A Software Defined Radio on an FPGA

Colin Chaney and Charles Lindsay
Fall 2019 - 6.111 Project Proposal

Section 1: Introduction

For our final project, we will implement the digital end of a software defined radio (SDR) entirely on an FPGA. Our goal is to be able to demodulate all types of modulation schemes (AM, FM, FSK, BPSK, QPSK, etc.) and to play the encoded audio signal over a speaker. Along with the audio output, we want to allow users to display different things to an accompanying monitor. The monitor should be able to display the audio output waveform, the frequency spectrum of the received signal, and a waterfall display that shows the intensity of the frequency spectrum over the past values. If the technology is available we want to be able to hook up an analog front end and interpret real AM audio broadcast in it's entire spectrum.

Section 2: Goals

Absolute Bare Minimum Functions

- Demodulate AM audio and output to speaker
- Perform FFT on input spectrum and display full spectrum on VGA monitor
    - Show Intermediate Frequency with Bandwidth
- Display audio signal on VGA monitor

Desired Functions

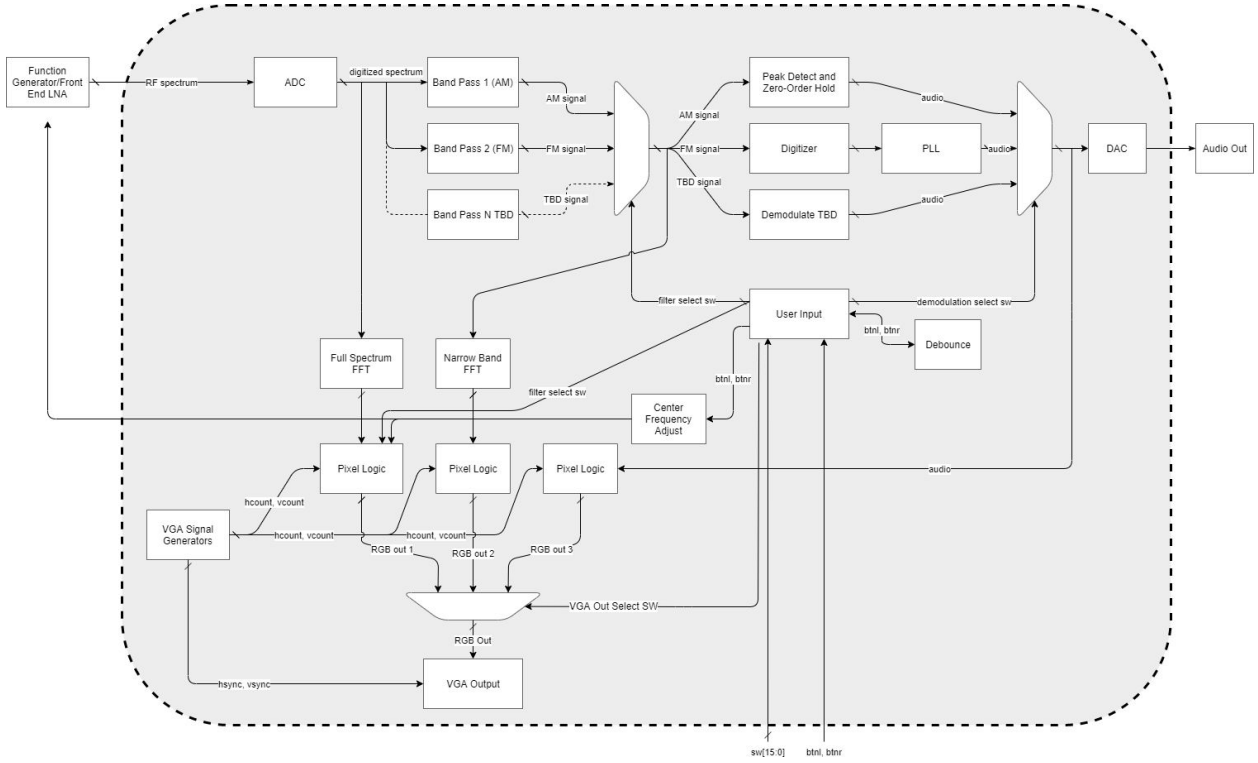
- Demodulate FM audio
- Display waterfall on VGA monitor

Stretch Goals

- Interface with analog front end to be able to detect and display entire AM spectrum
    - Control local oscillator to tune desired frequency corresponding to output intermediate frequency
- Demodulate FSK, BPSK and QPSK
- Apply sound effects to output audio (e.g. echo) played on speaker
- Create custom pots for center frequency adjustment

Section 3: Design and Implementation

Block Diagram



Modules

ADC (onboard OR front end ADC) (Charles):

Sample analog output from function generator or linear amplifier output on RF front end. XADC wizard from IP catalog will be used to create module, employing onboard ADC for function generator testing or communicating via digital protocol with ADC on front end. Samples will be passed at maximum ADC sampling rate (1 MSPS) to next module with minimum delay.

DAC (Chaney):

Imported from the IP catalog and used to output the signal_out to a speaker or headphones with a 3.5 mm jack (just like in lab 5a).

Bandpass 1 - N (Charles):

N number (between 0 and 8, depending on modulation schemes implemented in final version) of filters with input from ADC and output to modulation modules. Center frequency set to intermediate frequency. Bandwidth for AM will be 10 kHz.

Full Spectrum FFT (Chaney):

Performs FFT on output of ADC module band pass filter. Outputs to display module.

Narrow Band FFT (Chaney):
Performs FFT on output of selected band pass filter. Outputs to display module.

Peak Detect and Zero Order Hold (AM) (Charles):
Demodulates AM via envelope detection. Identifies peaks in input signal, and when detected outputs value of input signal at peak, if a positive value. Requires a set of previous input values to be stored in order to calculate derivative. Registers a peak if the absolute value of derivative is less than a certain threshold.

Digitizer (FM) (Charles):
Translates input sinusoidal signal into square wave with defined amplitude (min = 0, max = 1). This is performed in real-time so the output will be a binary value. The input is an FM signal. This could potentially be accomplished through identifying the zero crossing point.

Phase Locked Loop (PLL) (FM) (Charles + Chaney):
Input is a digitized clock-like FM signal ranging from 0 to 1. While tracking the phase of the input signal, the input to the VCO will be the derivative of the phase which is equal to the original information frequency, in this case audio frequency. This output from this will be the information frequency.

Center Frequency Adjust (Charles):
The user should be able to tune the radio from the FPGA board by pressing the left button and the right button. If we are using a function generator for our test, this is somewhat irrelevant. If we are using our RF front-end, though, we should be able to output a voltage that makes the RF frontend shift our desired center frequency to the intermediate frequency that we do our filtering at.

VGA Signal Generators (Chaney):
Module that creates the hcount, hsync, vcount, vsync signals based on the correct input frequency for a specific resolution

Pixel Logic (Chaney):
Depending on what our user wants to display to the accompanying monitor, we will have different pixel logics that determine what our RGB out will be. The different outputs are as listed:

Output 1 - Based on the FFT of our input signal, we can display the entire frequency spectrum of our input signal rather than just the intermediate frequency plus the band width. Based on the input from the user, we will also show the box for the bandwidth around our intermediate frequency. With our analog front end, we can also specify a center frequency. This would change the numbers on our scale and would accurately reflect what our front end is doing

Output 2 - Based on the FFT of our input signal after bandpassing for a specific frequency, we will store that information over time to create a waterfall display. The first thing we will output at the waveform for the FFT (only within our bandwidth). Below that will be the waterfall display where different colors can be used to describe the intensity of the signal over time

Output 3 - If the user simply wants to see the audio going into the speaker, the waveform being sent to the DAC will be displayed to the screen

Section 4: Testing Procedures

For testing, we will be heavily using the function generators in the lab. Since having an RF-front end is not guaranteed, we will simulate these signals with the function generators. The function generators are able to modulate input signals from outside sources in the schemes that we wish to demodulate. When necessary, we will try to create test benches for testing modules without uploading to the FPGA. Some testing will ultimately have be done primarily on the FPGA, such as VGA output to the monitor.

Section 5: Project Timeline

We want to focus on getting our bare minimum requirements done first, then from there working on our more complex goals. Below is a general outline for the timeline of getting things done.

Week	Actionable(s)	Class Deadlines	Milestones
Week of 10/29	1. Take in un-modulated audio from function generator, output to speaker, and display on monitor (Chaney) 2. Setup testing configuration such that we can modulate audio source from phone and input into FPGA (Charles)	Block Diagram Meetings / Project Proposal	
Week of 11/04	1. Create a bandpass filter for the IF and bandwidth of AM (Charles) 2. Demodulate the AM signal such that we can play the audio signal and display it on the monitor (Charles) 3. Implement an FFT for the original signal and get it to display on the monitor with the bandwidth (Chaney)	Project Design Presentation	Absolute Bare Minimum Met
Week of 11/11	1. Create a bandpass filter for the IF and bandwidth of FM Radio (Charles)	Project Checklist Meeting	

	2. Demodulate the FM signal such that we can play the audio signal and display it on the monitor (Charles + Chaney)		
Week of 11/18	1. Perform an FFT on the band-passed signal we are outputting and write the intensity back in memory. Use this to display a waterfall along with the narrow FFT (Chaney) 2. Develop interface for RF front end, including control of local oscillator and ADC read (Charles)		Desired Functions Met- Rest is just stretch goal
Week of 11/25	1. Re-evaluate the feasibility of having a working RF front-end and see what all has to be done to get it to work (Charles + Chaney)	Short Week	
Week of 12/02	1. Select one or more of the stretch goals and try to implement them (Charles + Chaney) 2. Debug all past points to make sure they are operating as expected (Charles + Chaney)		Project Development ends
Week of 12/09	1. Make Report 2. Film Checkoff	Project Report and Checkoff	

Section 6 External Components:

RF Front End:
Antenna -> LNA -> ADC (10 MSPS)
Working with Joe Steinmeyer on development of analog components