**ENEE 630**

**Project Description**

**Revision A.03**

**October 14, 2021**

**REVISION HITORY**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date of Issue** | **Scope** |
| A.01 | 10/11/2021 | Initial draft |
| A.02 | 10/12/2021 | Last Update |
| A.03 | 10/14/2021 | Last Update – Final Version for Part I (floating point RX) |
|  |  |  |

**TABLE OF CONTENTS**

**SECTION PAGE**

1.0 Introduction 1-1

1.1 Summary 1-1

1.2 Guildline, deadline for the First PART – (floating point Reciver) 1-1

1.3 Project Description 1-1

2.0 Part I : FLoating point simulations and benchmark 2-1

2.1 Signal desription 2-1

2.2 TRANSMITTER & Modulator 2-2

2.3 Channel 2-2

2.4 Reciever 2-3

2.5 A FEW notes and tips 2-4

2.6 Evaluation of the algorithm and Project Report 2-5

3.0 Reciver Fixed point Realizations and optimization 3-6

**LIST OF FIGURES**

**FIGURE PAGE**

Figure 1 Burst Format 2-2

Figure 2 BPSK modulator 2-2

Figure 3 Channel 2-2

Figure 4 Ambiguity of time/delay and frequency 2-3

Figure 5 Receiver block diagram 2-4

Figure 6 BPSK Ideal Receiver (Demodulator) 2-5

**LIST OF TABLES**

**TABLE PAGE**

Table 1 Floating point RX MIPS table 2-6

# Introduction

## Summary

The goal is to be able to design a complete TX/RX communication system in real time. We would like to optimize the usage of memory, processing power and cycles of the RX algorithm while keeping the quality of the system at the best possible level.

This document will be updated with respect to the lectures in the class. We cover each module in the class and go over the details.

You can use MATLAB or C or any other tools to do this project. We start with floating point simulations and then try to move to a fixed-point FPGA / DSP realization for the receiver.

We try to use the concepts of this course to improve the quality of the receiver and/or cut the processing time and memory usage.

## Guildline, deadline for the First PART – (floating point Reciver)

**The first part of the project, is due 11/8/2021. You need to deliver these items**

1. A report explaining your algorithm, your design and methods. You need to analyze any strong or weak points of your design and explain the theoretical reason behind it.
2. The code, in C or MATLAB with clear instruction on how to **compile and run it**

**(If there is any error or bug, it is regarded as no CODE and it is graded as such)**

1. The results including plots, figures, and tsbles according to section 2.5

Grading is based on the quality of our design and submitted material.

Please let me know if you need any help or guidance.

## Project Description

Design a receiver to demodulate a frame containing 664 bits of data, every 50 msec. Sampling rate is 16 KHz and the signal is 16 times over sampled. The throughput should not be less than 83%.

The baseband signal is modulated and passing through an AWGN channel with delay uncertainty of +/- 2.5 msec (5 msec total) and the frequency uncertainty of +/-1500 Hz.

# Part I : FLoating point simulations and benchmark

## Signal desription

Length = 50 msec

Sampling Frequency = 16000 Hz

Total Length in Bits = 800 bits

* 128 bits of CW – continuous waveform or a tone at 0 frequency – (All “1”)
* 8 bits: A Key for some future use (All “0” for now)
* 664 bits of random data (Throughput = 83%)

664 random bits

128 bits - CW

KEY

Figure Burst Format

## TRANSMITTER & Modulator

The modulator is Pi/4 BPSK. I suggest using the following protocol for convenient, but you can use yours. You need to follow the same protocol in the receiver side.

X(k)

Pi/4 BPSK

Data(k)

Figure BPSK modulator

This simple formula shows how to generate the modulated signal values. Note that the values are complex.

* Up-sample X(k) to 16 to generate X1(k)
* Pass X1(k) thorough a lowpass filter to cut any signal after and generate S(k)
  + Suggestion: Use RRC with . This filter covers +/- 3 symbols, with length = 97 taps.
  + S(k) is the base band signal

## Channel

R(k)

S(k)

Channel

Figure Channel

The channel is AWGN. It also adds time and frequency (and phase) uncertainty to the signal.

* is AWGN noise samples. Noise power defined by SNR values.
* is the index representing the random time delay or time uncertainty. The range of this uncertainty is . (You need to find representing this time delay)
  + Note: Use the *sampling frequenc*y and *up sampling* rate to change these values form msec to discrete time
* is random frequency uncertainty. The range of this uncertainty is < +/- 1500 Hz
* *Optional:* is a random phase ( )

## Reciever

The first step to design the receiver is to design a lowpass filter. After LPF, you need to down sample the signal from 16X to 1X:

* LPF filter

(Suggestion: use the same filter as the transmit filter)

* Down sample by 16
  + Suggestion: Find the correct “sampling time” using energy method – (as discussed in the class)

The second step is to use DFT method to resolve the time/delay and frequency. – (as discussed in class)

Here, we are trying to find ( and ) in the following square. (see Figure 4)

1500 Hz

2.5 msec Hz

-2.5 msec Hz

-1500 Hz

Figure Ambiguity of time/delay and frequency

The last step is to use the estimated time and frequency and recover the original data. If the DFT algorithm estimations are then the recovered signal will be:

The random data starts at .

Signal recovery

DFT algorithm

Matched Filter

Figure Receiver block diagram

## A FEW notes and tips

A few notes and helpful tips about testing your RX algorithm:

1. Before you start testing your algorithm, note that this algorithm is basically a BPSK demodulator and should works as such. So if your demodulator has perfect time and frequency offset, it should behave exactly as Figure 6 . i.e., ideal BPSK demodulator.
2. Note that, the frequency resolution is related to the size of DFT … or (in Hz?)
3. The time resolution is 1 symbol. (in msec?)
4. First, try to make the DFT algorithm work for the and = 0 under clean channel (or very high SNR). Then see if the algorithm works for the other point, the edges, and the extreme points.
5. If you set or or any fraction of the resolution, the algorithm fails or generate bad results.

(*Try to analysis it via theory and see what kind of solutions is needed to so this failure can be avoided! – Of course, you cannot change the design of the burst, TX or Channel – only your receiver*)

1. Note that, you do not need to search the whole DFT coefficient. (We will use this point to optimize the algorithm later).

Chart, line chart

Description automatically generated

Figure BPSK Ideal Receiver (Demodulator)

## Evaluation of the algorithm and Project Report

1. What is the mean and STD of the time/delay estimate or  *(in msec)* under these conditions?
   1. SNR = 100 dB / Clean channel
   2. Hz
   3. Time offset = -2.5 msec to 2.5 msec, step = 0.0625 msec
   4. Number of simulations = 100 per point
2. What is the (mean and STD) of frequency estimate or (in Hz) under these conditions
   1. SNR = 100 dB / Clean channel
   2. msec
   3. = -1500 Hz to 1500 Hz, step = 125 Hz
   4. Number of simulations = 100 per point
3. BER and FER curves (bit error rate (BER) and frame error rate (FER) ), Mean/STD of time estimate (msec) , Mean/STD of frequency estimate (Hz) . These curves are for SNR = -3 to 15 dB, step size = 0.5 dB
   1. = 0 Hz, , . Compare it with Ideal BPSK BER curve.
   2. = 600 Hz, Time offset = 2.5/10 msec ,
   3. = 62.5Hz, ,

Note: Continue with the simulation until both of the following conditions are satisfied:

* + 1. Number of simulations > 1000
    2. number of frame errors > 50

1. Fill out the following table for one burst (50 msec).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| COST OF RX Algorithm | Number of Additions | Number of Multiplications | Number of Sin() and Cos() functions | Number of divisions | Any non linear operation |
| LPF RX Filter |  |  |  |  |  |
| Down Sampling |  |  |  |  |  |
| DFT |  |  |  |  |  |
| Time & Frequency compensation |  |  |  |  |  |
| Total |  |  |  |  |  |

Table Floating point RX MIPS table

What is the rough estimate of memory used in the RX algorithm? Did you use any tables, or vectors?

# Reciver Fixed point Realizations and optimization

In this part of the project, we will use the transmitter and the channel as the previous section

Our main goal is to find a fixed point (FPGA ready) realization of the algorithm.

(Tentative he deadline is the week after Thanksgiving)

After midterm (10/18) we start to talk more about this part.

We try to use the concepts discussed in the class with some standard and simple routines to:

1. Optimize number of mathematical operations needed for the algorithm.
2. Optimize the memory needed for the algorithm
3. Maintain the benchmark performances (see section 2.5) same as it is with floating point realization