Intersymbol Interference (ISI) and the Eye Diagram[*]

Ed Doering

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License [†]

	This module refers to LabVIEW, a software development environment that features a graphical programming language. Please see the LabVIEW QuickStart Guide[1] module for tutorials and documentation that will help you:
	• Apply LabVIEW to Audio Signal Processing
	• Get started with LabVIEW
	• Obtain a fully-functional evaluation edition of LabVIEW

Table 1

NOTE: Visit LabVIEW Setup[2] to learn how to adjust your own LabVIEW environment to match the settings used by the LabVIEW screencast video(s) in this module. Click the "Fullscreen" button at the lower right corner of the video player if the video does not fit properly within your browser window.

1 Summary

This project studies intersymbol interference (ISI) in an intuitive way by using a LabVIEW VI to simulate a pulse transmitter, finite bandwidth channel, and received signaling waveform. Rectangular pulses are considered first to demonstrate the ISI problem, and then two alternative pulse shapes are explored as a way to minimize ISI. The eye diagram is also introduced in this project as a visual aid to present the time-domain signaling waveform to promote understanding of the ISI phenomenon.

2 Objectives

1. Understand the root cause of intersymbol interference (ISI)
2. Explain the significance of the sinc pulse and raised cosine pulse as a means to eliminate ISI
3. Understand the construction of an eye diagram
4. Be able to measure performance metrics (peak ISI, noise margin, jitter, and timing sensitivity) directly from the eye diagram

---

[*]Version 1.1: Nov 29, 2008 2:47 pm -0600
[†]http://creativecommons.org/licenses/by/2.0/
[1]"NI LabVIEW Getting Started FAQ" <http://cnx.org/content/m15428/latest/>
[2]"LabVIEW Setup for "Communication Systems Projects with LabVIEW"" <http://cnx.org/content/m17319/latest/>

3 Deliverables

1. Summary write-up of your results
2. Any plots or diagrams requested

NOTE: You can easily export LabVIEW front-panel waveform plots directly to your report. Right-click on the waveform indicator and choose "Export Simplified Image."

4 Setup

1. LabVIEW 8.5 or later version

5 Textbook Linkages

Refer to the following textbooks for additional background on the project activities of this module; see the "References" section below for publication details:

- Carlson, Crilly, and Rutledge – Ch 11
- Couch – Ch 3
- Haykin – Ch 4
- Haykin and Moher – Ch 6
- Proakis and Salehi (FCS) – Ch 9
- Proakis and Salehi (CSE) – Ch 8
- Stern and Mahmoud – Ch 4

6 Prerequisite Modules

If you are relatively new to LabVIEW, consider taking the course LabVIEW Techniques for Audio Signal Processing[3] which provides the foundation you need to complete this project activity, including: block diagram editing techniques, essential programming structures, subVIs, arrays, and audio.

7 Introduction

Introductory digital logic courses present digital waveforms as essentially rectangular pulses. Indeed, the internal signals of a digital integrated circuit ideally exist at one of two voltage levels (high and low), with minimal time spent changing from one state to the other. Waveform displays from digital circuit simulators further emphasize the two-level rectangular shape of ideal digital signals.

Rectangular pulses are not ideal for transmission through communication links, however, since communication channels always restrict the bandwidth available between the transmitter and the receiver. Rectangular signaling pulses contain significant spectral energy across a wide frequency range due to the step-like transition between levels, and yet most communication systems do not allocate nearly enough bandwidth to faithfully transmit these abrupt changes. Passing a rectangular pulse through a limited-bandwidth channel distorts the pulse by "smearing" it – that is, the pulse stretches out in time. The transmitter sends a series of pulses to convey the message, therefore this time smearing causes **interference** between adjacent time slots (or **bit slots**). This **intersymbol interference** (abbreviated **ISI**) adds extraneous signal energy at the exact moments when a receiver's bit sampler decides whether a received bit should be called a logic "1" or a logic "0." ISI is not the same as additive random noise, but plays a similar role by reducing the **noise margin**, i.e., the room for error before the receiver's bit sampler makes an error.

---

[3] *Musical Signal Processing with LabVIEW – Programming Techniques for Audio Signal Processing* <http://cnx.org/content/col10440/latest/>

This project studies intersymbol interference in an intuitive way by using a LabVIEW VI to simulate a pulse transmitter, finite bandwidth channel, and received signaling waveform. Rectangular pulses are considered first to demonstrate the ISI problem, and then two alternative pulse shapes are explored as a way to minimize ISI.

The **eye diagram** is also introduced in this project as a visual aid to present the time-domain signaling waveform to promote understanding of the ISI phenomenon. The eye diagram also reveals other key performance measures such as **noise margin**, **timing jitter**, and **timing sensitivity**.

8 ISI_and_EyeDiagram.vi

Download the LabVIEW VI ISI_and_EyeDiagram.vi[4] , an interactive tool to study various pulse shapes as they pass through a band-limited channel.

Open the VI which starts running automatically, and then view the Figure 1 screencast video for a short orientation tour of the VI.

Figure 1: [video] Orientation tour of the "ISI_and_EyeDiagram.vi" LabVIEW VI

9 Rectangle Pulse Shape

Restore the front panel controls of "ISI_and_EyeDiagram.vi" to their default values by selecting "Edit | Reinitialize Values to Defaults."

Set the `symbols` control to 1 to produce a single rectangular pulse. The channel bandwidth should already be set to its maximum value of 0.49, which corresponds to essentially unlimited bandwidth. Note that this VI uses **normalized frequency**, therefore the sampling frequency corresponds to 1 and the Nyquist frequency is 0.5.

Compare the "transmitted waveform" and the "received waveform" plots in the lower-right front panel. How well does the received pulse match the transmitted pulse? Also, to what extent does the received pulse "spill out" of its designated time slot?

Decrease the channel bandwidth until you begin to observe noticeable pulse shape distortion. At what bandwidth does this occur?

Continue decreasing the channel bandwidth. What effects do you begin to observe?

Make a series of plots that show the progressive degradation of the rectangular pulse shape as the channel bandwidth is restricted. Right-click on the plot and choose "Export Simplified Image" to copy the graph to the clipboard for pasting into your report. Be sure to indicate the channel bandwidth for each plot.

10 Sinc Pulse Shape

Restore the front panel controls of "ISI_and_EyeDiagram.vi" to their default values.

Set the `symbols` control to 1 to produce a single pulse, and set the `bandwidth` control to 0.02. The received pulse should show noticeable distortion.

---

[4]http://cnx.org/content/m18662/latest/ISI_and_EyeDiagram.vi

Now set the `pulse shape` control to "Sinc." How much distortion is evident at the receiver? How much lower can you restrict the bandwidth while still preserving the basic sinc waveform shape?

The sinc function's ability to maintain its basic shape through a restricted channel bandwidth is important, but its true significance extends beyond this fact, as explored in the next section.

11 Multiple Pulses

A transmitter converts a message, or sequence of bits, into a series of analog pulses to create the signaling waveform. A receiver recovers the bitstream by periodically sampling the signaling waveform and comparing the sample to a threshold value to decide "1" or "0." Sinc-shaped pulse do not interfere with adjacent bit slots, **provided** that the bit slots are sampled at the correct instant in time.

To see this, reinitialize the front panel control values to their default settings, choose the "Sinc" pulse shape, and choose 2 symbols. Look carefully at the transmitted and received pulses on the lower-left front panel plots. The white trace shows the first pulse in the sequence, while the red trace shows the second pulse in the sequence. The first pulse has an amplitude of $+1$, while the second pulse has an amplitude of -1, corresponding to a bit sequence "10"; refer to the `message bitstream` indicator to confirm that the first bit is `T` (green LED indicator active) and the second bit is `F` (inactive LED indicator).

The waveform plots on the lower-right front panel show the actual transmitted and received waveforms, which superimpose (i.e., add) the individual pulses together. The plots on the lower-left front panel illustrate the contribution of each individual pulse.

Look carefully at the time 450 samples, in which the second (red) pulse is at its most negative value. What is the value of the first (white) pulse at this instant? Hopefully you can see that it is **zero**, indicating that the first (white) pulse produces **zero interference** at the instant the second (red) pulse attains its maximum absolute value.

Now, increase the number of symbols to 3, and also to higher values. Study the waveforms to convince yourself that even though a single sinc pulse extends over many bit time intervals, the contribution of all adjacent pulses is always zero at the moment that a given sinc pulse attains its maximum absolute value. Therefore, the sinc pulse shape achieves zero ISI when properly sampled.

Make a series of plots from the "received pulses" waveform display and explain your understanding of the sinc pulse shape and its ability to achieve zero ISI.

Restrict the channel bandwidth to 0.02, and confirm that the sinc pulses remain essentially unchanged. Now, set the pulse shape to "Rectangular." Set the `symbols` control to 2 and study the lower-left front panel plots. Identify where the second (red) pulse attains its maximum absolute value. How much interference is present from the first (white) pulse?

Make a series of plots from the "received pulses" waveform display and explain your understanding of the rectangle pulse shape and its susceptibility to intersymbol interference.

12 Eye Diagram

Study the transmitted and received waveform plots on the lower-right front panel as you increase the number of symbols to 40, and try this for the two pulse shapes considered so far. Also try varying the channel bandwidth. The received signaling waveform is reasonably easy to understand for rectangle pulses, but is rather difficult to interpret when sinc pulses are used. For example, you should find that you can easily correlate the "message bitstream" sequence with the high and low regions in the received waveform when rectangular pulse shapes are transmitted; the correlation is more difficult when sinc pulses are used.

The **eye diagram** provides a powerful visualization tool to interpret the behavior of the received waveform regardless of the pulse shape. The Figure 2 screencast video continues the discussion by explaining the eye diagram plot on the upper-right front panel of "ISI_and_EyeDiagram." Follow along with video, matching the front panel controls of "ISI_and_EyeDiagram.vi" to those of the video.

Image not finished

Figure 2: [video] Explanation of the eye diagram plot in the "ISI_and_EyeDiagram.vi" LabVIEW VI

The eye diagram reveals important performance metrics for a communication link, including **noise margin**, **ISI**, **timing sensitivity**, and **zero-crossing jitter**. In addition, the eye diagram shows the optimum sampling time for the receiver to regenerate a bitstream from the received signaling waveform. View the Figure 3 screencast video to learn how to measure these performance metrics and how to determine the optimum sampling time.

Image not finished

Figure 3: [video] Measuring noise margin, ISI, timing sensitivity, zero-crossing jitter, and optimum sampling time using an eye diagram

---

13 Eye Diagram Measurements

Figure 4 illustrates a generic eye pattern superimposed on a measured eye diagram plot and summarizes the definition of the various performance metrics discussed earlier. Use these definitions for the following measurements.
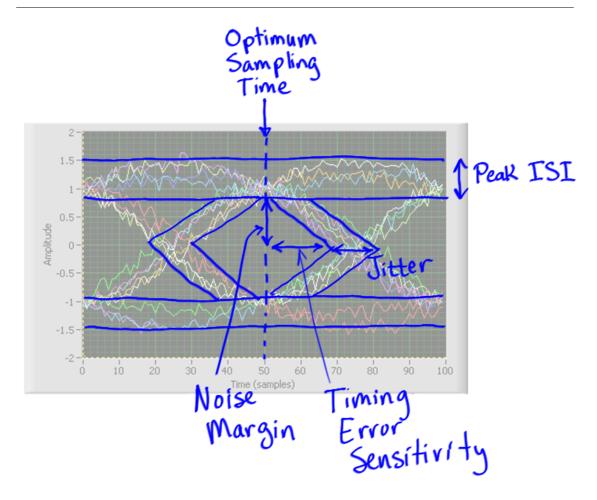
Figure 4: Generic eye pattern and definition of performance metrics

13.1 Rectangle Pulse

Restore the front panel controls of "ISI_and_EyeDiagram.vi" to their default values, and set the `symbols` control to 40. Vary the channel bandwidth and observe its effect on the eye diagram plot, and then set the channel bandwidth to 0.05. Increase the eye diagram `start time` to 245 samples to center the eye in the plot window.

Export the eye diagram plot to a piece of paper, and then use the eye diagram cursor as a tool to measure the following (show and label the relevant distances you measured on your hardcopy plot):

1. Optimum sampling time; report this as the number of samples from the nearest zero crossing
2. Peak ISI
3. Zero crossing jitter; report this as the maximum variation in time samples
4. Noise margin

13.2 Sinc Pulse

Ensure that the front panel controls of "ISI_and_EyeDiagram.vi" are the same as in the previous step, and then select the "Sinc" pulse shape. Adjust the eye diagram `start time` and `time span` to maximize the number of displayed bit intervals and also to avoid the initial startup transient that causes lines to cross through the center of the eye; also make adjustments to place the maximum eye opening at the center of the plot window.

As in the previous step, export the eye diagram plot to a piece of paper, and then use the eye diagram cursor as a tool to measure the following (show and label the relevant distances you measured on your hardcopy plot):

1. Peak ISI
2. ISI at the optimum sampling time
3. Zero crossing jitter; report this as the maximum variation in time samples
4. Noise margin
5. Timing error sensitivity; report this in terms of time samples

13.3 Raised Cosine Pulse

Keep the front panel controls of "ISI_and_EyeDiagram.vi" at the same settings you used for the previous "Sinc" pulse measurements, and then select the "Raised Cosine" pulse shape. You should expect to see the maximum eye opening remain centered in the eye diagram plot.

As in the previous steps, export the eye diagram plot to a piece of paper, and then use the eye diagram cursor as a tool to measure the the same five metrics as for the "Sinc" pulse. Show and label the relevant distances you measured on your hardcopy plot.

Compare your results for the raised cosine pulse and the sinc pulse. What appears to be advantageous about the raised cosine pulse shape?

See the video screencast in pam_RaisedCosinePulse.vi[5] for more background about the raised cosine pulse, the most widely-used pulse shape in digital communication systems.

14 Noise

Add random channel noise to the received waveform by moving the `noise standard deviation` control away from zero. Note how the eye pattern begins to close as the noise level increases.

Report the noise standard deviation value at which the eye just begins to close completely for each of the three pulse shapes. Make hardcopy plots of the eye diagram for each value that you report.

15 Channel Filter

"ISI_and_EyeDiagram.vi" uses a 10th-order IIR lowpass filter to model the limited-bandwidth channel. This type of filter is fairly realistic, and produces **delay distortion**, an effect caused by the filter's nonlinear phase response. Delay distortion causes the various signal frequency components to arrive at the receiver at different times, and can severely distort the originally-transmitted pulse shape.

Engage the `FIR` pushbutton control to select a linear-phase FIR filter that does not introduce delay distortion.

16 Final Comments

The `seed` front-panel control in the "Pulses" section primes the random number generator that creates the message bitstream. Feel free to try other seed values to produce other bit sequences for the message.

---

[5]"pam_RaisedCosinePulse.vi" <http://cnx.org/content/m18566/latest/>

"ISI_and_EyeDiagram.vi" uses two programming techniques that you may wish to investigate further, namely, an **event structure** and **property nodes**. Type Ctrl+E to show the block diagram window, and observe the event structure within the while-loop structure. The event structure only "fires" when a front-panel control value changes; the CPU does not do any work except to instantly respond to front-panel activity. The event structure makes the VI very responsive to user input, but does not burden the CPU as would a plain while-loop.

The property nodes allow front panel controls and indicators to be programmatically controlled. For example, changing the value of the `samples` front-panel control causes the upper limit of `start time` to automatically change to the same value.

17 References

1. Carlson, A. Bruce, Paul B. Crilly, and Janet C. Rutledge, "Communication Systems," 4th ed., McGraw-Hill, 2002. ISBN-13: 978-0-07-011127-1
2. Couch, Leon W. II, "Digital and Analog Communication Systems," 7th ed., Pearson Prentice Hall, 2007. ISBN-10: 0-13-142492-0
3. Haykin, Simon. "Communication Systems," 4th ed., Wiley, 2001. ISBN-10: 0-471-17869-1
4. Haykin, Simon, and Michael Moher, "Introduction to Analog and Digital Communication Systems," 2nd ed., Wiley, 2007. ISBN-13: 978-0-471-43222-7
5. Proakis, John G., and Masoud Salehi, "Fundamentals of Communication Systems," Pearson Prentice Hall, 2005. ISBN-10: 0-13-147135-X
6. Proakis, John G., and Masoud Salehi, "Communication Systems Engineering," 2nd ed., Pearson Prentice Hall, 2002. ISBN-10: 0-13-061793-8
7. Stern, Harold P.E., and Samy A. Mahmoud, "Communication Systems," Pearson Prentice Hall, 2004. ISBN-10: 0-13-040268-0