LAB 7

Amplitude-Shift Keying

Prerequisite: Lab 2 – Amplitude Modulation

7.1 Objective

Amplitude-shift keying (ASK) is the simplest form of digital modulation. We will use it to provide an introduction to digital communications, and as a vehicle to introduce some of the features that are common to digital communication systems, such as symbol mapping, pulse shaping, matched filtering, threshold detection, and pulse synchronization.

In this lab project, design of the transmitter and design of the receiver each present challenges and opportunities for investigation. The lab project is consequently divided into several parts. The transmitter part investigates creation of the ASK signal and the effect of transmitted pulse shape on the bandwidth of the transmitted signal. The receiver part investigates demodulation, matched filtering, and signal detection. There is a third part investigating alignment of the receiver and transmitter bit streams.

7.2 Part 1: Transmitter

Background

ASK is simply AM with a binary message waveform. To illustrate, suppose $m(t)$ is a binary message represented in a polar, non-return-to-zero (NRZ) format, as shown in Figure 1.

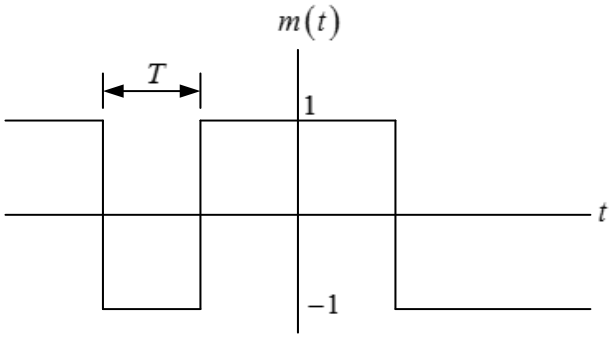

Figure 1. Binary Message Waveform

In the polar NRZ format, a binary 1 is represented by a pulse of amplitude $+1$ and a binary 0 by a pulse of amplitude -1. Each pulse has a duration of T seconds. Let us define an AM signal as

$$g_{ASK}(t) = A\left[1 + m(t)\right]\cos(2\pi f_c t), \qquad (1)$$

where f_c is the carrier frequency and A is the carrier amplitude. It is evident in Eq. (1) that either $g_{ASK}(t) = 2A\cos(2\pi f_c t)$ or $g_{ASK}(t) = 0$, depending on whether the corresponding message bit is a 1 or a 0. Thus the carrier is turned "on" to transmit a 1 and "off" to transmit a 0. This mode of ASK is sometimes referred to as "on-off" keying.

The pulse duration T that appears in Figure 1 will be called the "symbol time" in this and subsequent lab projects. The "symbol rate" is then $1/T$. In a binary modulation method such as ASK, the symbol rate and the bit rate are identical. We will encounter other modulation methods, however, such as phase-shift keying, in which multiple bits can be transmitted on each symbol and the bit rate may therefore be faster than the symbol rate.

As straightforward as ASK is, several distinct steps are needed to actually produce a modulated signal. These are:

1. Symbol Mapping. The input data arrives as a stream of bits. Bits can be represented in any of a variety of formats. We will see below that the *MT Generate Bits* function produces an array of bytes (8-bit integers) containing the numbers 1 and 0. A bit stream can also be represented as a Boolean array. In the symbol mapping step, the bits are replaced by numerical values. For ASK we will represent a binary 1 by the complex double $1 + j0$ and a binary 0 by the complex double $-1 + j0$. Note that we are representing bits by complex numbers, even though the imaginary parts are zero. This is because the USRP requires a complex-valued input, and because in a future lab we will use the imaginary part to carry additional data. The complex numbers that represent the input bits are known as "symbols." Table 1 shows the ASK symbol mapping.

Table 1. ASK Symbol Mapping

Bit Value	Symbol
0	$-1 + j0$

1	$1+j0$

2. Upsampling. We will see below that the symbols are carried on pulses whose shape is important in establishing the bandwidth of the transmitted signal. As a first step toward replacing symbols by pulses, we will place $L-1$ zeroes after each symbol. This produces a sample interval of

$$T_x = \frac{T}{L}, \tag{2}$$

or a sample rate of

$$\frac{1}{T_x} = L\frac{1}{T}. \tag{3}$$

A higher upsampling factor L makes the D/A conversion in the transmitter easier, but requires faster digital processing. We will use $L = 20$ in this lab project. The *Upsample* function is made to order for implementing this step.

3. Pulse Shaping. If the upsampled signal is applied to a filter whose impulse response $g_{TX}[n]$ is a rectangular pulse of unit amplitude and length L samples, then at the filter output, each symbol will be represented by a rectangular pulse. Figure 2 shows the effect of upsampling and filtering. Waveform (a) represents the symbol sequence, with symbols occurring every T seconds. Waveform (b) shows the symbol sequence after upsampling. Waveform (c) shows the upsampled symbol sequence after filtering by the pulse-shaping filter. Note that waveform (c) is a discrete-time version of the polar NRZ message waveform shown in Figure 1. An important advantage to following this particular sequence of steps to generate the message waveform is that the impulse response $g_{TX}[n]$ of the pulse-shaping filter does not have to be a rectangular pulse. We will see later in this lab project that there can be advantages to using other pulse shapes.

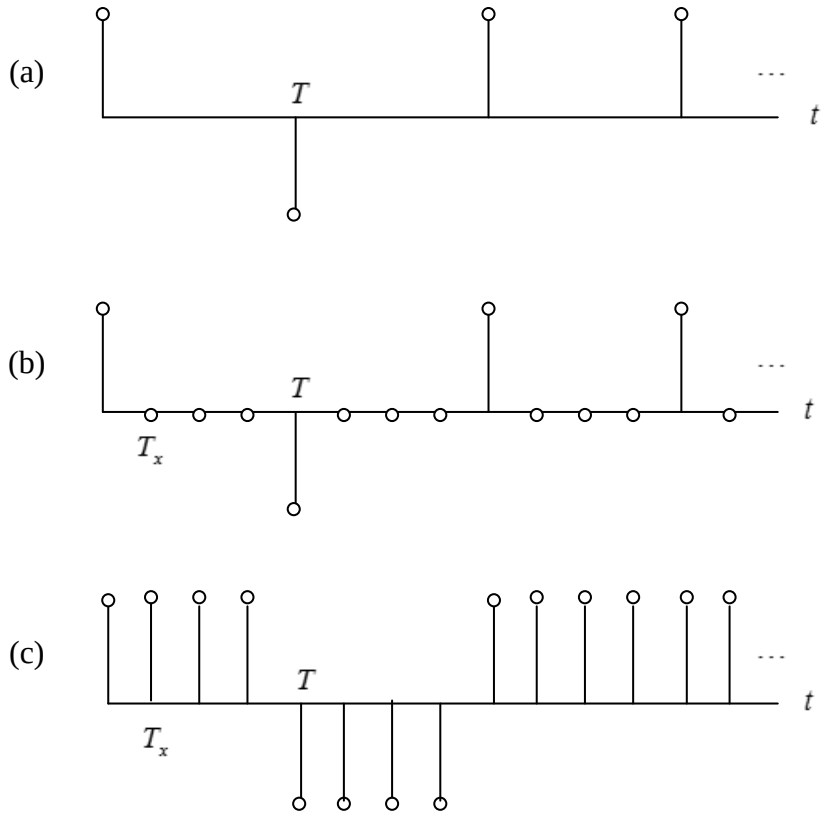

Figure 2. (a) Symbol sequence, (b) Upsampled symbol sequence, (c) Filtered upsampled symbol sequence.

4. Modulation. If $m[n]$ represents the message waveform at the transmitter filter output, then the final step is to apply Eq. (4):

$$\tilde{g}_{ASK}[n] = A(1+m[n]), \tag{4}$$

where the constant A is chosen to keep the magnitude of $\tilde{g}_{ASK}[n]$ less than 1. This signal can be sent to the USRP transmitter. The USRP transmitter will perform the D/A conversion and multiplication by the carrier $\cos(2\pi f_c t)$.

Prelab

1. Create a program to generate an ASK signal using the USRP. A template for the transmitter has been provided in the file *ASKTxTemplate.gvi*. This template contains the four functions for interfacing with the USRP along with *MT Generate Bits* from the Modulation Toolkit. *MT Generate Bits* will create a pseudorandom sequence of bits that can serve as a data sequence for testing your ASK system. Note that by default, *MT Generate Bits* will produce the same sequence of bits every time you run the program.

This is useful for debugging, but if you would like to generate a different sequence of bits every time, wire a random number to the "seed in" input. The steps below contain details about how to create the required transmitter.

2. First do the symbol mapping, as shown in Table 1. This will convert the integers 0,1 from *MT Generate Bits* to complex doubles $\pm 1 + j0$. In future lab projects, where the symbol mapping may be more elaborate, this might be implemented as a sub-vi. In this lab project the symbol mapping is relatively simple, and a sub-vi implementation is optional.

3. Upsample using *Upsample* from the Analysis→Signal Processing→Conditioning subpalette. In this lab project you are given control inputs to set the symbol rate $1/T$ and the IQ rate $1/T_x$. Set the symbol rate to 10,000 symbols/s and the IQ rate to 200×10^3 Sa/s. Use these two inputs to calculate the upsampling factor L.

3. Use *MT Generate Filter Coefficients* from the Modulation Toolkit to generate the pulse shaping filter. (*MT Generate Filter Coefficients* can be found on the Analysis→Communications→Digital→Utilities subpalette.) Set the modulation type to ASK, and the pulse-shaping filter "samples per symbol" to your calculated value of L. Create a front-panel control for "pulse shaping filter" and set this initially to "none." The setting of "none" will generate rectangular pulses. ("none" does not mean that there is no filter!) Wire the "pulse shaping filter coefficients" output to the "Y" input of a *Convolution*. The *Convolution* is available from the Analysis→Signal Processing→ Operation subpalette. Wire the output from your upsampler to the "X" input of the *Convolution*.

4. Normalize the amplitude of your filtered message signal to a maximum absolute value of 1. This will be important later when we investigate alternative pulse shapes. Check out the *Quick Scale 1D* in the ExternalFiles folder for finding the maximum of the absolute value. To ensure that your scaled message remains complex-valued, use a separate division function to do the actual scaling. (That is, do not use the $Y[i] = X[i]/\mathrm{Max}|X|$ output of *Quick Scale 1D*.)

5. Implement Eq. (4). Let the constant A be $1/2$, so that $\tilde{g}_{ASK}[n]$ varies between zero and one. Combine $\tilde{g}_{ASK}[n]$ with T_x using the *Build Waveform function* provided in the template to produce the "Baseband Signal." Note that $1/T_x$ is available as the "actual IQ rate." Also connect $\tilde{g}_{ASK}[n]$ to the "data" input of the *Write Tx Data* function.

6. An *FFT Power Spectrum for 1 Chan* has been provided in the template to allow you to observe the spectrum of the transmitted signal.

This completes construction of the ASK transmitter. Save your transmitter in a file whose name includes the letters "ASKTx" and your initials (e.g. *ASKTx_BAB.gvi*).

Lab Procedure

1. Connect a loopback cable and attenuator between the TX 1 and RX 2 connectors of the USRP. Connect the USRP to your computer and plug in the power to the USRP. Run LabVIEW and open the transmitter that you created in the prelab.

2. Ensure that the transmitter is set up to use

   Carrier Frequency: 915.1 MHz (Note: The 100 kHz offset from the receiver carrier frequency is deliberate.)

   IQ Rate: 200 kHz. Note: This sets the value of $1/T_x$.

   Gain: 0 dB

   Active Antenna: TX1

   Symbol rate: 10,000 symbols/s

   Message Length: 1000 bits

   Pulse shaping filter: None

   Run the transmitter. Use the large STOP button on the front panel to stop transmission connectors.

3. After running the transmitter, observe the spectrum of the transmitted signal. You should be able to clearly see the carrier at frequency "zero." Two additional features are significant: the bandwidth and the rate of spectral rolloff. Using the Capture Data

feature or using cursors on the Power Spectrum graph, measure the null-to-null bandwidth of the transmitted signal. Relate this bandwidth to the symbol rate $1/T$. The rate of spectral rolloff is a measure of the interference that your signal will cause to signals using nearby carrier frequencies. Print a copy of the spectrum for comparison with the spectrum you will obtain in Step 4 below.

4. Rectangular pulses are rarely used in practice because of the very gradual spectral rolloff they produce. Change the pulse-shaping filter to "Root Raised" for a root-raised-cosine filter. Run the program examine the spectrum again. Measure the null-to-null bandwidth of the transmitted signal. Print a copy of the spectrum and compare the rolloff rate with the spectrum you obtained using rectangular pulses.

Questions

1. In step 2 you are given $\frac{1}{T} = 10{,}000\,\frac{symbols}{s}$ and $\frac{1}{T_x} = 200 * 10^3\,\frac{Sa}{s}$. Find the corresponding value for the number of samples per symbol L.

2. Relate the symbol rate to the null-to-null bandwidth of the ASK signal for (a) rectangular and (b) root-raised-cosine pulses.

3. Compare the rates of spectral rolloff of the transmitted signal for rectangular and root-raised-cosine pulses.

7.3 Part 2: Receiver

Background

An ASK receiver begins as an analog AM receiver. We will offset the transmitter and receiver carrier frequencies by 100 kHz, so that the signal retrieved from the USRP receiver will be an AM signal having an "intermediate" carrier frequency of 100 kHz. The retrieved AM signal will be passed through a bandpass "intermediate frequency" filter and then demodulated using an envelope detector. The envelope detector is implemented by taking the magnitude of the bandpass filter output, and then lowpass filtering using a second filter. If you completed Lab 2, Amplitude Modulation, these steps should be familiar.

For digital communications, the lowpass filter should be designed to minimize the effects of noise and to also minimize the effects of intersymbol interference that can be caused when the filtered received pulses overlap. The best filter for eliminating noise is a so-called "matched" filter. A matched filter has a frequency response whose magnitude matches the magnitude of the frequency response of the transmitter's pulse-shaping filter. That is, if $g_{RX}[n]$ is the impulse response of the receiver's filter, then we want $|G_{RX}(e^{j\omega})| = |G_{TX}(e^{j\omega})|$. By using *MT Generate Filter Coefficients* at both the transmitter and receiver, we will ensure that the appropriate receiver filter is chosen to match the pulse-shaping filter at the transmitter.

To complete the digital receiver, several additional steps follow the AM demodulator.

1. Pulse Synchronization. The AM receiver output is an analog baseband signal that must be sampled once per symbol time, i.e. once every T seconds. Because of filtering and propagation delays and distortion caused by the communication channel, it is necessary to determine the optimum time to take these samples. A function *PulseAlign(real)* has been provided in the *BasicUSRPLabs folder* to align the baseband signal so that the sample at index 0 is the correct first sample.

2. Sampling. The *Decimate* function will sample the aligned baseband waveform at index 0 and every T seconds thereafter.

3. Detection. Once the baseband waveform has been sampled, each sample must be examined to determine whether it represents a symbol of value 1 or a symbol of value 0.

4. Symbol Mapping. The detected symbol values must be converted to bits. For ASK, this step is easily included in the detection step.

Prelab

1. A template for the receiver has been provided in the file *ASKRxTemplate.gvi*. This template contains the six interface functions for interfacing with the USRP.

We want the receiver to capture two frames of data each time it is run. Since the receiver's starting point is random, this will ensure that there will always be one complete frame in the captured data. Using the message length and symbol rate available from front panel inputs, and the "Actual IQ Rate" available from *Configure Signal*, have the receiver calculate the number of samples in a frame. Then double this number and provide the result as the "number of samples" input to the *Fetch Rx Data*.

Fetch Rx Data returns a complex double waveform. Pass the complex double waveform through a bandpass filter. Filters can be found in the palette Analysis>>Signal Processing>>Filters. Configure a fifth-order Chebyshev Filter with a high cutoff frequency of 110 kHz and a low cutoff frequency of 90 kHz. The default passband ripple of 0.1 dB is acceptable.

2. Get the "Y" component of the waveform at the output of the Chebyshev filter by using the *Waveform Properties* function. Extract the real part of the complex array from the "Y" component of the waveform. To obtain the envelope, take the absolute value and pass the result through a matched filter. The absolute value functions as a full-wave rectifier. For the matched filter, use *MT Generate Filter Coefficients* just as you did for the transmitter. Set the modulation type to ASK, and calculate the "matched samples per symbol" M from the "actual IQ rate ($1/T_z$)" and the symbol rate ($1/T$) obtained from the front-panel control. Create a front-panel control for "pulse shaping filter" and set this initially to "none." The setting of "none" will generate a matched filter with a rectangular impulse response (not the absence of a filter, as you might imagine). Wire the "matched filter coefficients" output to the "Y" input of a *Convolution*. The *Convolution* is available from the Analysis→Signal Processing→ Operation subpalette. The output of your matched filter should be connected to the *Cluster Properties function* provided in the template. The *Cluster Properties function* feeds the Baseband Output graph.

3. A convenient way to visualize the output of a digital demodulator is the so-called "eye diagram." An eye diagram is a plot of the baseband output signal with the horizontal axis scaled to be one or two symbol times long and successive symbols superimposed.

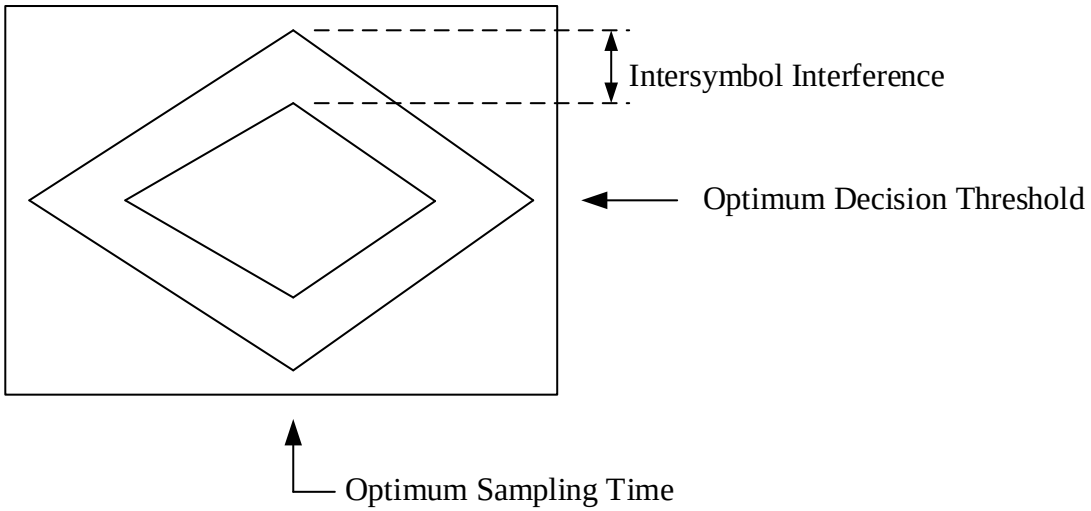

Figure 3. Stylized Eye Diagram

Figure 3 shows a stylized eye diagram. Some of the useful information that can be learned from the eye diagram is shown in the figure.

The Modulation Toolkit function *MT Format Eye Diagram* has been provided in the receiver template. Wire the baseband output waveform to the "waveform" input. The "symbol rate (Hz)" input value is available from the front panel control. Set the "eye length" parameter to 2.

4. Place the *PulseAlign(real)* on your block diagram and wire the baseband output waveform to the "input waveform" input and wire the calculated M samples/symbol value to the "receiver sampling factor" input.

   Once the baseband waveform is aligned, it can be sampled. *Decimate (single shot)* can be obtained from the Analysis→Signal Processing→Conditioning subpalette. The "decimating factor" is M .

5. To determine whether each received sample is more likely to represent a 1 or a 0, the sample must be compared with a threshold. Use the *Mean (DBL)* function to compute the threshold. This function is one configuration of the *Measures of Mean* function which can be found in the Math→Statistics subpalette. The result of the comparison is the receiver's digital output. The output of the comparison will be a Boolean array. You can convert this array to an integer array by using a *Boolean To Integer* function.

This completes construction of the ASK receiver. Save your receiver in a file whose name includes the letters "ASKRx" and your initials (e.g. *ASKRx_BAB.gvi*).

Lab Procedure

1. Ensure that the receiver is set up to use

   Carrier Frequency: 915.0 MHz

   IQ Rate: 400 kHz

   Gain: 0 dB

   Active Antenna: RX2

   Symbol rate: 10,000 symbols/s

   Message length: 1000 bits

   Pulse shaping filter: none

2. Run the transmitter, then run the receiver. Once the receiver has acquired a block of data, you may stop the transmitter.

3. Use the horizontal zoom feature on the Baseband Output graph palette to expand the demodulated waveform so that you can see individual pulses. Ideally, rectangular pulses passed through the receiver's matched filter should produce triangular output pulses. Note whether the demodulated pulses have the expected shape.

4. Observe the eye diagram. Make note of the optimum sampling time and the presence of intersymbol interference.

5. Change the "pulse shaping filter" control at both the transmitter and the receiver to "Root Raised" for root-raised-cosine filters. Setting both the transmitter and receiver filters will ensure that the filters remain matched for optimum performance in the presence of noise. The cascade of two root-raised-cosine filters produces a raised-cosine pulse at the receiver filter output. The raised-cosine pulse is designed to minimize, or ideally eliminate, intersymbol interference.

Run the transmitter and then run the receiver. Once the receiver has acquired a block of data, you may stop the transmitter. Observe the baseband output signal and the eye diagram. Comment on the changes to the eye diagram. Has intersymbol interference been reduced?

6. To see the effect of pulse synchronization, move the waveform input of *MT Format Eye Diagram* to the "aligned waveform" output of *PulseAlign(real)*. Run the transmitter and receiver again. Observe the eye diagram. What is the optimum sampling time now?

Questions

1. Give a formula showing how the IQ Sampling Rate, the symbol rate $1/T$, and the number of samples per symbol M are related. Determine the value of M for an IQ Sampling Rate of 1 MHz and a symbol rate of 10,000 symbols/s.

2. Note that the IQ sampling rate at the receiver is different from the IQ sampling rate at the transmitter. Using a higher IQ sampling rate requires faster digital processing. What is the advantage to using a higher IQ sampling rate at the receiver? (Hint: It has something to do with the action of the *PulseAlign(real)* and the value of M.)

3. Compare the rectangular and raised-cosine pulse shapes by examining the eye diagrams. What evidence for intersymbol interference do you see in each case?

4. Using root-raised-cosine pulses and receiver filtering, observe the eye diagram when the input of *MT Format Eye Diagram* is set to "baseband output" and when the input of *MT Format Eye Diagram* is set to the "aligned waveform" output of *PulseAlign(real)*. What function is *PulseAlign(real)* performing?

5. The transmitter is programmed to generate the same "frame" of 1000 symbols over and over. The receiver grabs a single block of 2000 symbols each time it is run. Can you identify, by examining the receiver's baseband output plot, where the symbol sequence ends and starts over? Frame synchronization of the receiver is an essential component of a digital communication system. We will examine frame synchronization in the next part of this lab project.

7.4 Part 3: Aligning the Received Bits

At this point you have a working transmitter and receiver, but it is hard to know whether the two are working together correctly as a system unless you can compare the received bits with the transmitted bits and verify that the bit patterns are the same. To allow this comparison, it is necessary that the receiver recognize the beginning of the transmitted sequence. The *AddFrameHeader(real)*, available in the *BasicUSRPLabs folder*, inserts a specific 26-bit sequence at the start of transmission. At the receiver, the *FrameSync(real)*, also available in the *BasicUSRPLabs folder*, looks for this specific sequence and cuts off all bits received before this

frame header. The *FrameSync(real)* also cuts off the frame header. This way, the bit sequence at the output of the receiver should match the bit sequence sent to the transmitter.

1. Add the *AddFrameHeader(real)* to the transmitter. Place *AddFrameHeader(real)* after the symbol mapping, but before conversion of the symbols to complex.

2. Add the *FrameSync(real)* to the receiver. Place *FrameSync(real)* immediately following *Decimate*. Wire the output of *Decimate* to the "Sampled Input" of *FrameSync(real)*. Leave the remaining inputs of *FrameSync(real)* unwired. Wire the "Aligned Samples" output of *FrameSync(real)* to the threshold comparison function and to *Mean* described in Receiver Prelab, Section 5.

   Wire the array of output bits from the threshold comparison to the "array" input of an *Array Subset function*. Set the "index" input to zero, and set the "length" input to the length specified by the "message length" control. Display the output of *Array Subset function* as "Output bits" on the receiver front panel.

   Note that the "Output Signal" and "max index" outputs of *FrameSync(real)* will not be used in this lab project.

3. Run the transmitter; then run the receiver; then stop the transmitter. Compare the first dozen or so received bits with the corresponding transmitted bits.

4. Measurement of the bit error rate (BER) can be automated using the *MT Calculate BER* from the Modulation Toolkit (Analysis→ Communications→ Digital→Measurements subpalette). From the Configure ribbon, select "PN Fibonacci." Set the "BER trigger threshold" to 0.4. Connect indicators to the "BER" and "trigger found?" outputs. When you run the program, "trigger found?" will be true whenever the measured BER is below the BER trigger threshold. Run the transmitter and receiver. The measured BER should be identically zero unless you add noise.

Questions

1. In step 3, do the received bits match the transmitted bits?

2. What would the measured BER be if there were something wrong with the receiver's bit alignment? Note that when the measured BER is higher than the BER trigger threshold, the "trigger found?" output will be false, and any BER reading shown will not be meaningful.

7.5 Report

Prelab

Hand in documentation for the programs you created for the transmitter and receiver. Also include documentation for any functions you created. To obtain documentation, print out legible screenshots of the front panel and block diagram.

Lab

Submit the programs you created for the transmitter and receiver. Also submit any functions you created. Be sure your files adhere to the naming convention described in the instructions above.

Resubmit documentation for any functions you modified during the lab.

Answer all of the questions in each of the sections marked *Questions* above.