FINAL REPORT

18-758 Wireless Communications
Group Members: Mukundh Balajee, Vikram Marmer

Basic Design

For the transmitter, we implemented a 16-QAM modulation scheme, to transmit an image over a USRP channel, receive the same image and compare BER. The transmitted signal, $x^{base}(t)$, is padded with a preamble comprising 1600 frequency synchronization bits, 200 pseudorandom timing synchronization bits, 200 pseudorandom frame synchronization bits, and 200 pseudorandom pilot sequence bits. To ensure a BER of less than 1 percent for the message of 20520 information bits, we split the message into 20 segments. The final signal consists of the preamble followed by 20 segments of data with a pilot signal before each one aside from the data segment directly after the preamble. The pilot sequences were used to update the equalizer to be optimal for each frame. Once the message is assembled in the form of bits, the signal is translated into 16-QAM, and QAM symbols are upsampled and then convolved with the SRRCR pulse. To keep the signal within the required bandwidth, an alpha value of 0.3 was used and the filter length was set to 30 samples per symbol for 12 symbols. The signal is then transmitted through a USRP radio server, and we receive a receivedsignal.mat file from the remote server. Our transmitted signal has a symbol rate of 16.7M symbols/s.

At the receiver, this signal, $y^{base}(t)$, is convolved with the matched filter, followed by timing synchronization and frame synchronization processes. The timing sync finds where the preamble starts while the frame sync process finds the beginning of each pilot sequence before the 20 data frames. Once the beginning of the preamble and each frame is known, sampling is done to change down from 200M samples per second. The sampled signal is then run through an MMSE-LE equalizer. This is a change from the mid-project review since we used a one-tap equalizer before. The MMSE-LE (with the LMS algorithm) uses a 6-tap setup and a mu value of 0.2. These values yielded the best results in terms of the lowest BER. The MMSE didn't perform well enough to compensate for the phase offset that worsened for each frame. This problem may have resulted from insufficient training at the beginning of the preamble before starting to equalize data bits. To add additional phase compensation, we offset the phase for frames by the average phase offset of the pilot signals before and after the data. This significantly and reliably lowered BER to under 1 percent during all our testing. The receiver then guesses constellation points from the samples and converts from 16-QAM back to bits. The final step is to simply check against the reference image for calculating BER.

Advanced Design

For the advanced design, we decided to implement CDMA for two users using gold codes. In the transmitter, the setup is similar to the basic design. We start with a shortened version of the preamble in bits since the spreading code will greatly increase the length of the signal. The goal was to transmit 100 bits for each user over CDMA. We split this message into 10 segments to allow the receiver to adapt its equalizer and keep BER below 10e-6. The new transmitted signal, $x^{base}(t)$, is padded with a preamble comprising 20 frequency synchronization bits, 20 pseudorandom timing synchronization bits, 20 pseudorandom frame synchronization bits, and 20 pseudorandom pilot sequence bits. We generate two gold codes by XOR'ing two carefully chosen m-sequences. With the use of Linear Shift Feedback Registers, we generate two different m-sequences of the same length, such that they produce gold codes with low cross-correlation. These m-sequences have the same phase offsets and are chosen such that they produce unique gold codes when XOR'ed with each other. The bits are shifted by 1 place for either one of the m-sequence, to ensure unique gold codes for each user.

Before we can multiply data by the gold codes, we first produce the 4-QAM symbols and spread the symbols by creating 32 duplicates of each symbol because our spreading gain spec was 32. Once the duplicates are created, the entire symbol vector is 32 times longer. Now that the data is spread, We multiply the vector by the gold codes that also have a length of 32. Therefore, each symbol has the same number of copies as the length of the gold code. User 1's data is multiplied by one gold code and user 2's data is multiplied by a second. Two different gold codes have low cross-correlation and have a spectrum similar to noise. Once the two user data streams are summed, this is upsampled, convolved with an SRRCR pulse, and transmitted. The pulse in this case is different from the basic design in that it now has 40 samples per symbol instead of 30 to keep the signal within the bandwidth limits.

In the receiver, we use the same pulse to first demodulate. We follow the same process of doing time synchronization, frame synchronization, and sampling. The only difference is the sequences that are searched for by the synchronization code. We also use the same MMSE-LE equalizer and additional phase compensation. We then run the signal through a rake receiver. We experimented with 2 and 3 finger receivers. The first finger had zero sample delay. The second finger is delayed by one sample, so the sample vector is shifted to the left by one. The third finger (when used) is delayed from the original by 2 samples. These samples are post-sampling, not from the "analog" signal sampled at 200 MHz. After the rake receiver, the spread samples (the 32 duplicates) are added together and then the sums are divided by 32 to finish despreading the signal. Once the despreading is done, the received symbols are put through the guessing code for 4-QAM. The symbols are converted back to bits and are checked for correctness with BER.

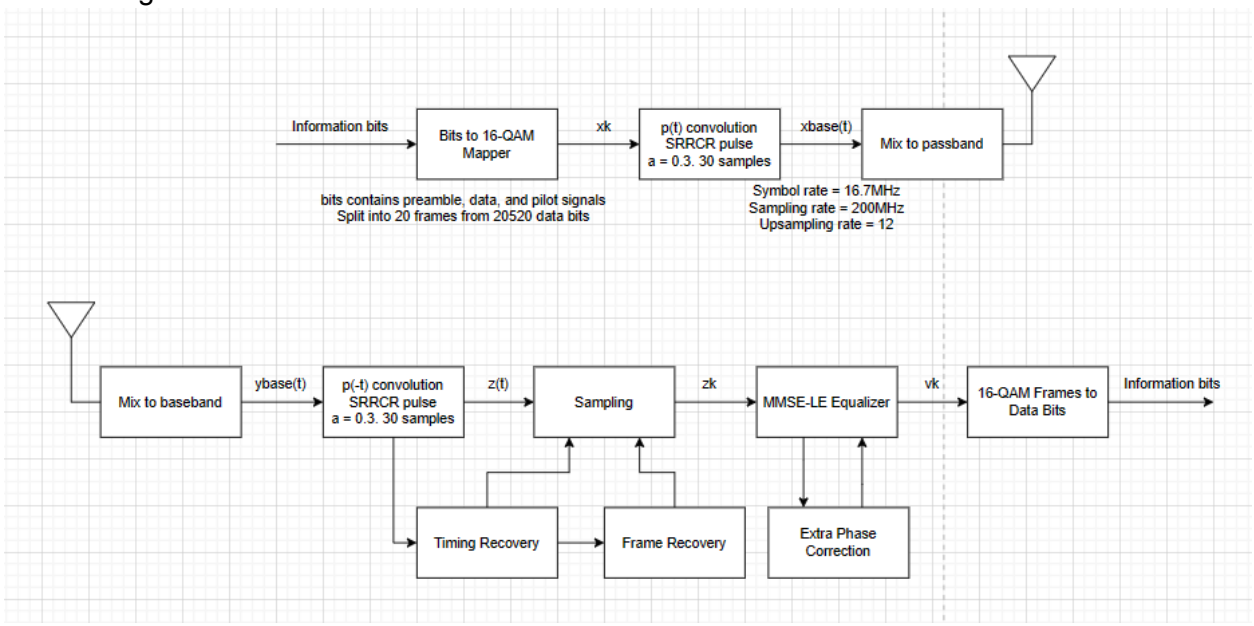
Differences from Class Handout

- **MMSE-LE:** While implementing MMSE-LE equalization, we had some challenges when accounting for the phase issues we had, as mentioned in our basic design. Implementing the LMS algorithm to train the filter was also a challenge considering our design choice.
- **Spreading/Despreading:** For generating the gold codes, we used the slides as a starting point, however, the despreading process proved to be challenging. The lecture slides recommended using a matched filter for the output of each finger, before combining the output of each finger, but we found it unnecessary. For the despreading process, we repeat the gold code to match the length of the signal, and despread the signal, and combine the outputs of each finger directly. To account for the spreading gain, we reshape our received signal for each user from each rake before we demodulate the signal.

Learnings/Takeaways

- **MATLAB:** We have used MATLAB in prior classes, however, it was our first time making an entire project based solely on MATLAB.
- **Trellis Coded Modulation:** Initially, we planned on implementing an 8-state Trellis Code, but ran into several problems and hence, pivoted to CDMA. Since we spent over a week trying to implement Trellis Code, we gained a good understanding on the Parity Check Matrix and Generator Matrix, and how to find these matrices and how they are related.
- **Modulation/Demodulation Techniques:** For the Basic Design, we implemented the modulation and demodulation ourselves, which helped in providing a better understanding of the process and steps taken and conversion from QAM/PSK Symbols to bits and vice versa.

Basic Design



Advanced Design