chosen are usually positioned with uniform angular spacing around a circle. This gives maximum phase-separation between adjacent points and thus the best immunity to corruption. They are positioned on a circle so that they can all be transmitted with the same energy.

For more information about phase shift keying, please refer to Section 5.4 of the course textbook [6].

1.2 Power Efficiency

When analyzing a constellation, one must consider its power efficiency, defined by:

$$\epsilon_p = \frac{d_{min}^2}{\bar{E}_b}, \tag{3}$$

where d_{min} is the minimum Euclidean distance between two points that occurs in a signal constellation. $\bar{E}_b$ refers to the average energy per bit.

Example: Suppose we have the signals:

$$S_1(t) = A\cos(w_c t + \theta) \quad \text{and} \quad S_0(t) = 0.$$

Finding the energy of each signal, we see that:

$$E_1 = \frac{A^2T}{2}, E_0 = 0$$

$$d_{min}^2 = E_1 = \frac{A^2T}{2}$$

Assuming that each signal is equiprobable:

$$E_b = E_{S1} \times p(1) + E_{S0} \times p(0) = E_{S1}\frac{1}{2} = \frac{A^2T}{4}$$

$$\epsilon_p = \frac{d_{min}^2}{\bar{E}_b} = 2$$

1.3 Probability of Bit Error

One of the most common metrics for measuring the performance of a communications system is the probability that a bit transmitted will be decoded erroronously (or the probability of bit error, BER). The probability of error, $P(e)$, is expressed as a sum of pairwise error probabilities, or, the probability of one received symbol being another, specific received symbol. The pairwise error probability of A being decoded when B was transmitted is given as:

$$Q\left(\frac{d_{ab}^2}{2N_0}\right), \tag{4}$$

where N_0 is the variance of the noise. An important note here is that we are assuming the noise is AWGN, since Q functions apply specifically to Gaussians. So the complete $P(e)$ is expressed as:

$$Q\left(\frac{d_{min}^2}{2N_0}\right) \leq P(e) \leq Q\left(\frac{d_{I1}^2}{2N_0}\right) + ... + Q\left(\frac{d_{IM-1}^2}{2N_0}\right), \tag{5}$$

where the second half of the relationship is the summation of every single pairwise error probability.

1.3.1 Error Bounding

Computing each pairwise error probability is not always practical. It is possible to create an upper and lower bound on $P(e)$ by computing only the pairwise errors of points that are within one degree of the point of interest. Consider the behavior of the $Q(X)$. As X increases, the solution approaches zero. You will find that computing the pairwise error probability of points farther away yields negligible contributions to the total $P(e)$, but can save a significant amount of time as well as cycles. Thus, an accurate estimate of $P(e)$ can be computed from the following bounds, as shown in Figure 4.

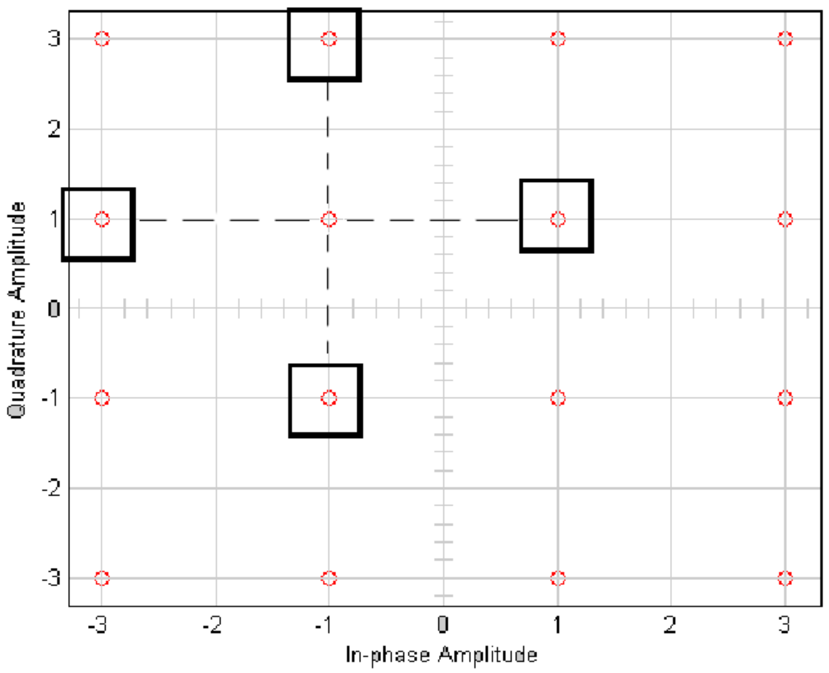


Figure 4: Error bound demonstrated, where there are four points denoted B, C, D, E to calculate the pairwise error probability in Eq. 6.

These upper and lower bounds can be expressed as:

$$Q\left(\frac{d_{min}^2}{2N_0}\right) \leq P\left(e\right) \leq Q\left(\frac{d_{AB}^2}{2N_0}\right) + ... + Q\left(\frac{d_{AE}^2}{2N_0}\right). \quad (6)$$

In order to accurately assess the performance of a communications system, it must be simulated until a certain number of symbol errors are confirmed [8]. In most cases, 100 errors will give a 95% confidence interval. In this laboratory, we will simulate 100 errors to characterize the bit error rate of our system.

For more information about probability of error, please refer to Section 6.1 and 6.2 of textbook [6].

1.4 Suggested Readings

Although this laboratory handout provides some information about the fundamentals of digital modulation schemes, the reader is encouraged to review the material from the following references in order to gain further insight on these topics.

- Overview of modulation & demodulation
  - Chapter 5 in [6]
- Overview of performance of different modulation schemes
  - Chapter 6 in [6]
- Introduction to simulation techniques for baseband modulation and demodulation
  - Chapter 2 in [7]

1.5 Problems

1. Find and compare the power efficiency for the three binary signal sets below. Give the relative performance in decibels. Assume the $s_1(t)$ and $s_2(t)$ are equally likely.

   (a) $s_1(t) = Bsin(\omega_0 t + \phi)$ and $s_2(t) = Bsin(\omega_0 t - \phi)$ for $0 \leq t \leq T$ and where $\bar{E}_b \leq \frac{A^2 T}{2}$. Find the best (B, ϕ).

   (b) $s_1(t) = Asin(\omega_0 t + \theta)$ and $s_2(t) = Bsin(\omega_0 t)$ for $0 \leq t \leq T$ and where $\bar{E}_b \leq \frac{A_0^2 T}{2}$ and A_0 is known. Find the best (A, B, θ).

   (c) $s_1(t) = Asin(2\pi f_0 t + \Delta t)$ and $s_2(t) = Acos(2\pi f_0 t - \Delta t - \theta)$ for $0 \leq t \leq T$. Find the best peak frequency deviation, Δ, in terms of T.

2. Three different eight-point constellations are proposed as shown in Figure 5. Draw the appropriate decision boundaries for each technique in two dimensions, and express the probability of symbol error, P_S in terms of the *peak* energy-to-noise density ratio. Repeat for an *average* energy normalization. Which of the demodulators would be easier to implement? Justify.

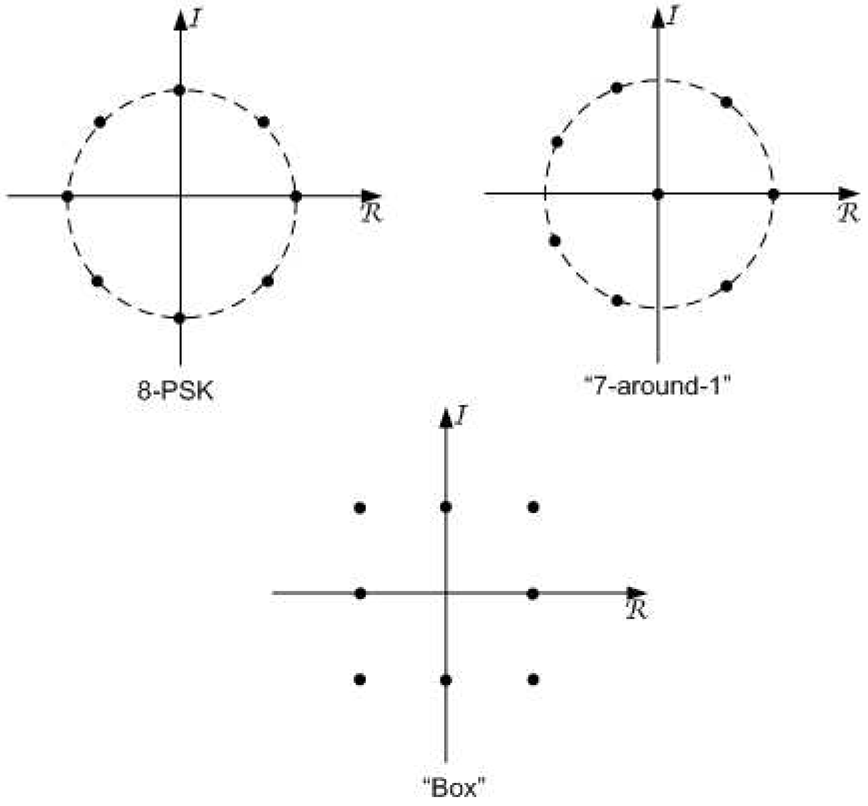


Figure 5: Three 8-point signal constellations (8-PSK, "7-around-1", and "Box").

3. The signal constellation for a communications system with 16 equiprobable symbols is shown in Figure 6. The channel is AWGN with noise power spectral density of $\frac{N_0}{2}$. Compute and compare the power efficiency of this system with a 16-level PAM system with adjacent amplitude differences of $2A$.

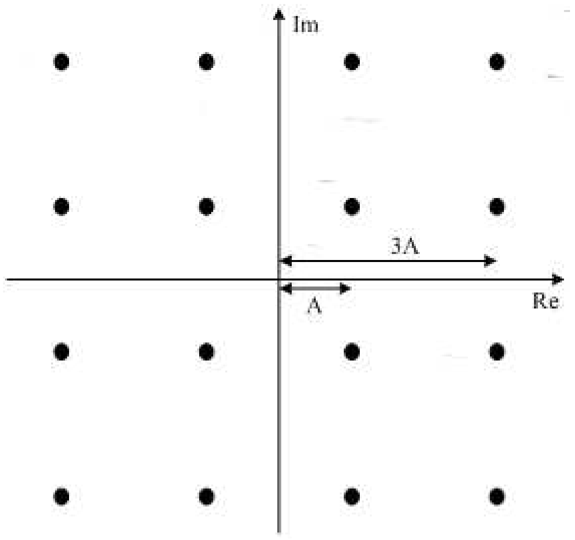


Figure 6: 16 QAM signal constellation.

4. Exercise 6.4 from course textbook [6].

2 Software Implementation

Open the `txbasic.mdl` file available on the course website.

This model demonstrates a very basic transmitter. There is no channel, and therefore no impairments. Random “1s” and “0s” are generated then transmitted inside 36 bit long frames. The frames pass through some encoding to make them more robust to noise and are then modulated to 16 QAM.

While you can see all of the blocks of this system, it is not complete. The user defined functions are empty. It is up to you to complete them. Before you begin writing code, you should examine the assigned parameters of the defined blocks in the `Mask Parameters` and attempt to understand why they are set as they are. When working in Simulink, the format of your data is extremely important. If you send a [36x1] signal to a port expecting a [37x1] signal not only will your model not run, it will often throw far more than 1 error, unnecessarily complicating the issue. Understanding how each block affects your data format will greatly simplify your work in Simulink.

2.1 Repetition Coding

Forward error correction (FEC) is a method of adding redundant data to the transmitted stream to make it more robust to channel error. There are many types of FEC. You will implement a simple repetition coder (repetition factor $R = 4$). This means that if a 0 symbol is to be transmitted, 0000 is the actually codeword transmitted. Double click on the repetition coder block to write your MATLAB code. Remember, you can set break points in your functions.

- What are the trade-offs to consider when choosing between a high or a low repetition factor?

2.2 Interleaving

A repetition code is a good start towards a robust transmission, but it is not enough. If your transmitted data is 101101, employing a repetition factor of 4 yields:

$$111100001111111100001111.$$

While this looks robust to error, what happens if during your transmission a noise spike yields a received binary steam:

$$111100 - - - - - - - - -1100001111,$$

which is corrupted?

Some of the original data is completely irretrievable. A simple interleaver will mix up the repeated bits to make the redundancy in the data even more robust to error. Running 111100001111111100001111 through an interleaver might yield something like:

$$101101101101101101101101.$$

Different mixing algorithms will change the effectiveness of the interleaver.

When you are done, combine the repetition coder and the interleaver into a single FEC sub-system.

2.3 BER Calculator

Bit error rate (BER) is one of the primary metrics to consider when measuring the performance of a communications system. BER is calculated by dividing the total number of errors by the total number of bits transmitted. The alignment block is accounting for the delay of the signal as it propagates through your other blocks.

As the model is configured, your BER should be zero. Check this by connecting both the `input` ports of the alignment block to the initial bit sequence.

- Note the two `persistent` variables - what does persistent mean?
- Once you believe you have achieved success, intentionally invert one of the bits. Does your BER calculated behave as expected?
- What effect might different signal constellations have on the bit error rate?

2.4 Receiver Implementation over an Ideal Channel

Now you will implement the receiver for the transmitter you just built. Since this is an ideal channel, there is no noise and therefore no impairments to account for. The receiver design is simply the inverse of your transmitter design, as shown in Figure 7. Feed the decoded signal into your BER calculator and you should get a BER of zero.

- When configuring your user functions, remember that formatting is important! You may need to set the expected port sizes in the model explorer.
- By default, the output of a `user-defined function` is sample based. Certain blocks you implement expect frame based inputs. You can use the `To frame` block to rectify this. A double-lined connection indicates the signal is frame based; a single line connection indicates the signal is sample based.

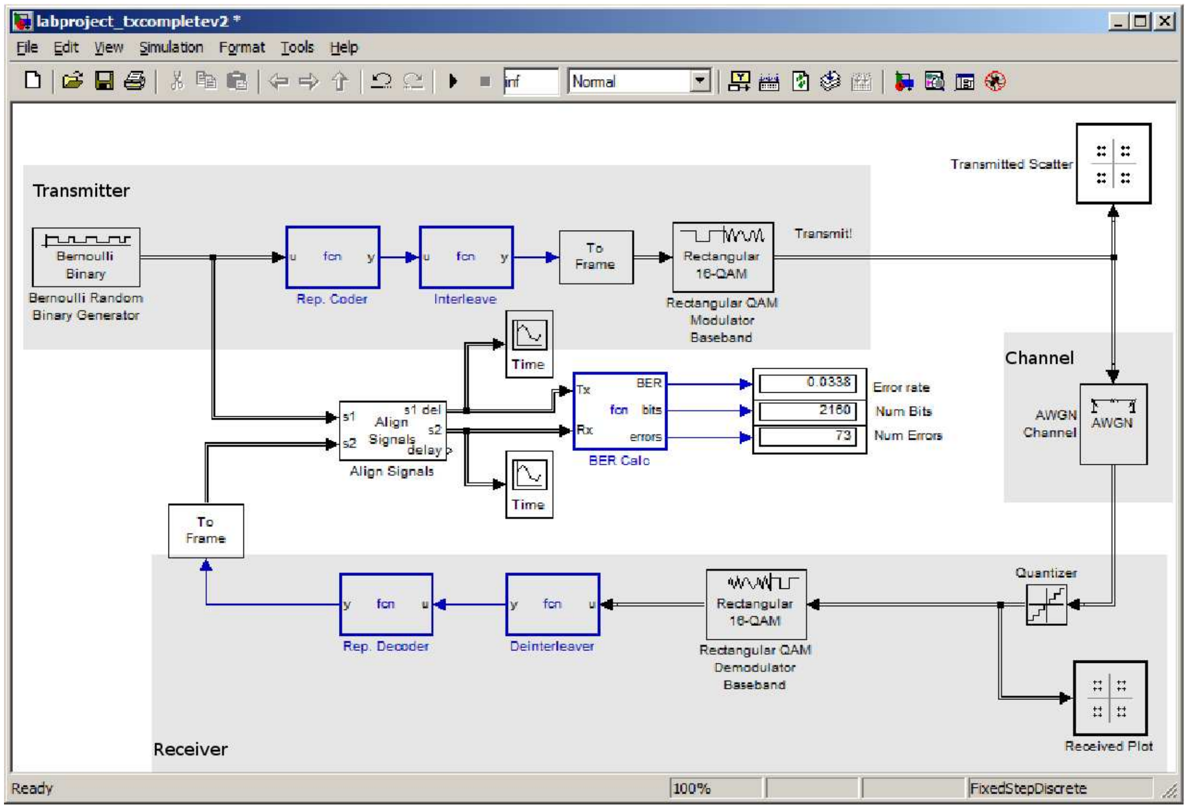


Figure 7: Approximate appearance of final Simulink transceiver model. This model implements a digital communication system using 16-QAM and repetition coding. You can simulate the communication channel by adding the `AWGN Channel` block.

3 USRP2 Hardware Implementation

In Laboratory 1, you have already observed a digital communication system employing differential binary phase-shift keying (DBPSK). In this lab, we will revisit this system, and then you are required to construct a similar DQPSK system.

3.1 Differential Binary Phase-Shift Keying

Differential phase shift keying is a non-coherent form of phase shift keying which avoids the need for a coherent reference signal at the receiver. Non-coherent receivers are relatively easy and cheap to build, and hence are widely used in wireless communications.

3.1.1 Transmitter

The transmitter is very straightforward, which is made up of four blocks, as shown in Figure 8.

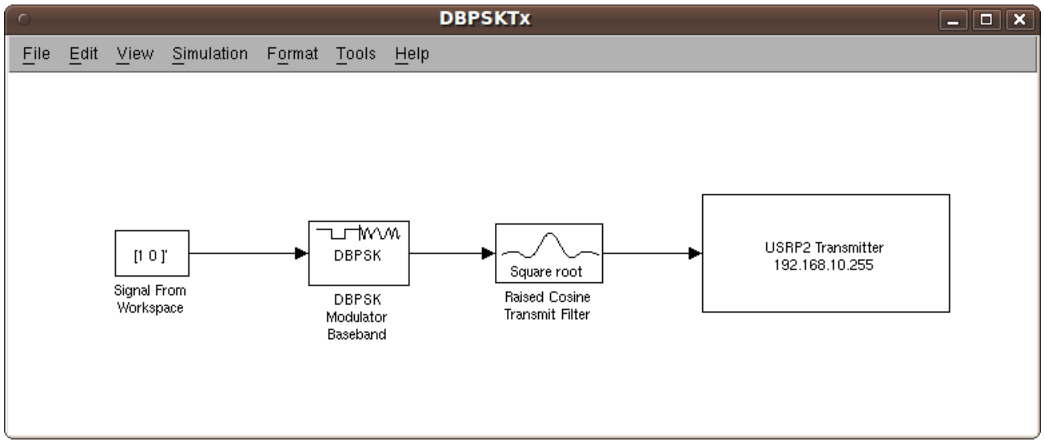

DBPSKTx
File Edit View Simulation Format Tools Help
[1 0]'
Signal From Workspace
DBPSK
DBPSK Modulator Baseband
Square root
Raised Cosine Transmit Filter
USRP2 Transmitter 192.168.10.255

Figure 8: The structure of DBPSK transmitter, where a binary source is modulated using DBPSK and filtered using raised cosine pulse shaping filter.

The purpose and special parameters of each block is as follows:

- `Signal From Workspace`:
  - `Signal`: We use a valid MATLAB expression ([1 0]') to specify the signal, which generates a repeated '10'. We use this signal since it is easy to observe.
  - `Sample time`: In this parameter, there is a variable named `BitRate`. This variable is specified by callback functions, which is introduced in Section 3.1.3.
  - `Samples per frame`: We set this number to be 179, because the input frame size of `USRP2 Transmitter` block needs to be 358, and `Raised Cosine Transmit Filter` has an up-sampling factor of 2.
- `DBPSK Modulator Baseband`: In DBPSK system, the input binary sequence is differentially modulated using a DBPSK modulator. In Simulink, this operation is conducted by `DBPSK Modulator Baseband` block. The output of this block is a baseband representation of the modulated signal.

- `Raised Cosine Transmit Filter`: This block upsamples and filters the input signal using a square root raised cosine FIR filter. The block's icon shows the filter's impulse response. Similar to the `Signal From Workspace` block, the variable named `oversampling` is defined in callback functions.

- `USRP2 Transmitter`: This block has been introduced in detail in Laboratory Tutorial. The only thing you need to keep in mind is that the best input frame size to this block is 358.

3.1.2 Receiver

The receiver behaves in an inverse manner relative to the transmitter, but there are more things going on, so it is more complicated than the transmitter. The receiver model is made up of three parts, as shown in Figure 9.

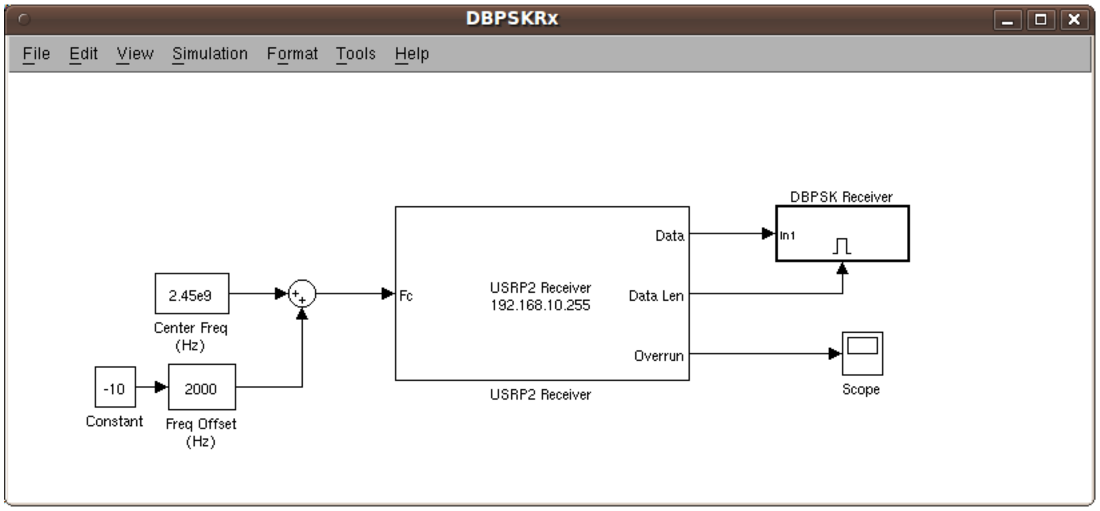


Figure 9: The structure of DBPSK receiver, where the transmitted signal is received by `USRP2 Receiver` block and fed into the `DBPSK Receiver` subsystem. Frequency offset between two USRP2s should be compensated to ensure signal reception.

The first part is the frequency offset compensator. You should use the frequency offset found in previous laboratory in this part. The second part is the `USRP2 Receiver` block, which has been introduced in Laboratory Tutorial. Finally, the third part is an enabled subsystem, where all the main operations of this model reside, as shown in Figure 10.

The purpose and special parameters of each block is as follows:

- `Frame Conversion`: Since the output of the `USRP2 Receiver` block is sample-based, we use this block to convert it to frame-based. This block does not make any changes to the input signal other than the sampling mode.

- `Raised Cosine Receive Filter`: This block filters the input signal using a square root raised cosine FIR filter. The Raised Cosine Receive Filter block's icon shows the filter's impulse response.

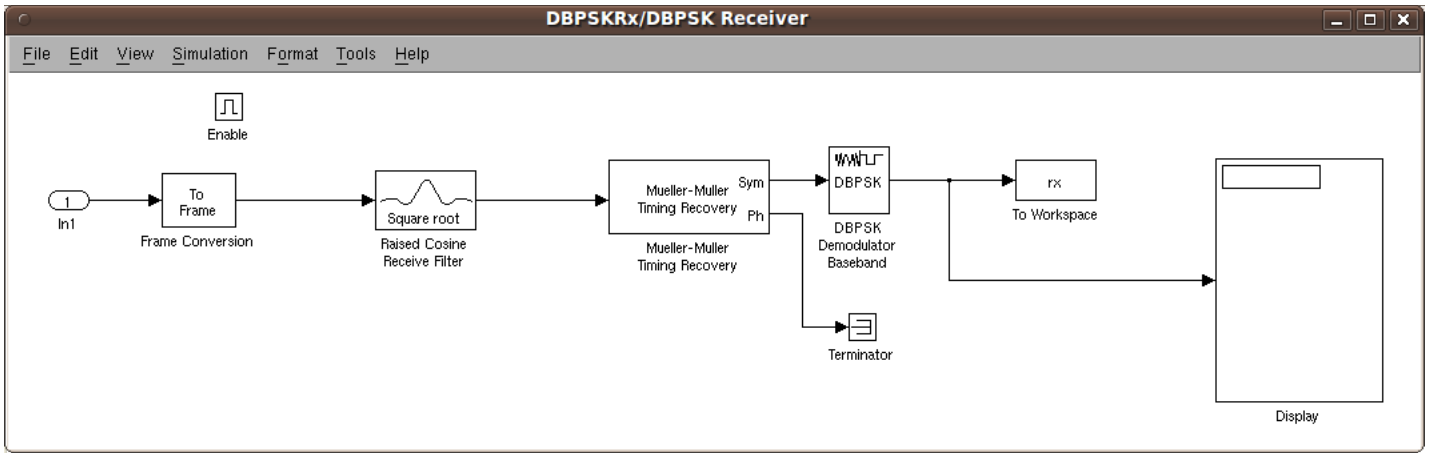

DBPSKRx/DBPSK Receiver
File Edit View Simulation Format Tools Help
Enable
1
In1
To Frame
Frame Conversion
Square root
Raised Cosine Receive Filter
Mueller-Muller Timing Recovery
Sym
Ph
Mueller-Muller Timing Recovery
DBPSK
DBPSK Demodulator Baseband
rx
To Workspace
Terminator
Display

Figure 10: The structure of `DBPSK Receiver` subsystem, where a symbol is filtered using raised cosine pulse shaping filter and demodulated using DBPSK. The timing recovery is conducted by `Mueller-Muller Timing Recovery` block.

- `Mueller-Muller Timing Recovery`: One of the most significant advantages of a DPSK modulation scheme is that you do not need to worry about the *carrier recovery*, which estimates and compensates for frequency and phase differences between a received signal's carrier wave and the receiver's local oscillator for the purpose of coherent demodulation. However, you still need to implement some form of *timing recovery*. The purpose of the timing recovery is to obtain symbol synchronization. Two quantities must be determined by the receiver to achieve symbol synchronization. The first is the sampling frequency, and the other is sampling phase. The `Mueller-Muller Timing Recovery` block recovers the symbol timing phase of the input signal using the Mueller-Muller method. This block implements a decision-directed, data-aided feedback method that requires prior recovery of the carrier phase.

- `DBPSK Demodulator Baseband`: This block demodulates a signal that was modulated using the differential binary phase shift keying method. The input is a baseband representation of the modulated signal.

- `To Workspace` and `Display`: These two blocks serve as the sink of this model. Using these two blocks, you can observe the received data in two ways. You can either observe a portion of data directly through `Display`, or save a larger number of data to workspace through `To Workspace`.

3.1.3 Using Callback Functions

About Callback Functions

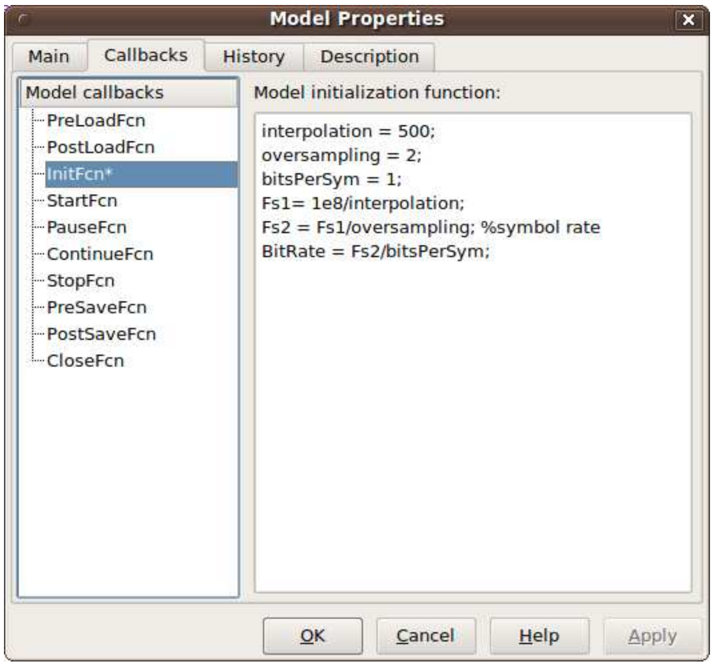
Callback functions are a powerful way of customizing your Simulink model. A *callback* is a function that executes when you perform various actions on your model, such as clicking on a block or starting a simulation. You can use callbacks to execute a MATLAB script or other MATLAB commands. You can use block, port, or model parameters to specify callback functions.

Common tasks you can achieve by using callback functions include:

- Loading variables into the MATLAB workspace automatically when you open your Simulink model. [1]
- Executing a MATLAB script by double-clicking on a block.
- Executing a series of commands before starting a simulation.
- Executing commands when a block diagram is closed.

Creating Model Callback Functions

You can create model callback functions interactively or programmatically. Use the **Callbacks** pane of the model's **Model Properties** dialog box to create model callbacks interactively, as shown in Figure 11. You can access it by right click on the blank space of your model. In **Callbacks** pane, the main callbacks you can define include the following: PreLoadFcn, PostLoadFcn, StartFcn, StopFcn, CloseFcn.



```
interpolation = 500;
oversampling = 2;
bitsPerSym = 1;
Fs1= 1e8/interpolation;
Fs2 = Fs1/oversampling; %symbol rate
BitRate = Fs2/bitsPerSym;
```

Figure 11: Create model callback functions using **Callbacks** pane of the model's **Model Properties** dialog box. In our DBPSK example, several parameters are defined in InitFcn.

To create a callback programmatically, use the `set_param` command to assign a MATLAB expression that implements the function to the model parameter corresponding to the callback. Read the online documentation *Using Callback Functions* for more information about creating a callback programmatically.

3.2 Differential Quadrature Phase-Shift Keying

With the theory of DBPSK presented, it is now time to construct a DQPSK system. Perform the following tasks:

[1]This is what happens in our model.

- Construct a DQPSK transmitter named `DQPSKTx.mdl`. Use the `DQPSK Modulator Baseband` block and `Raised Cosine Transmit Filter` block. Choose all the parameters appropriately for your model.
- Construct a DQPSK transmitter named `DQPSKRx.mdl`. Use the `DQPSK Demodulator Baseband` block and `Raised Cosine Receive Filter` block. Choose all the parameters appropriately for your model. Do not forget to compensate the frequency offset between two USRPs.
- Plot the data you have saved to workspace.
- Compare the performance between DBPSK and DQPSK, which one is better? Why?

3.3 USRP2 In-phase/Quadrature Representation

Understanding how a signal is represented can greatly enhance one's ability to analyze and design baseband digital communication systems. As mentioned in Laboratory 1, a bandpass signal can be represented by the sum of its in-phase (I) and quadrature (Q) components. Moreover, the data input to the USRP2 board is complex, which includes I and Q. Since I and Q play a very important role in digital communications, now you are going to observe the I and Q data on transmitter and receiver sides.

3.3.1 Observing In-phase and Quadrature Data

For the transmitter side, you can use `siggen.mdl` from Laboratory 1.

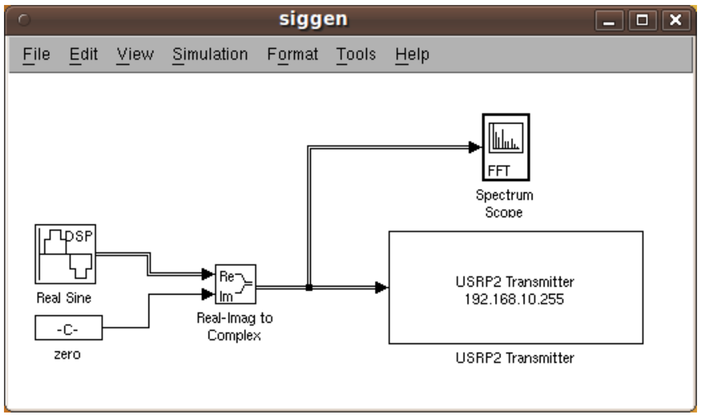


Figure 12: The structure of `siggen.mdl`. In this model, since the `USRP2 Transmitter` block requires the complex input, the `Real-Imag to Complex` block converts real and imaginary inputs to a complex-valued output signal.

For the receiver side, you can download `observeiq.mdl` from course website. The main subsystem of this model is shown in Figure 13, where two `Vector Scope` blocks are attached to the real and imaginary output.

Specify the following six sets of values in `siggen.mdl`:

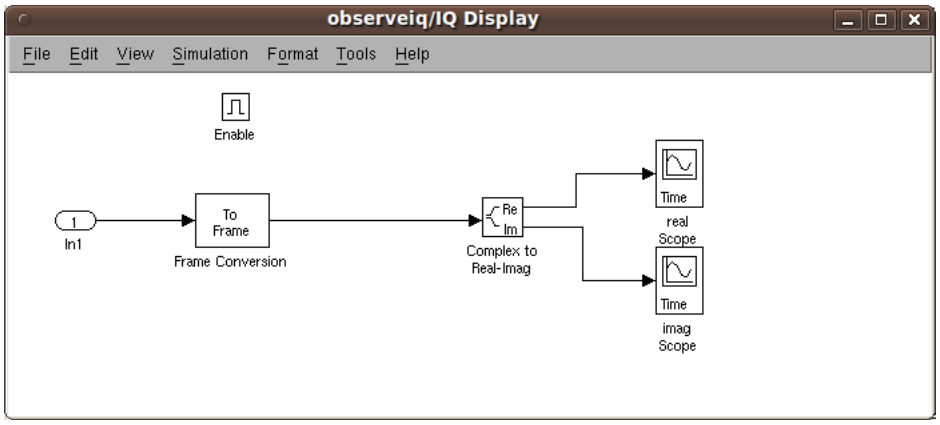


Figure 13: The structure of main subsystem of `observeiq.mdl`. Since the `USRP2 Receiver` block produces the complex output, the `Complex to Real-Imag` block converts the complex-valued signal to real and imaginary components, which you can observe using two `Vector Scope`.

Set Index	Real	Imaginary
Set 1	0	0
Set 2	0	1
Set 3	1	0
Set 4	0	sine
Set 5	sine	0
Set 6	sine	sine

Record the corresponding output in `observeiq.mdl` by saving the plots of `Vector Scope`. Is the output the same as the input you have specified?

3.3.2 Mathematical Derivation

The behavior of the in-phase and quadrature data is closely related to the structure of USRP2 board, especially the digital up converter (DUC) and digital down converter (DDC).

At the transmitter path, a baseband I/Q complex signal is sent to the USRP2 board. The digital up converter will interpolate the signal, upconvert it to the IF band, and finally send it through the DAC. The structure of DUC is shown in Figure 14, which is named as *complex multiplier*.
At the receiver path, the standard FPGA configuration includes digital down converters (DDC) implemented with 4-stage cascaded integrator-comb (CIC), as shown in Figure 15. First, the DDC down-converts the received signal from the IF band to the base band. Next, it decimates the signal so that the data rate can be adapted by the USB 2.0 and is reasonable for the computers' computing capability. The complex input signal (IF) is multiplied by the constant frequency (usually the IF) exponential signal. The resulting signal is also complex, and centered at 0. Then the signal is decimated with a factor M. The decimator can be treated as a low pass filter followed by a down sampler.
Given this insight on the structure of DUC and DDC, you can now perform the following mathematical derivations:

- In Figure 14, suppose the input I/Q complex signal is $I_{in}(t)$ and $Q_{in}(t)$, the carrier frequency is ω_c, what is the output of the DUC, denoted as $I^{'}(t)$ and $Q^{'}(t)$?

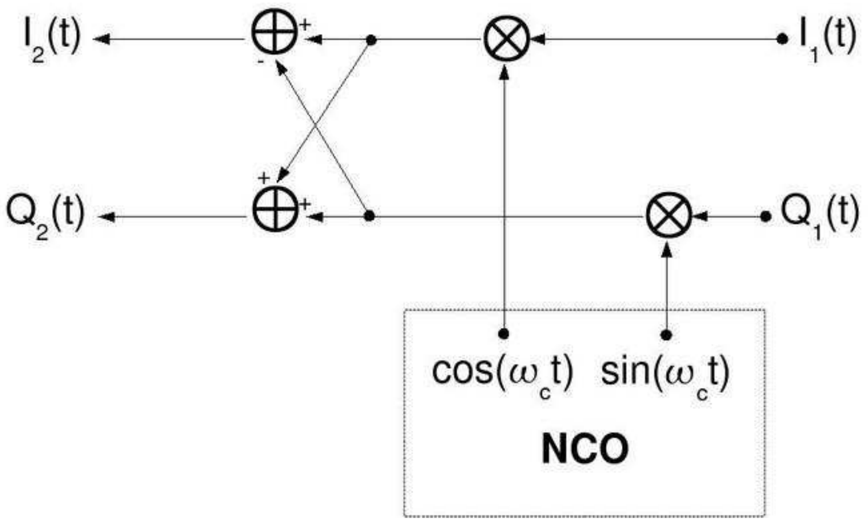


Figure 14: The structure of digital up converter on transmitter path, where a baseband I/Q complex signal $I_1(t)$ and $Q_1(t)$ is upconverted to the IF band by a carrier frequency of ω_c.

- In Figure 15, suppose the input I/Q complex signal is $I^{'}(t)$ and $Q^{'}(t)$, which has been expressed in the previous step. Also assume that the the carrier frequency has a little offset $\Delta\omega$ compared to the DUC, so it is $\omega_c + \Delta\omega$. What is the output after CORDIC and before CIC? Please express them as $I^{''}(t)$ and $Q^{''}(t)$.
- In Figure 15, for simplicity, the rest part after CORDIC, including CIC and 1/2 Band, can be viewed as a low pass filter. $I^{''}(t)$ and $Q^{''}(t)$ will go through the low pass filters, where the double-frequency component like $2\omega_c$ will disappear. What is the output of the low pass filters, denoted as $I_{out}(t)$ and $Q_{out}(t)$?
- Given $I_{out}(t)$ and $Q_{out}(t)$, can you accurately recover $I_{in}(t)$ and $Q_{in}(t)$? If yes, mathematically give the relationship between $I_{in}(t)$, $Q_{in}(t)$ and $I_{out}(t)$, $Q_{out}(t)$. If no, give the reasons.

HINT: You need to use the trigonometric identities when expressing the I and Q data.

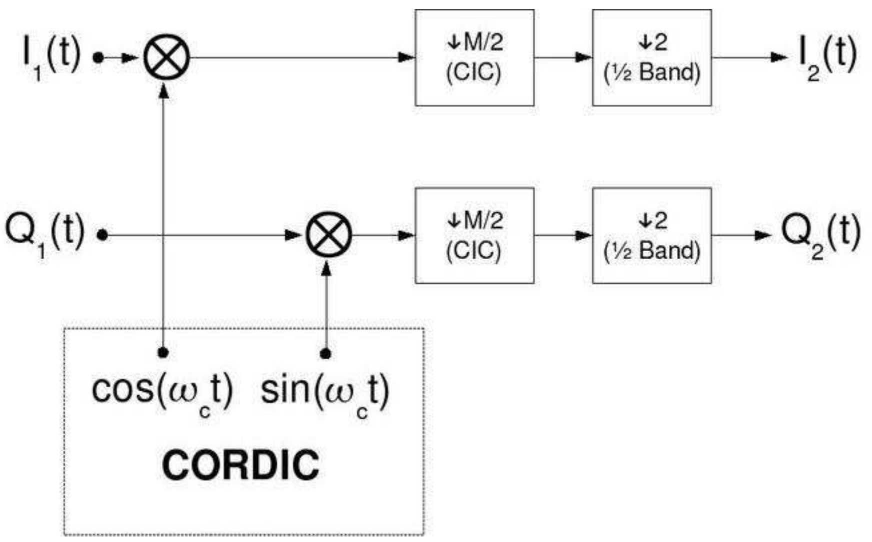


Figure 15: The structure of digital down converter on receiver path, where the input RF signal $I_1(t)$ and $Q_1(t)$ is first down-converted to the IF band by a carrier frequency of ω_c, and then decimated with a factor M. Please note, the $I_1(t)$ and $Q_1(t)$ here is different from the $I_1(t)$ and $Q_1(t)$ in Figure 14.

4 Open-ended Design Problem: Frame Synchronization

4.1 Frame Synchronization

Frame synchronization is the process in the telecommunications transmission system to align the digital channel (time slot) at the receiving end with the corresponding time slot at the transmission end as it occurs. For example, you transmit a packet which contains numerous frames. At the receiver side, you want to know where a specific frame actually starts, then you need to implement frame synchronization.

Frame synchronization involves the following steps: In the first step, the transmitter injects a fixed-length symbol pattern, called a marker, into the beginning of each frame to form a marker and frame pair, which is known as a *packet.* Packets are then converted from symbols into a waveform and transmitted through the channel. The receiver detects the arrival of packets by searching for the marker, removes the markers from the data stream, and recovers the transmitted messages. Marker detection is the most important step for frame synchronization.

In this problem, we will first introduce two Simulink models that implements frame synchronization. In these models, Barker code is employed as the marker. Then, you are going to build upon the second model and incorporate the USRP2 into it. In the end, you need to transmit "Hello world" from end to end using two USRP2s. The purpose of this problem is to prepare you for the course design project, in which you will your values in a serial string format.

4.2 Barker Code

A Barker code is a sequence of N values of +1 and -1:

$$a_j \text{ for } j = 1, 2, ..., N \tag{7}$$

such that:

$$\left| \sum_{j=1}^{N-v} a_j a_{j+v} \right| \leq 1 \tag{8}$$

for all $1 \leq v < N$.

Barker codes are commonly used for frame synchronization in digital communication systems. Barker codes have a length of at most 13 and possess low correlation sidelobes. A correlation sidelobe is the correlation of a codeword with a time-shifted version of itself. An example of autocorrelation function of Barker-7 code is shown in Figure 16. It is obvious from this figure that the sidelobes are low.

4.3 Simulink Models

In this section, we will introduce two Simulink models that implements frame synchronization using Barker code as the marker. The first model illustrates the basic idea and the second model employs the idea.

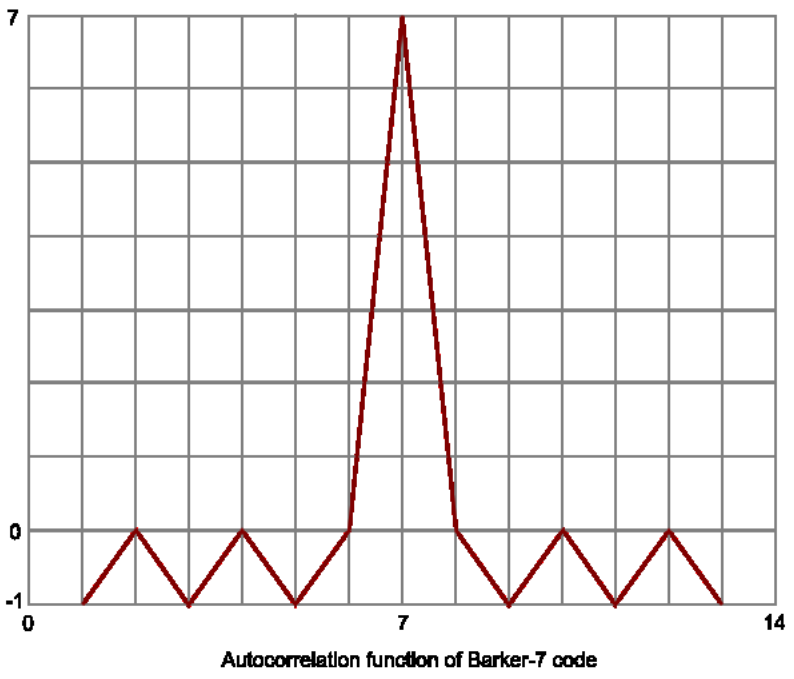
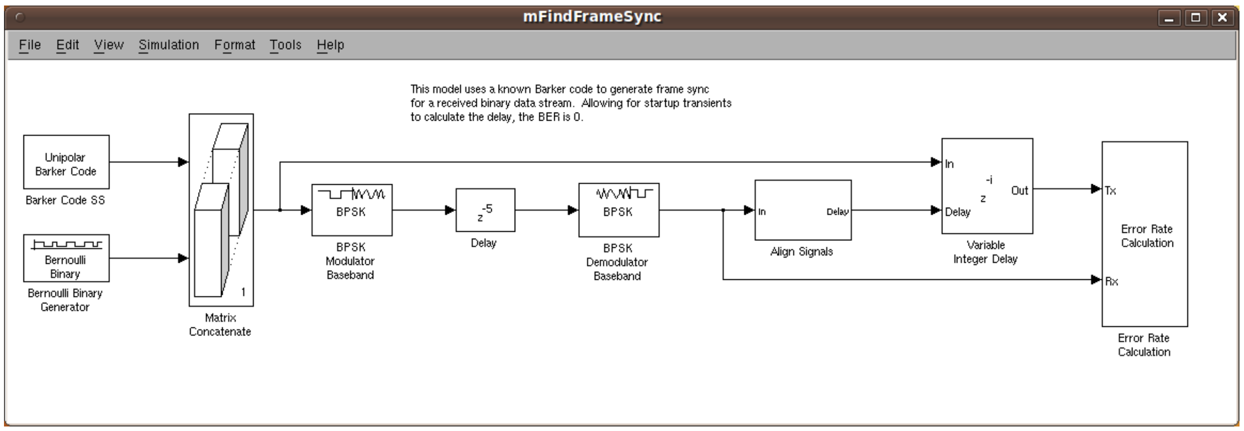


Figure 16: Autocorrelation function of Barker-7 code, which has a low sidelobes.



This model uses a known Barker code to generate frame sync for a received binary data stream. Allowing for startup transients to calculate the delay, the BER is 0.

Figure 17: Simulink model that realizes frame synchronization using Barker code.

4.3.1 Basic Model

Download and open `mFindFrameSync.mdl` from course website, as shown in Figure 17.

There are several key blocks in this model:

- `Barker Code Generator`: In this block, you specify the length of the Barker code, and this block will give the corresponding codeword.
- `Bipolar to Unipolar Converter`: The Barker code is either 1 or -1. However, the Bernoulli binary is either 1 or 0. In order to attach the Barker code to your frame, you need to change its format.
- `Matrix Concatenate`: The Barker code is attached to the beginning of each frame to form a marker and frame pair, which is known as a packet.
- `Delay`: This block is used to simulate a real channel, which incurs a delay of 5.
- `Error Rate Calculation`: This block uses the delay you have found to calculate the computation delay and verifies whether you have found the correct. If it is, the error rate should be 0. You can run the model and see.

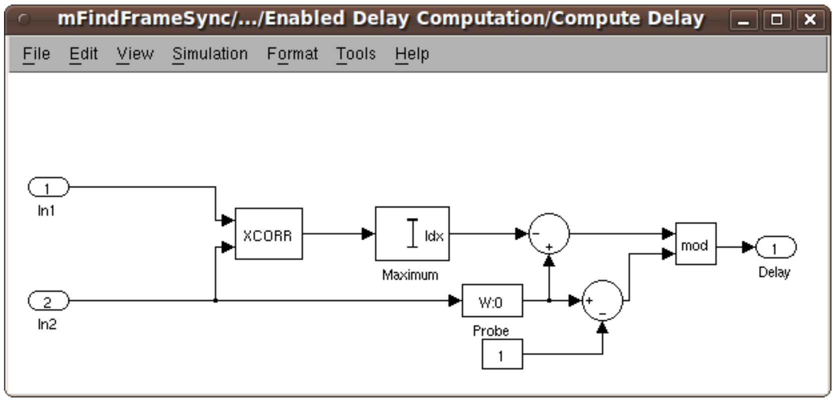
There are two key subsystems in this model:

- `Compute Delay`: Align Signals → Enabled Delay Computation → Compute Delay, as shown in Figure 18(a).
  This subsystem calculates the delay of the channel by detecting the peak of the correlation of the received data and the Barker code.
- `Consecutive Delay Comparison`: Align Signals → Consecutive Delay Comparison, as shown in Figure 18(b).
  This subsystem checks whether the calculated delay remains the same for numDelayCalcs iterations. If it is, the `Enabled Delay Computation` subsystem is disabled, such that you don't need to calculate the delay again and again. This subsystem guarantees the result of this model, and also improves the efficiency.

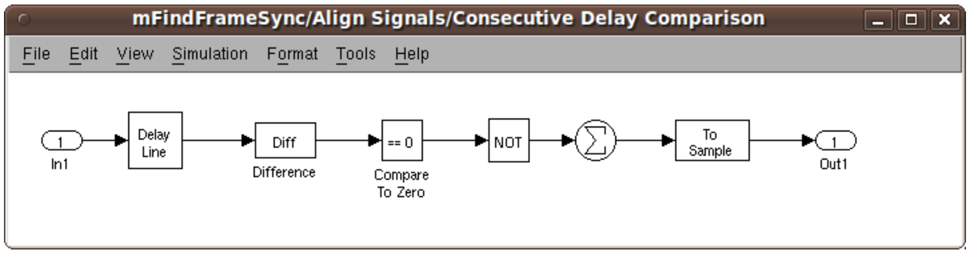
4.3.2 Application of the Basic Idea

According to the basic idea, we can construct a more advanced digital communication system. Download and open `mFindFrameStart.mdl` from course website, as shown in Figure 19. Previously, what we have done is to transmit binary bits. However, using this model, we can transmit a sentence, such as "Hello world". Therefore, in addition to this Simulink model, we need another m-file to conduct the converter between characters and binary bits. You can download the m-file `charToBitsAndBack.m` from course website.
The variable 'Delay' is a most important value in this model, which provides the information concerning where the start of the frame is, such that the receiver could know where should be the starting point of decoding.

Compared to the basic model, there are three different blocks:

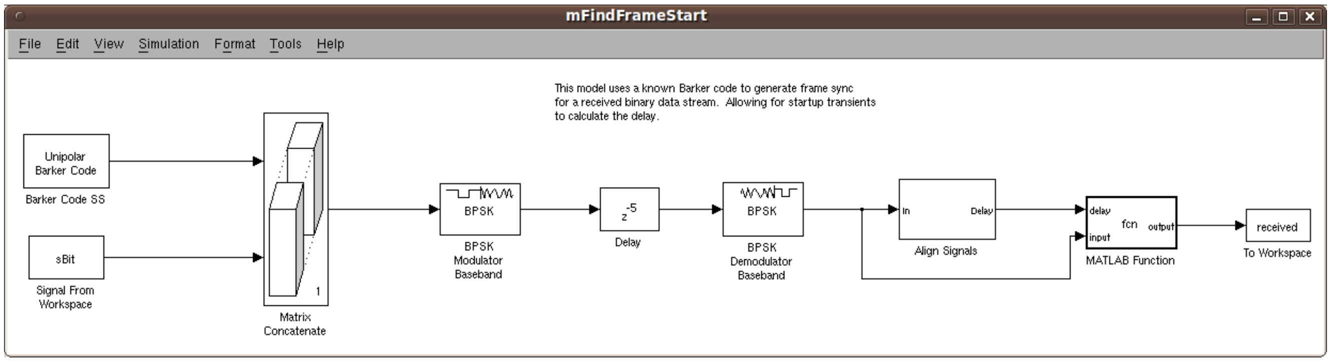

mFindFrameSync/.../Enabled Delay Computation/Compute Delay
File Edit View Simulation Format Tools Help
1
In1
2
In2
XCORR
Idx
Maximum
W:0
Probe
1
mod
1
Delay

(a) `Compute Delay` subsystem.


mFindFrameSync/Align Signals/Consecutive Delay Comparison
File Edit View Simulation Format Tools Help
1
In1
Delay Line
Diff
Difference
== 0
Compare To Zero
NOT
To Sample
1
Out1

(b) `Consecutive Delay Comparison` subsystem.

Figure 18: Two key subsystems in Simulink model `mFindFrameSync.mdl`.



This model uses a known Barker code to generate frame sync
for a received binary data stream. Allowing for startup transients
to calculate the delay.

Figure 19: Simulink model that realizes frame transmission using Barker code.

- `Signal From Workspace`: In this block, you specify the source from workspace. For example, we would like to transmit "Hello world", so we run `charToBitsAndBack.m` and get the corresponding bit streams. We save this bit stream in a variable called 'sBit' so that it can be used as the signal source. In this block, 'samples per frame' should be the same as the length of 'sBit'. Although the length of bit stream for "Hello world" is 77, 10 zeros are added to the end of this bit stream, so the length of 'sBit' becomes 87.

- `MATLAB Function`: We use a MATLAB Function to pick up the useful information out of a frame. In this example, delay+13+1 is the first bit in the stream and delay+13+77 is the last bit. 13 corresponds to the length of the Barker code, and 77 is the length of bit stream for "Hello world". Therefore, if you want to transmit some other sentences, or you want to use a Barker code of some other length, these values need to be modified.

- `To Workspace`: We save the useful bits to workspace and these bits can be converted to ASCII characters using the second half of `charToBitsAndBack.m`.

4.3.3 Incorporating USRPs

When incorporating USRPs into the second model, there are two things you need to pay attention to. The first one is the frame size, since USRP2 Transmitter and Receiver blocks require a frame size of 358. The second thing is the modulation scheme, you need to use DBPSK to ensure the best performance of bit transmission.

4.4 Hints for Implementation

- You will need to buffer N frames of data
  - Makes this less "real-time", but that is OK.
  - Trade-off between buffer size and latency.
- Try working with the raw data offline in MATLAB before trying out rea-time data retrieval or/and Simulink model that do this automatically

4.5 Hints for Debugging

- Collect incoming data from USRP2 block to .mat file
- Xcorr Barker sequence with stored incoming data in MATLAB
- Find the mode of the largest correlation value every 179 samples (visualize it, don't expect it to be "13")
- Take the mean of the mode (mode can be off by ± 1)
- Feed incoming data back into your Simulink receiver model (do not use USRP2 again) and fix delay
- Once verified, implement entirely in Simulink

5 Lab Report Preparation & Submission Instructions

Include all your answers, results, and source code in a lab report formatted as follows:

- Cover page: includes course number, laboratory title, names and student numbers of team, submission date
- Table of contents
- Pre-lab
- Responses to laboratory questions and explanation of observations
- Responses to open-ended design problem
- Source code

Please include images and outputs wherever possible, as well as insights on your laboratory.

Each group is to submit a single report electronically (in PDF format not exceeding 2MB) to `alexw@ece.wpi.edu` by scheduled due date. Reports that do not meet these specifications will be returned without review.

References

[1] Josh Blum. GNU Radio Companion. `http://www.joshknows.com/grc`.

[2] GNU Radio. GNU Radio Website. `http://www.gnuradio.org`.

[3] The MathWorks. Matlab documentation. `http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/index.html?/access/helpdesk/help/toolbox/matlab/`.

[4] The MathWorks. Simulink documentation. `http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/index.html?/access/helpdesk/help/toolbox/simulink/`.

[5] Jeffrey Reed. *Software Radio*. Prentice Hall, Englewood Cliffs, 2002.

[6] Michael Rice. *Digital Communications*. Pearson/Prentice Hall, Upper Saddle River, 2009.

[7] Dennis Silage. *Digital Communication Systems using MATLAB and Simulink*. Bookstand Publishing, Gilroy, CA, 2009.

[8] Alexander M. Wyglinski. *Physical Layer Loading Algorithms for Indoor Wireless Multicarrier Systems*. PhD thesis, McGill University, Montreal, QC, Canada, 2004. Appendix E.