LAB 11

Equalization

Prerequisite: Lab 10 – The Eye Diagram

11.1 Objective

In most digital communication systems, the transmitter's pulse shaping filter and the receiver's matched filter are designed so that the pulses that emerge from the receiver's matched filter do not exhibit any intersymbol interference. As we discussed in the eye diagram lab project, however, the communication channel often introduces additional filtering that can result in intersymbol interference.

One way to deal with intersymbol interference created by the communication channel is to add an additional filter in the receiver. Ideally, this new filter will have a response that is "inverse" to the filtering caused by the channel. A filter intended to counter the adverse effects of channel filtering is called an *equalizer*.

In wireless systems, the channel distortion is often caused by multipath propagation. That is, the received signal may include reflections from buildings and other objects in the environment. The distortion that is observed in any particular wireless link will depend on the location of the transmitter, the location of the receiver, and the locations of nearby reflecting objects. Since the transmitter, the receiver, and even some of the reflecting object can move, channel distortion can change gradually with time. In this kind of application it is helpful to have an equalizer that is *adaptive*; that is, we would like the equalizer to be able to change its properties slowly with time to follow changes in the channel.

In this lab project you will use the BPSK transmitter and receiver that you created for Lab 9 as a "test bed" system. The *Channel.gvi* that you used in the eye diagram lab project will create the intersymbol interference. The equalizer that we will investigate is provided in the Modulation Toolkit.

11.2 Background

Equalizers, particularly adaptive equalizers, are nearly always implemented as FIR filters. There are two reasons for this. First, and most important, a FIR filter is always stable. As a result, we do not have to be concerned that the equalizer might become unstable as it adapts. Second, it sometimes turns out that the "optimum" equalizer is not causal. A non-causal FIR filter can always be made causal by adding a finite delay. This easy fix cannot be applied to a non-causal IIR filter.

Experience shows that it is often possible to model the communication channel as an FIR filter as well. In a multipath environment successive reflections become weaker, and eventually drop below the noise floor. Commonly, only a few of the strongest reflections are significant causes of ISI. Unfortunately, the inverse of an FIR filter is always an IIR filter, and this means that an FIR equalizer will never be able to completely remove the ISI caused by an FIR channel. We will find that even a very long FIR equalizer always leaves a small amount of residual ISI.

Suppose that $\tilde{y}[k] = \tilde{y}(kT)$ represents the sampled output of the receiver's matched filter. The parameter T represents the time between samples and also the time between transmitted symbols. Recall from the eye diagram lab project that each sample $\tilde{y}[k]$ is the sum of a desired sample value, ISI, and noise. The block diagram representation of an FIR equalizer is shown in Figure 1.

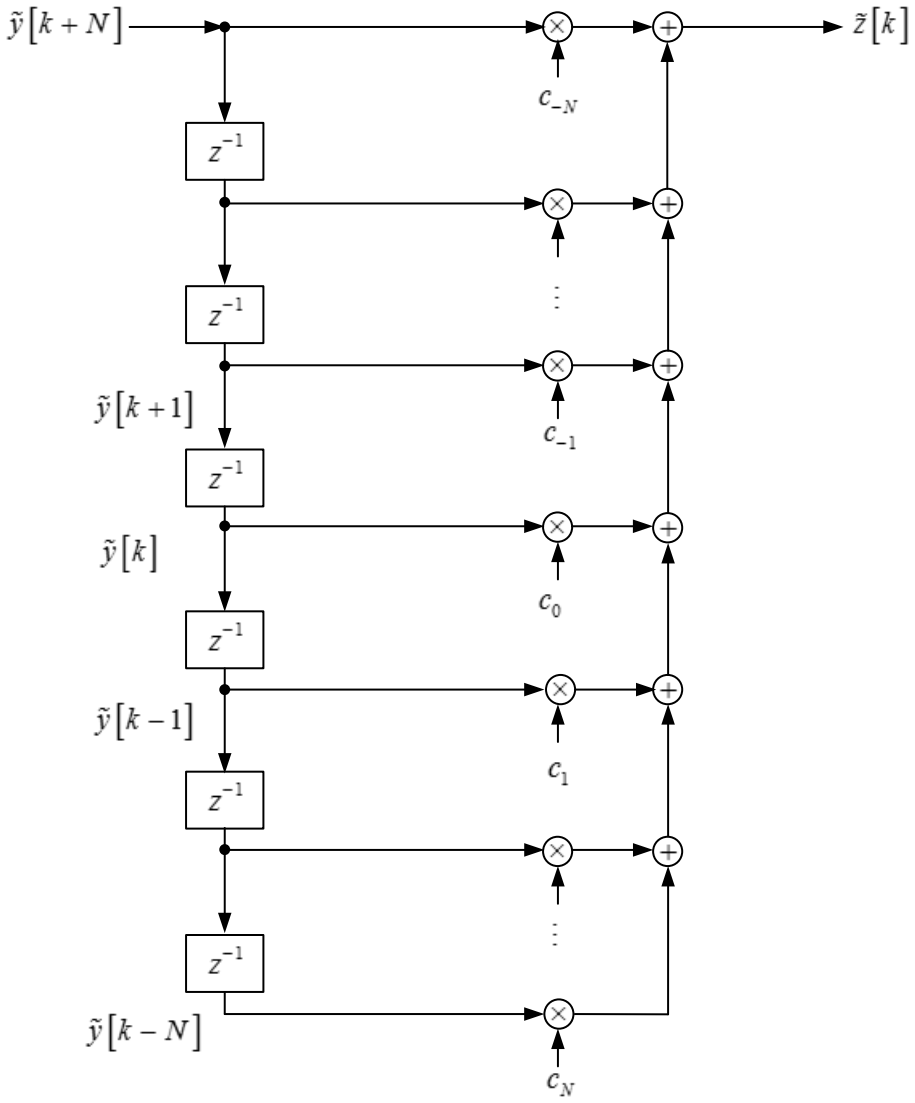

Figure 1. An FIR Equalizer

The output of the equalizer is denoted as $\tilde{z}[k]$. In the diagram, blocks designated z^{-1} represent one-sample delay elements. Arrows leading to multiplication by coefficients c_n are traditionally known as "taps," and the coefficients themselves are known as "tap gains." This block diagram represents the equation

$$\tilde{z}[k] = \sum_{n=-N}^{N} c_n y[k-n]. \tag{1}$$

Our goal in designing an equalizer is to find values for the tap gains $c_{-N},\ldots,c_N$ that will minimize any residual ISI. Even better, we would like to find a way to have the tap gains adjust themselves to reduce ISI as the equalizer runs.

To provide a basis for adjusting the equalizer tap gains $c_{-N},\ldots,c_N$, the receiver must know what data the transmitter is sending. Common practice is to begin transmission with a "training sequence" $b_k, k=0,\ldots,N_t-1$ that is known to the receiver. In this lab project we will use the same 26-symbol sequence that we are already using for frame synchronization. It is assumed that once the $N_t=26$ symbol training sequence has been received, the tap gains will have had time to adjust themselves to nearly their optimum values. Thus from this time on, the sequence $\tilde{z}[k]$ should be nearly ISI-free. Following the equalizer, the BPSK receiver performs detection and symbol mapping by comparing $\tilde{z}[k]$ with a threshold of zero. If $\hat{b}_k$ represents the output data, then

$$\hat{b}_k=\begin{cases}1, & \text{if } \tilde{z}[k]\geq 0\\ 0, & \text{if } \tilde{z}[k]<0.\end{cases} \tag{2}$$

What will the receiver use as a basis for adjusting the equalizer tap gains once the training sequence is complete? In a *decision-directed* equalizer, the decisions $\hat{b}_k$ are used to replace the training sequence bits b_k. Even when the probability of error is relatively high (e.g. 0.01), the decisions $\hat{b}_k$ are hardly ever wrong. It turns out that an occasional incorrect tap gain adjustment causes little change in the ISI output of the equalizer. Figure 2 shows the organization of the decision-directed equalizer.

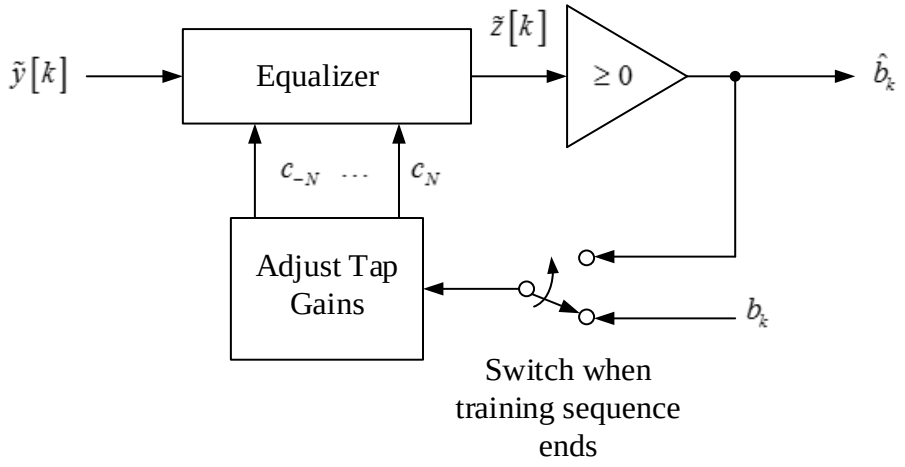

Figure 2. Decision-Directed Equalizer

There are a number of alternative criteria for deciding when the equalizer tap gains are optimally adjusted. One possibility is to adjust the tap gains to maximize the eye opening. It turns out, however, that equalizers have a tendency to amplify the noise that forms part of the input sequence $\tilde{y}[k]$. Maximizing the eye opening also tends to enhance the noise. An

alternative criterion minimizes the mean squared error between the sequence $\tilde{z}[k]$ and the symbol sequence a_k based on b_k. Since the mean squared error includes both ISI and noise, this criterion is less prone to noise enhancement than methods that ignore noise. Now if a_k is the symbol sequence corresponding to the training sequence bits b_k, we can write the mean squared error E as

$$\mathrm{E} = E\left(\tilde{z}[k] - a_k\right)^2, \tag{3}$$

where E is the expectation operator. Substituting Eq. (1) gives

$$\mathrm{E} = E\left(\sum_{n=-N}^{N} c_n \tilde{x}[k] - a_k\right)^2. \tag{4}$$

If we were designing a fixed equalizer, we would find the tap gains $c_{-N},\ldots,c_N$ that minimize E in Eq. (4). Instead, we want to find a way to let the tap gains adjust themselves. Suppose $c_{-N}^{(q)},\ldots,c_N^{(q)}$ represent the values of the tap gains after the q-th update. We can further update the tap gains according to the algorithm

$$c_n^{q+1} = c_n^q - \Delta\frac{\partial \mathrm{E}}{\partial c_n}, \quad n = -N,\ldots,N, \quad q = 0,1,\ldots. \tag{5}$$

This algorithm is an example of a *steepest descent* procedure, as it moves the coefficients in the direction of the negative of the gradient of the mean-squared error with respect to the coefficient values. This is the direction of the steepest descent down the slope to the minimum of the mean-squared error. The parameter Δ adjusts the step size. We can evaluate the derivative $\partial\mathrm{E}/\partial c_n$ by using Eq. (4):

$$\begin{aligned}\frac{\partial \mathrm{E}}{\partial c_n} &= E\left[2\left(\sum_{i=-N}^{N} c_i \tilde{y}[k-i] - a_k\right)\tilde{y}[k-n]\right] \\ &= 2E\left[\left(\tilde{z}[k] - a_k\right)\tilde{y}[k-n]\right] \\ &= 2E\left[e[k]\tilde{y}[k-n]\right], \quad n = -N,\ldots,N,\end{aligned} \tag{6}$$

where $e[k] = \tilde{z}[k] - a_k$ is the error signal (including both noise and ISI) at index k.

It turns out that the expectation operator in Eq. (6) poses something of a problem, since there is no way to evaluate it in an actual filter implementation. We can avoid the problem by approximating the expectation by the current value. That is,

$$E\left[e[k]\tilde{y}[k-n]\right] \cong e[k]\tilde{y}[k-n]. \tag{7}$$

Then the rule for updating the tap gains becomes

$$c_n^{q+1} = c_n^q - \Delta e[k]\tilde{y}[k-n], \quad n = -N,\ldots,N, \quad q = 0,1,\ldots. \tag{8}$$

Normally we will update the equalizer tap gains after every received symbol. Thus the index k and the index q increment together. We have

$$c_n^{k+1} = c_n^k - \Delta e[k]\tilde{y}[k-n], \quad n = -N,\ldots,N, \quad k = 0,1,\ldots. \tag{9}$$

The procedure expressed by Eq. (9) is called the *least mean square* (LMS) algorithm for adjusting the equalizer tap gains. We initialize the procedure by taking

$$c_n = \begin{cases} 1, & n = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

A block diagram of the LMS adaptive equalizer is shown in Figure 3.

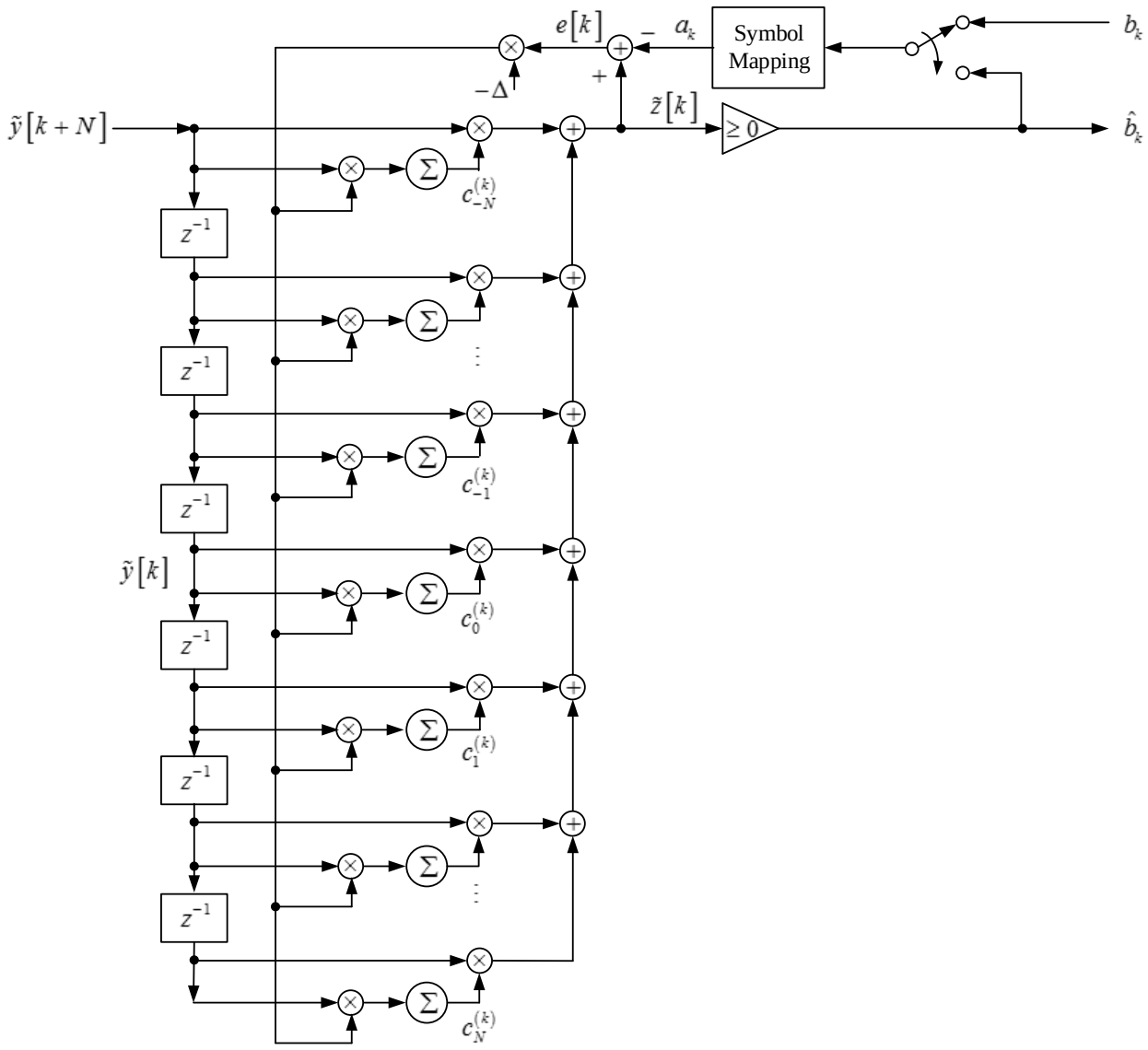

Figure 3. LMS Adaptive Equalizer

Note in Eq. (9) that if the step size Δ is small, the tap gain values will never change very much on a single update. This means that the tap gain values at any time are the accumulated sums of many small adjustments. Since accumulation is a form of averaging, Eq. (9) justifies our approximating the expectation operator by a single sample in Eq. (7). Further, since the tap gains change very little each time they are adjusted, a single symbol error does not push the equalizer very far out of adjustment when decision feedback is in use.

11.3 Pre-Lab

1. For this lab project you will use the BPSK transmitter and receiver you created for the phase-shift keying lab project. Be sure your receiver includes the *Channel.gvi* containing the channel model, and also includes the capability of displaying an eye diagram. You may use either *MT Format Eye Diagram* from the Modulation Toolkit, or the eye diagram program that you created for the eye diagram lab project. Wire the "ISI Channel Model" *Channel.gvi* input to a front panel control on your receiver.

2. Create a program to perform adaptive equalization. A template *EqualizerTemplate.gvi* has been provided to get you started. The inputs to your equalizer are to be connected as follows:

    a. Input Complex Waveform input to *Equalizer.gvi* will be connected to Output Complex Waveform on the *FrameSync(real)*. This waveform has been aligned to the start of the frame, but has not been downsampled. This waveform also includes the received header symbols that will be used to train the equalizer.
    b. Receiver Sampling Factor input to *Equalizer* will be connected to the number of samples per symbol parameter that is calculated in your BPSK receiver.
    c. Equalizer Length input to *Equalizer* should be connected to a control on your receiver front panel.

In addition to the inputs and outputs, the equalizer template contains a copy of the training sequence, an array containing the symbol map, and a cluster of "feedforward equalizer parameters." These parameters include the number of equalizer taps per symbol (one), and the values of step size Δ to use during training and during decision-directed operation (0.05 during training, 0.001 in decision-directed mode).

To complete the equalizer, you will need to add two functions from the Modulation Toolkit. These are *MT Generate System Parameters* from the Analysis→Communications→Digital→Utilities subpalette and *MT PSK Feedforward Equalizer* from the Analysis→Communications→Digital→Equalization subpalette.

Click on the MT PSK Feedfordward Equalizer function then, from the Configure ribbon, choose PSK(M) for the *MT Generate System Parameters*. Create a constant at the "PSK type" input and select "Normal." Wire your "Receiver Sampling Factor" to the "samples per symbol" input. Wire a constant of value 2 to the "M-PSK" input. The only output of the *MT Generate System Parameters* that you will use is the "PSK system parameters" cluster. Wire this to the *Cluster Properties function* provided in the template to replace the default symbol map with the symbol map provided. The modified PSK system parameters cluster will be used by the *MT PSK Feedforward Equalizer*.

From the Configure ribbon, choose Specify Length for the *MT PSK Feedforward Equalizer.gvi*. Wire up the "input complex waveform" and "PSK system parameters" inputs. Wire the "equalizer length" input to the appropriate control. Wire the "training bits" input to the training bits array provided in the template. Wire "feedforward equalizer (LMS) parameters" to the cluster provided in the template. The "reset" input can be left unwired, since it will default to the correct value of "true." Wire all of the outputs to the appropriate indicators and you are ready to go.

3. Save your equalizer in a file whose name includes the letters "Equalizer" and your initials (e.g. *Equalizer_BAB.gvi*).

4. Open the block diagram of your BPSK receiver. Wire up the remaining inputs of *FrameSync(real)*; that is, connect the "Input Signal" input to the "Aligned Signal" output of *PulseAlign(real)* and connect the "Receiver Sampling Factor" input to an appropriate location. The "Sampled Input" *FrameSync(real)* input should remain connected to the output of *Decimate*.

5. Add your equalizer to your BPSK receiver. Wire the inputs as described in Step 2 above. During the lab you will be asked to take data to compare the eye diagrams before and after equalization. Wire the "Equalized Complex Waveform" equalizer output to a second eye diagram display. Also, wire the "Squared Error" output to a waveform graph indicator so that you will be able to observe the error decrease as the equalizer adapts. Wire the "Output Bits" and "equalizer coefficients out" outputs to indicators.

Questions

1. The default impulse response provided by *Channel.gvi* when the propagation delay is set at $100\ \mu s$ is $h[n]=\delta[n]+0.2\delta[n-1]-0.08\delta[n-2]$. Taking the z-transform gives channel system function $H(z)=1+0.2z^{-1}-0.08z^{-2}$. An ideal equalizer will have a system function that is the reciprocal of the channel system function. That is,

$$H_{eq}(z)=\frac{1}{1+0.2z^{-1}-0.08z^{-2}}\ . \tag{11}$$

   Find an expression for the ideal equalizer impulse response $h_{eq}[n]$. The coefficients of $h_{eq}[n]$ are the tap gains of an ideal (i.e. infinite length) equalizer. Find numerical values for the first five tap gains $c_0,\ldots,c_4$.

2. You should find that the ideal equalizer in Question 1 has a causal impulse response. The equalizer of Figure 3 can be non causal, if necessary. As a non-causal example, repeat Question 1 for $h[n]=0.1\delta[n]+\delta[n-1]+0.3\delta[n-2]$. Find numerical values for $c_{-4},\ldots,c_4$. (Hint: The ideal equalizer is an IIR filter. While it need not be causal, it must be stable.)

11.4 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: 915.0 MHz

   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: Root Raised

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

3. Ensure that the receiver is set up to use

   Carrier Frequency: 915.0 MHz

   IQ Rate: 200 kHz. Note: This sets the value of $1/T_z$. Note that T_z is the same parameter as dt .

   Gain: 0 dB

   Active Antenna: RX2

   Symbol rate: 10,000 symbols/s

   Message Length: 1000 bits

   Pulse shaping filter: Root Raised

   Use Channel: off

4. Run the transmitter, then run the receiver. Once the receiver has acquired its data, you may stop the transmitter. The receiver should show a BER of 0.0 or 1.0. Do not be concerned about a BER of 1.0 in this lab project.

5. Set the channel model for a propagation delay of $100\ \mu s$ with the default channel model. Set Use Channel to "on." Set the "Equalizer Length" to 11. Run the transmitter and then the receiver. Once the receiver has acquired data, you may stop the transmitter. Compare the eye diagrams before and after equalization. Use cursors to measure V_w and V_b and calculate the eye opening for each case.

6. Repeat Step 5 for propagation delays of $50\ \mu s$ and $150\ \mu s$. Compare the eye opening before and after equalization for each case. Prepare a table showing eye opening before and after equalization for each of the three propagation delays.

7. Set the propagation delay of the channel back to $100\ \mu s$ and run the transmitter and receiver. Observe the "Squared Error" equalizer output. Approximately what value does the squared error reach in steady state?

8. At a propagation delay of $100\ \mu s$ and an equalizer length of 11, record the values of tap gains $c_{-4},\ldots,c_4$. Compare with the values you computed in Prelab Question 1. (Do not expect extremely close agreement. The prelab calculations were for an infinite-length equalizer. Further, the adaptive equalizer tap gains are constantly being adjusted around their optimal values. Try running the receiver a few times to see how much variation there is in the steady-state tap gains.)

9. Change the ISI channel model to the model given in Prelab Question 2. Keep the propagation delay at $100\ \mu s$ and the equalizer length at 11. Run the transmitter and receiver. Check the eye diagrams to verify that the equalizer is working. Record the values of tap gains $c_{-4},\ldots,c_4$. Compare with the values you computed in Prelab Question 2.

10. With the propagation delay set at $100\ \mu s$ and using either of the two channel models, measure the eye opening after equalization as the equalizer length is increased in odd-numbered steps from 1. At what equalizer length does the eye opening stabilize?

11. Phase Synchronization Revisited. You may recall from the BPSK lab project that any phase difference ϕ between the transmitter and receiver oscillators will produce a factor $\cos(\phi)$ in the demodulated signal. This factor affects the amplitude of the demodulated signal, and can also affect its polarity. It turns out that the equalizer can remove this phase error just as if it were a channel impairment. To see this happen, make the following changes to your receiver:

    a. The *MT PSK Feedforward Equalizer* includes a threshold comparator and symbol mapper. The output bits that the equalizer produces are available at the Output Bits output of *Equalizer.gvi*. Your receiver should have an *Array Subset function* following the threshold to limit the received bit sequence to the proper frame length. Replace the wire running from the threshold comparator to *Array Subset function* with a wire from the equalizer Output Bits.

    b. The output bit sequence from the equalizer includes the 26-bit frame header. To remove this header, set the index input to *Array Subset function* to 26 instead of 0.

    Run the transmitter and the receiver. You should get a BER of 0. Try running the receiver with Use Channel both off and on. You should get a BER of 0 every time, showing that the phase ambiguity caused by the phase correction algorithm has been removed.

    c. Now remove the phase synchronizer entirely. Figure 4 shows an expedient way to do this.

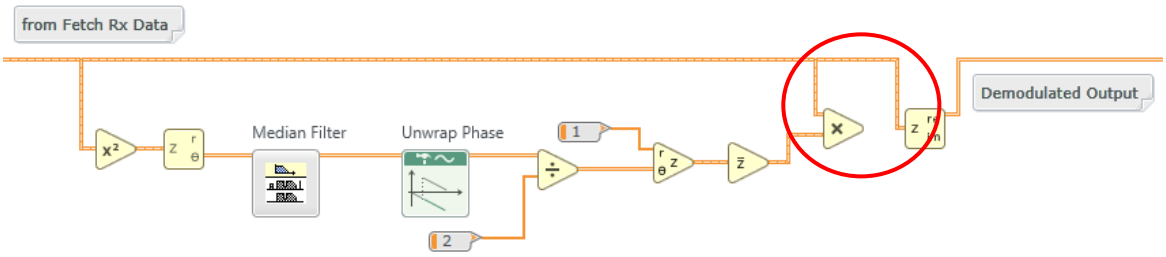

Figure 4. Remove the Phase Synchronizer

Run the transmitter and receiver again. Use Channel can be off or on. The BER should be zero every time.

Questions

1. Answer the questions from Lab Procedure Steps 6 through 10.

11.5 Report

Prelab

Hand in documentation for your equalizer program and your modified receiver that includes the equalizer and a second eye diagram. Also include documentation for any functions you may have created. To obtain documentation, print out legible screenshots of the front panel and block diagram.

Answer all of the questions in the Prelab section marked *Questions*.

Lab

Submit your equalizer program and your revised receiver program. 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 the questions in the Lab Procedure Steps 6 through 10. Comment on your success in using the equalizer as a phase synchronizer in Step 11.