FM Stereo Radio Final Report

Group 6

Zijian Wang, Yijie Wei, Yang Liu

1. Basic FM Radio background/theory

Frequency modulation is a form of analog angle modulation in which the baseband information-carrying signal, typically called the message or information signal m(t),varies the frequency of a carrier wave. Audio signals transmitted by FM radio communications are the most common. However, FM radio can also transmit digital data with the low bandwidth digital information known as Radio Data System (RDS) in Europe and Radio Broadcast Data System (RBDS) in the U.S. The simplest approach to generating FM signals is to apply the message signal directly to a voltage-controlled oscillator (VCO) as shown in Figure.

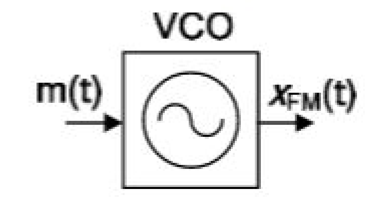


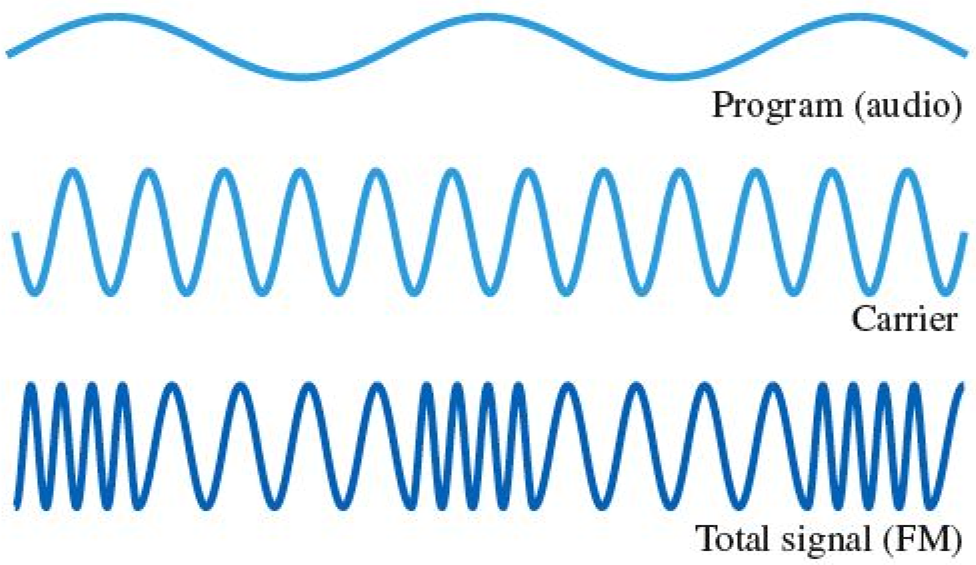
Figure 1. FM Generation with a VCO

A few observations can be made from the FM output signal. First, the amplitude of an FM signal is constant regardless of the message signal, giving it a constant envelope property with an output power equal to 2 2 Ac into a 1 Ω resistor. Second, the frequency-modulated output, xFM(t), has a nonlinear dependence to the message signal, m(t), making it difficult to analyze the properties of an FM signal. To estimate the bandwidth of an FM signal, a single tone message signal is used as shown below. where Am is the amplitude of the message signal and fm is the frequency of the message signal. Substituting this message signal into the above formulas, we find

$$x_{FM}(t) = A_c \cos\left(2\pi f_c t + \frac{K_{VCO} A_m}{f_m} \sin(2\pi f_m t)\right)$$
$$= A_c \cos\left(2\pi f_c t + \frac{\Delta f}{f_m} \sin(2\pi f_m t)\right)$$
$$x_{FM}(t) = A_c \cos(2\pi f_c t + \beta \sin(2\pi f_m t))$$

FM radio uses frequency modulation. The frequency of the signal in unchanged or un-modulated, so there's no useful information contained. But once information has been introduced to this signal, the combination results in a *change to the frequency*, which is directly proportional to the information. When the frequency is modulated between low and high, music or voice is being transmitted by the carrier frequency. But only the frequency changes as a result; the amplitude remains constant the entire time.

$$\begin{aligned} y(t) &= A_c \cos\left(2\pi \int_0^t f(\tau) d\tau\right) \\ &= A_c \cos\left(2\pi \int_0^t \left[f_c + f_\Delta x_m(\tau)\right] d\tau\right) \\ &= A_c \cos\left(2\pi f_c t + 2\pi f_\Delta \int_0^t x_m(\tau) d\tau\right) \end{aligned}$$

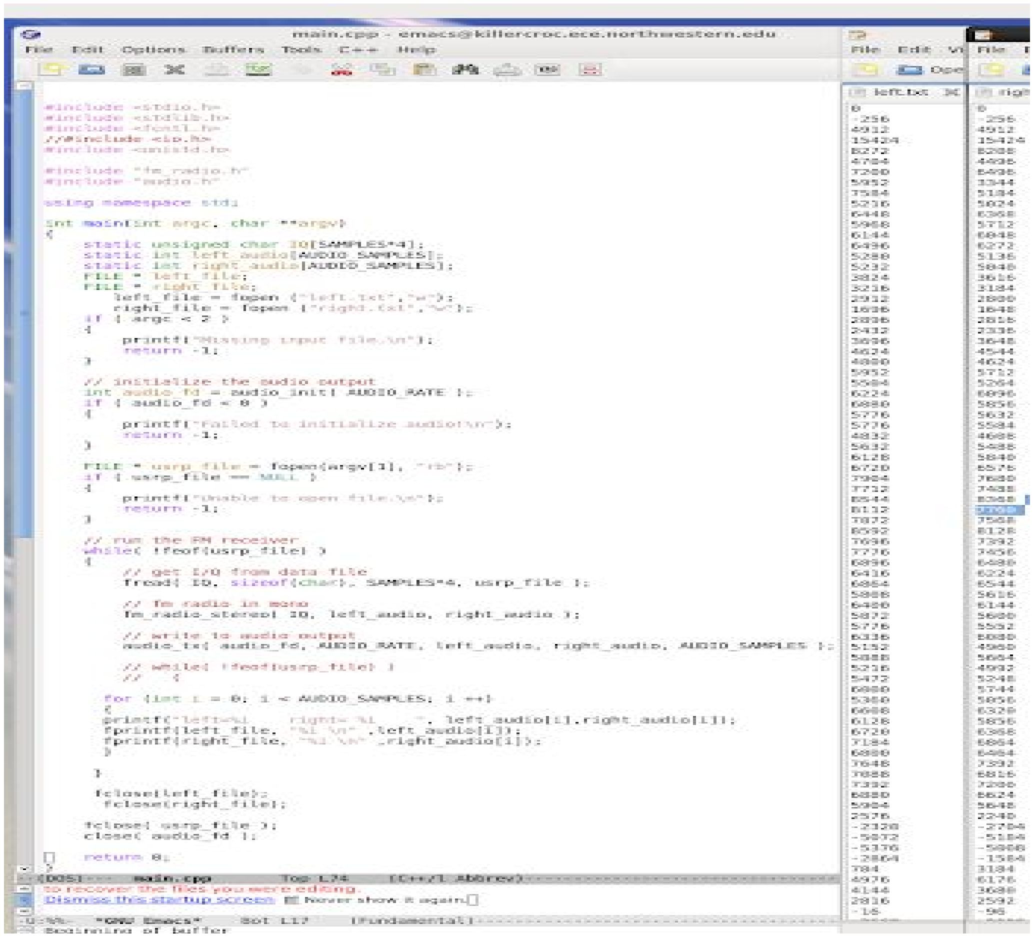


2.The system architecture

Components of read_IQ, Deemphasis_n, FIR, compute, demodulate_n are connected with structural in fm_radio_stereo. Two components FIFO and fm_radio_stereo are connected in fm_radio_top, two processes: input_fifo_to_signal_process, result_to_output_fifo_process . The input and output are processed with two processes in fm_radio_tb with FIFO in and FIFO out: input_file_to_input_fifo_process, result_to_output_fifo_process.

3.Design process

In the beginning, we used Linux to compile the CPP codes and successfully decode the USRP.dat file and got the decoded music played out, then we change the main file for generate the decimal number output for getting stander result for guiding our designing process.



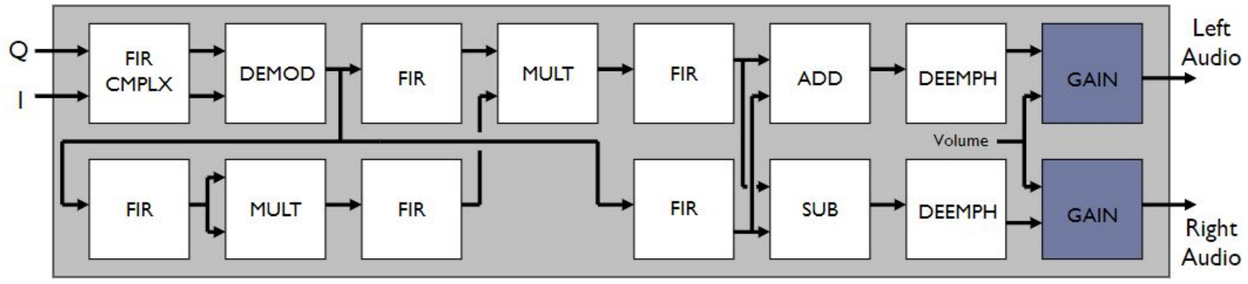
```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
//#include <io.h>
#include <unistd.h>

#include "fm_radio.h"
#include "audio.h"

using namespace std;

int main(int argc, char **argv)
{
    static unsigned char IQ[SAMPLES*4];
    static int left_audio[AUDIO_SAMPLES];
    static int right_audio[AUDIO_SAMPLES];
    FILE * left_file;
    FILE * right_file;
       left_file = fopen ("left.txt","w");
       right_file = fopen ("right.txt","w");
    if ( argc < 2 )
    {
        printf("Missing input file.\n");
        return -1;
    }

    // initialize the audio output
    int audio_fd = audio_init( AUDIO_RATE );
    if ( audio_fd < 0 )
    {
        printf("Failed to initialize audio!\n");
        return -1;
    }

    FILE * usrp_file = fopen(argv[1], "rb");
    if ( usrp_file == NULL )
    {
        printf("Unable to open file.\n");
        return -1;
    }

    // run the FM receiver
    while( !feof(usrp_file) )
    {
        // get I/Q from data file
        fread( IQ, sizeof(char), SAMPLES*4, usrp_file );

        // fm radio in mono
        fm_radio_stereo( IQ, left_audio, right_audio );

        // write to audio output
        audio_tx( audio_fd, AUDIO_RATE, left_audio, right_audio, AUDIO_SAMPLES );

        // while( !feof(usrp_file) )
        //     {

      for (int i = 0; i < AUDIO_SAMPLES; i ++)
      {
      printf("left=%i      right= %i      ", left_audio[i],right_audio[i]);
      fprintf(left_file, "%i \n" ,left_audio[i]);
      fprintf(right_file, "%i \n" ,right_audio[i]);
      }

     }

     fclose(left_file);
      fclose(right_file);

    fclose( usrp_file );
    close( audio_fd );

    return 0;
}
```

```
left.txt
0
-256
4912
15424
8272
4784
7280
5952
7584
5216
6448
5968
6144
6496
5280
5232
3824
3216
2912
1636
2836
2432
3696
4624
4880
5952
5584
6224
6880
5776
5776
4832
5632
6128
6720
7904
7712
8544
8112
7872
8592
7696
7776
6896
6416
6864
5888
6480
5872
5776
6336
5152
5888
5216
5472
6880
5360
6608
6128
6720
7184
6880
7648
7888
7392
6880
5904
2576
-2128
-5872
-5376
-2864
784
4976
4144
2816
-16
```

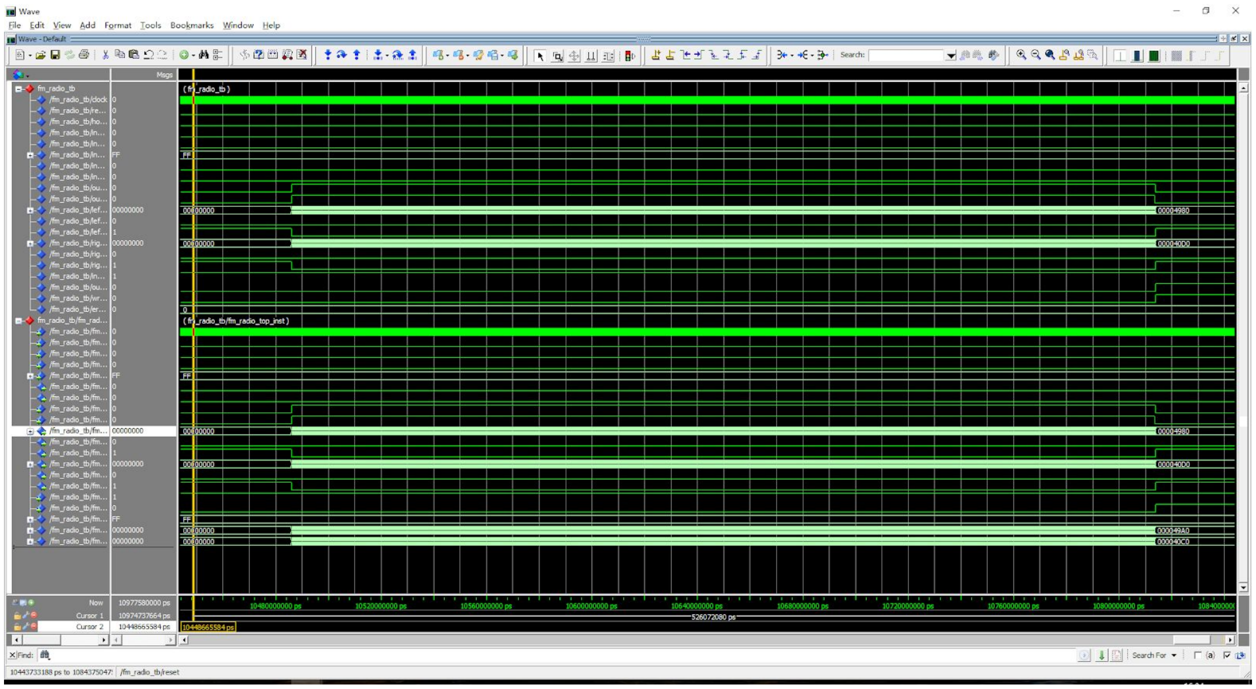
```
0
-256
4912
15424
8288
4496
6496
3344
5184
5824
6368
5712
6848
6272
5136
5848
3616
3184
2800
1648
2816
2336
3648
4544
4624
5712
5264
6896
5856
5632
5584
4688
5488
5840
6576
7680
7488
8368
[illegible]
7568
8128
7392
7456
6480
6224
6544
5616
6144
5600
5552
6080
4960
5664
4992
5248
5744
5856
6320
5856
6368
6864
6464
7392
6816
7200
6624
5648
2240
-2704
-5184
-5808
-1584
3184
6176
3680
2592
-96
```

Changed main.cpp and Left & Right channel data

After we got the original data, we separate the C code to each member in our team for translate to VHDL code.



We using one week to make sure all sub functions are working properly, then combined all sub functions together and write the sim.do file for generate the waveform output.

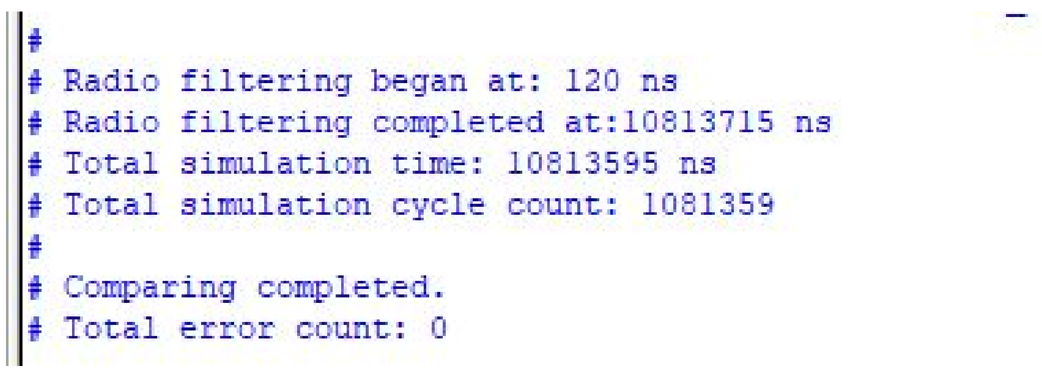


```
fm_radio_tb
  /fm_radio_tb/clock      0
  /fm_radio_tb/re...      0
  /fm_radio_tb/ho...      0
  /fm_radio_tb/in...      0
  /fm_radio_tb/in...      0
  /fm_radio_tb/in...      FF
  /fm_radio_tb/in...      0
  /fm_radio_tb/in...      0
  /fm_radio_tb/ou...      0
  /fm_radio_tb/ou...      0
  /fm_radio_tb/lef...     00000000
  /fm_radio_tb/lef...     0
  /fm_radio_tb/lef...     1
  /fm_radio_tb/rig...     00000000
  /fm_radio_tb/rig...     0
  /fm_radio_tb/rig...     1
  /fm_radio_tb/in...      1
  /fm_radio_tb/ou...      0
  /fm_radio_tb/wr...      0
  /fm_radio_tb/er...      0
fm_radio_tb/fm_rad...
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      FF
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      00000000
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      1
  /fm_radio_tb/fm...      00000000
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      1
  /fm_radio_tb/fm...      1
  /fm_radio_tb/fm...      0
  /fm_radio_tb/fm...      FF
  /fm_radio_tb/fm...      00000000
  /fm_radio_tb/fm...      00000000

Now       10977580000 ps
Cursor 1  10974737664 ps
Cursor 2  10448665584 ps

10443733188 ps to 1084375047!   /fm_radio_tb/reset
```

4.Optimizations and Impact on Design

We using Loop unroll technique in the Deemphasis module, unrolling the loop function to several lines of duplicate codes. After the simulation, we found the influence is limited, no significant difference.

5.Simulation and Performance Results



```
#
# Radio filtering began at: 120 ns
# Radio filtering completed at:10813715 ns
# Total simulation time: 10813595 ns
# Total simulation cycle count: 1081359
#
# Comparing completed.
# Total error count: 0
```

Radio filtering began at: 120 ns
Radio filtering completed at:10813715 ns
Total simulation time: 10813595 ns
Total simulation cycle count: 1081359
Total error count: 0

Data file size: 548KB
Left audio output: 320KB
Right audio output: 320KB
Time used: 10.8ms
Reading data rate:548KB/10.8ms = 50.74MB/s
Throughput: 640KB/10.8ms = 59.259MB/s
Throughput per Channel: 320KB/10.8ms= 29.62MB/s
Sampling rate : 8 bit* 100MHz = 0.8 Gbps