**EE 445S Real-Time Digital Signal Processing Laboratory – Prof. Brian L. Evans**

**Lab 5 Instructions Part 1**

**1. winDSK demonstration**

Connect your DSP board with the PC through the serial cable. Run winDSK8 on the PC, and then follow the instruction on page 262-267 in the real-time DSP book to run the “CommDSK” application. Check the result for rectangular pulse shaped BPSK and raised cosine pulse shaped BPSK.

**2. Simulation of block diagram (Matlab)**

* This task is intended to give you a feel for the system’s behavior prior to implementation on the DSP board. Complete the steps below with Matlab. Please note that we do not use the Matlab code from the textbook; you should write your own code based on the following requirements.

*Symbol generation*

* + Using the commands *round* and *rand,* generate 200 random bits.
  + These 200 bits will be directly used as the input symbols.

*Raised cosine filtering*

* + Now, we will be designing the baseband pulse shaping filter. The purpose of this filter is two – fold: Interpolation and elimination of Inter–symbol Interference (ISI). Usually, this filter is split between the transmitter and the receiver as a cascade of two SQRT of raised cosine filters. Instead, we will lump both of these filters into one raised cosine filter (called normal raised cosine filter).
  + You will use the commands *rcosflt* and *rcosine*. *rcosine* is used to obtain the numerator and denominator filter coefficients. *rcosflt* is used for the filtering operation itself. Please design a filter with the following specifications:
    - Input symbol rate: 2400 Hz (fsym, Fd in Matlab)
    - Output sample rate: 48 kHz (fs, Fs in Matlab)
    - Type of filter: fir/normal
    - Alpha (the excess bandwidth factor): 1 (, r in Matlab)
    - Truncation limit: +/- 4 symbol periods (delay in Matlab)
  + Since the filter is also used for interpolation, the ratio of the output rate (fs) to the input rate (fsym) gives an idea of the number of samples per symbol. In this case, there are 48kHz / 2400 Hz = 20 samples per symbol. This means, there are 20 samples used to interpolate the space between any 2 symbols. Thus, the digital data is transmitted as “analog” data.
  + The truncation limit dictates the length (in symbol periods) of the raised cosine impulse response.
  + The alpha factor is explained in slide 11. Basically, an  filter resembles an ideal low pass filter (a sinc function in the time domain). As  increases towards 1, the impulse response effectively becomes shorter at the expense of increased bandwidth in the frequency domain.
  + Now, filter the symbols you generated using the above filter.

*Eye diagram*

* + We will look at the output as an “eye diagram”. This represents the transitions between adjacent symbols. It also gives an idea of the ISI that is present and how the filter may help get rid of ISI so that each level can be thresholded accurately at the symbol period intervals at the receiver to recover the original data.
  + We will use the command *eyediagram* in Matlab to plot the eye diagram of the data. We will be plotting 2 symbol periods at a time which means 20\*2 = 40 samples at a time. Once 40 samples are plotted, the plot wraps around and starts plotting over the trace already present. This way, we get an idea as to how clean the transitions are between levels and how easy it is to threshold the data at the symbol period intervals. The optional “period” parameter is used to set the X axis and must be set to 2 symbol periods.
  + **Lab Report:** Output of the eye diagram.

**3. Simulation of BPSK transmitter (Matlab) (Contd.)**

* Repeat Task 2 for  = 0.125.
* **Lab Report:** Output of the eye diagram.

**4. Simulation of block diagram (Matlab) (Contd.)**

*Magnitude response*

* Obtain the filter coefficients of the filters you designed in Task 2 and Task 3 using the *rcosine* command. Please use *freqz* and *plot* to plot the magnitude response of both these filters.
* **Lab Report:** Plots of magnitude responses.

*Impulse response*

* Plot the impulse response of both pulse shaping filters.
* **Lab Report:** Plots of impulse responses.

*Understanding the plots*

* **Lab Report:**

Explain the major differences between the two filters with respect to their

A) Magnitude responses

B) Impulse responses

* **Lab Report:**

1. What is the width of the impulse response for the =0.125 case?
2. How would you obtain this number theoretically? (Hint: Look at the fsym and truncation limit you set).

**5. DSP implementation (Code Composer)**

* Create a new project with the source files in C:/CD/code/chapter\_16/ccs/DigTxIM
* Run this program and Check the result using oscilloscope.
* Understand the codes in the file “impulseModulatedBPSK\_ISRs.c” based on the explanation in page 274-276 of the textbook. Especially understand how the pulse shaping filter is implemented by filter banks.
* Show the screenshot in your lab report.

Now we will modify the code a little bit:

* In the original code, the symbols are generated using the *rand()* function. Now try to generate the ‘random’ symbols using the scrambler that you created in Lab 4. This time, we will be using the polynomial **1 + D18 + D23**. The **initial state** of the scrambler is **5**.
* Design your own raised cosine filters follow the specifications given in task 3 and 4. You need to use the function “rcosine” in Matlab. Export the coefficients into the source file “coeff.c” in the CCS project. **Note that the length of the filter is different from what we have in the original code. Please modify the code to make it compatible with the new filters.**
* Edit the line (“output = y\*cosine[counter & 3]”). We don’t need to multiply by the carrier since we just want to see the result of the baseband pulse shaping filter.
* Run the project.
* **Lab Report:** Show screenshots of the time-domain BPSK (2-PAM) signals for the two values of alpha. Note that pressing the *single* button on the oscilloscope may make it easier to visualize the signals since they change rapidly with time.