FM Radio Receiver with Digital Demodulation

A Senior Project

presented to

the Faculty of the Electrical Engineering Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science

by

Nicholas Burnett

June, 2010

© 2010 Nicholas Burnett
# Table of Contents

1. Introduction ........................................................................................................................................ 1

2. Background ......................................................................................................................................... 3

3. Requirements ....................................................................................................................................... 5

4. Design .................................................................................................................................................. 6

5. Implementation and Construction ........................................................................................................ 27

6. Integration and System Testing ............................................................................................................. 52

7. Conclusion .......................................................................................................................................... 53

8. Bibliography ........................................................................................................................................ 54

9. Appendix .......................................................................................................................................... 55

   A. Parts Lists and Costs ....................................................................................................................... 55

   B. Time Table ..................................................................................................................................... 56

   C. VHDL Code Listing ......................................................................................................................... 58
List of Figures

Figure 1 - Digilent NEXYS Board with the Xilinx Sparten3-200 FPGA ........................................ 3
Figure 2 - Phase Lock Loop Overview .......................................................................................... 4
Figure 3 - Phase Lock Loop with Signals ...................................................................................... 6
Figure 4 – FM Radio Spectrum of a Single Channel [3]................................................................. 9
Figure 5 - Accumulator Type NCO................................................................................................. 11
Figure 6 - FM Front End Block Diagram ....................................................................................... 12
Figure 7 - RF Amplifier Schematic ............................................................................................... 13
Figure 8 - RF Filter Simulated Magnitude Response ..................................................................... 14
Figure 9 - RF Filter Schematic ..................................................................................................... 15
Figure 10 - Gilbert Cell Mixer Schematic ..................................................................................... 16
Figure 11 - TDK107MS Filter ........................................................................................................ 17
Figure 12 - Automatic Gain Control Schematic ............................................................................ 18
Figure 13 - Colpitts Oscillator Schematic ..................................................................................... 20
Figure 14 - Colpitts Simulated Output, Harmonix Index 1.0=11.06MHz ....................................... 20
Figure 15 - VCO Frequency versus Control Voltage Measurements ............................................. 21
Figure 16 - Voltage Controlled Oscillator Schematic .................................................................... 23
Figure 17 - General 3-Pole Sallen Key Amplifier Circuit Diagram ................................................ 24
Figure 18 - Anti-Aliasing Filter and Amplifier Schematic ............................................................... 25
Figure 19 - Anti-Aliasing Filter Simulated Magnitude Response ................................................... 26
Figure 20 - FM Demodulator Block Diagram ............................................................................... 27
Figure 21 - Digilent PmodAD1....................................................................................................... 28
Figure 22 - PmodAD1 VHDL Interface Component ................................................................. 28
Figure 23 - PmodAD1 State Diagram ....................................................................................... 30
Figure 24 - Digilent PmodDA2 ............................................................................................... 31
Figure 25 - PmodDA2 VHDL Interface Component ............................................................... 32
Figure 26 - PmodDA2 State Diagram ....................................................................................... 33
Figure 27 - Phase Detector Component ................................................................................... 34
Figure 28 - Lowpass Filter Component .................................................................................... 35
Figure 29 - NCO component .................................................................................................. 37
Figure 30 - RF Amplifier Picture ............................................................................................ 40
Figure 31 - RF Signal before Amplifier Stage ....................................................................... 41
Figure 32 - RF Signal after Amplifier Stage .......................................................................... 41
Figure 33 - RF Filter Picture .................................................................................................. 42
Figure 34 - RF Filter Measured Magnitude Response ............................................................ 43
Figure 35 - 10.7MHz IF Gilbert Cell Mixer Picture ............................................................... 44
Figure 36 - 360KHz IF Gilbert Cell Mixer Picture .................................................................. 44
Figure 37 - 10.7MHz Bandpass Filter Measure Magnitude Response .................................... 45
Figure 38 - Automatic Gain Control Measured Response ....................................................... 46
Figure 39 - Automatic Gain Control Circuit Picture .............................................................. 47
Figure 40 - 11.06MHz Colpitts Oscillator Picture ................................................................. 48
Figure 41 - Voltage Controlled Oscillator Picture ................................................................. 49
Figure 42 - Anti-Aliasing Filter and Amplifier Picture ............................................................ 50
Figure 43 - Voltage Regulator ............................................................................................... 51
Figure 44 - Integrated FM Receiver ........................................................................................................... 52
Acknowledgements

I would like to thank my advisor Dennis Derickson for the skills in RF design that allowed me to complete this project. I learned about phase locked loops by taking Derickson’s Advanced Analog Circuits course. The lab for the Advanced Analog Circuits course, where we built a spectrum analyzer, is where I got all my hands on experience in RF circuitry and design that made this project possible.

-- Nicholas Burnett
Abstract

This paper reports on the design, construction, and testing of an FM receiver. The design is split into two portions, the analog FM front end and the digital demodulator. The job of the front end is to down convert the RF signal to a frequency that is low enough to sample with an analog to digital converter. The construction of the front end is done using what is known as “Ugly Construction.” That is, all the components are soldered together floating over a ground plane.

The second portion of the design is the demodulator. The phase lock loop method of demodulating FM signals is used. The phase lock loop demodulator is designed in the digital domain on an FPGA. The design approach used for the demodulator is a digital hardware implementation using VHDL.
1. Introduction

In the past radio receivers were designed with analog circuitry. This inherently has the same problems that all analog circuits have. That is, they are susceptible to temperature variations, electrical noise, component aging, and they are complicated and inflexible. Initially, as digital circuits and processors were developed, they were not useful for radio or any high frequency circuitry since they operated at low frequencies and their transistor density was not enough for the signal processing needed in receivers. However, with the exponential increase in transistor density, faster clock rates, and faster A/D converters radio frequency receivers and possibly higher frequency receivers and transmitters are now suited for the digital domain.

Today, with transistor densities in the billions and clock rates in the GHz range, digital receivers are everywhere. Because of the advantages of digital communication systems, a concept of Software Defined Radio (SDR) has become popular in the literature. The ideal concept of SDR is to sample the RF signal with as little as possible analog manipulation. That is, ideally we would have an A/D converter at the output of an antenna and do all of the require signal processing in the digital domain. However, sufficiently fast A/D converters are not cheap enough yet, therefore we still require a front end to generate an intermediate frequency to sample. Once the signal is in the digital domain the designer has all the benefits of digital signal processing as described before, and the ease of configuration and reconfiguration. This senior project paper reports on the design and implementation of an FM receiver front end and of a digital phase lock loop design to demodulate FM broadcast signals. The digital phase lock loop will be
designed in VHDL on a Digilent NEXYS FPGA board. The goal of the front end of the receiver is to down convert the RF signal to an IF frequency that is possible to sample using the A/D converter that Digilent sells for the NEXYS FPGA board.
2. Background

A digital FM demodulator design can either be done in software or hardware. There are tradeoffs between the two design approaches. Software is quicker to implement because you can write in high level languages such as C. However, if the design is done in software more clock cycles will be required to complete all the operations necessary since only one operation can happen per clock cycle. If the design is done in hardware many circuits can be running simultaneously synced by the same clock, resulting in a much faster design. However, the design will need to be done in a hardware description language such as VHSIC Hardware Description Language (VHDL) or Verilog, this will require a more tedious design process. The FM demodulator design chosen for this project is done in hardware on a Digilent NEXYS FPGA board. The NEXYS FPGA board uses the Xilinx Spartan3 FPGA chip. FPGAs are Field-Programmable Gate Arrays. That is, they contain programmable logic blocks that can be reconfigured with a hardware description language. The hardware description language used is VHDL. The NEXYS board that is used in this project is shown in figure 1.

![Figure 1 - Digilent NEXYS Board with the Xilinx Spartan3-200 FPGA](image-url)
The technique used to design the FM demodulator is the phase locked loop. A rough idea of how a phase lock loop can demodulate an FM signal follow: A phase comparator is used to generate an error signal corresponding to the difference in phase between the input signal and a reference signal. The phase comparator is often a multiplier and therefore we will produce additional signals that will need to be filtered out with a digital filter. The reference signal is generated by a numerically controlled oscillator whose frequency is determined by the error signal. A basic FM demodulator is shown in figure 2.

![Phase Lock Loop Overview](image)

While the digital demodulation is the most interesting part of the project, the larger and more complicated part of the project is the front end of the receiver. The front end is used to down convert the RF signal to a frequency band that is possible to sample using the A/D converters on the NEXYS board. A two stage superheterdyne receiver will be used for the frontend. The complexity of the analog superheterodyne receiver will illustrate why the concept of software defined radio has become so popular.
3. Requirements

The requirements for the front end of the FM demodulator are as follows:

1. The front end will step down the selected channel of the FM band to 360kHz so that the signal can be sampled by the NEXYS PmodAD1 A/D converter.

2. The front end will filter the input signal so that only the selected channel is sampled by the A/D converter.

3. The channel selection will be controlled by the NEXYS board.

The requirements for the digital FM demodulator are as follows:

1. The digital FM demodulator must demodulate the selected FM channel. The output should be sent through the D/A to an amplifier driving a speaker.

2. The buttons on the NEXYS board will be used to tune the frontend to the desired FM station.
4. Design

The design is broken into two sections. The first section is the FM demodulator design. The second section will be the design of the FM radio front end.

4.1 FM Demodulator Design

The technique used to demodulate the FM signal is the popular phase lock loop demodulator. The design approach is to design the FM demodulator as if it were an analog phase lock loop, only it will be implemented with digital components rather than their analog counter parts. Therefore, the following design presented here works whether an analog phase detector, analog filter, and analog voltage controlled oscillator are used or a digital phase detector, digital filter, and a digital numerically controlled oscillator are used. Instead of voltages, digital words are used to represent the signal. The most basic phase lock loop used for FM demodulation consists of a Phase detector, loop filter and a voltage controlled oscillator. The diagram in figure 3 shows how these components are arranged.

![Diagram of Phase Lock Loop with Signals](image-url)

*Figure 3 - Phase Lock Loop with Signals*
4.1.1 Phase Detector

The phase detector was implemented with a simple multiplier. While other phase detector designs exist, a multiplier is the simplest to implement. In the VHDL model we could use a Booth multiplier if area is a greater concern or a Wallace-tree multiplier if high speed is of greater concern [1]. However, FPGA board are optimized for certain multiplier architectures, therefore, we will let the Xilinx ISE determine the multiplier to use. We will not be using any negative numbers in this design. All signals will be positive integers. As a consequence, the output is more complicated since the inputs will be DC shifted. If we had signed inputs with a zero DC offset, the output would be as follows:

\[ V_{\text{ref}} = V_{\text{ref}_0} \cos(w_{\text{ref}} + \theta_{\text{ref}}) \quad V_{\text{vco}} = V_{\text{vco}_0} \cos(w_{\text{vco}} + \theta_{\text{vco}}) \]

\[ V_{\text{ref}} V_{\text{vco}} = V_{\text{ref}_0} V_{\text{vco}_0} \cos(w_{\text{ref}} + \theta_{\text{ref}}) \cos(w_{\text{vco}} + \theta_{\text{vco}}) \]

\[ \frac{V_{\text{ref}_0} V_{\text{vco}_0}}{2} \left[ \cos(w_{\text{ref}} + w_{\text{vco}} + \theta_{\text{ref}} + \theta_{\text{vco}}) \cos(w_{\text{ref}} - w_{\text{vco}} + \theta_{\text{ref}} - \theta_{\text{vco}}) \right] \]

When locked,

\[ V_{\text{ref}} V_{\text{vco}} = \frac{V_{\text{ref}_0} V_{\text{vco}_0}}{2} \left[ \cos(2w_{\text{ref}} + \theta_{\text{ref}} + \theta_{\text{vco}}) + \cos(\theta_{\text{ref}} - \theta_{\text{vco}}) \right] \]

We will design a filter to remove the 2\(w_{\text{ref}}\) term and will be left with

\[ \frac{V_{\text{ref}_0} V_{\text{vco}_0}}{2} \cos(\theta_{\text{ref}} - \theta_{\text{vco}}) \]

A term related to the phase error between the two signals. However, in our case we have DC shifted values and the output will be as follows:

\[ V_{\text{ref}} = V_{\text{ref}_0} \cos(w_{\text{ref}} + \theta_{\text{ref}}) + V_{\text{ref}_0} \quad V_{\text{vco}} = V_{\text{vco}_0} \cos(w_{\text{vco}} + \theta_{\text{vco}}) + V_{\text{vco}_0} \]

\[ V_{\text{ref}} V_{\text{vco}} = \left[ V_{\text{ref}_0} \cos(w_{\text{ref}} + \theta_{\text{ref}}) + V_{\text{ref}_0} \right] \left[ V_{\text{vco}_0} \cos(w_{\text{vco}} + \theta_{\text{vco}}) + V_{\text{vco}_0} \right] \]

\[ = V_{\text{ref}_0} V_{\text{vco}_0} \cos(w_{\text{ref}} + \theta_{\text{ref}}) \cos(w_{\text{vco}} + \theta_{\text{vco}}) + V_{\text{ref}_0} V_{\text{vco}_0} \cos(w_{\text{vco}} + \theta_{\text{vco}}) \]
\[ +V_{ref_0}V_{vco_0} \cos(w_{ref} + \theta_{ref}) + V_{ref_0}V_{vco_0} \]

The first term is what we had before when we had no DC offset. However, with the DC offset we get these extra terms:

\[ V_{ref_0}V_{vco_0} \cos(w_{vco} + \theta_{vco}) + V_{ref_0}V_{vco_0} \cos(w_{ref} + \theta_{ref}) + V_{ref_0}V_{vco_0} \]

In addition to filtering out the signal at twice the frequency, we will also need to filter out the signals at the reference frequency. After filtering and when locked, the signal should be as follows:

\[ \frac{V_{ref_0}V_{vco_0}}{2} \cos(\theta_{ref} - \theta_{vco}) + V_{ref_0}V_{vco_0} \]
4.1.2 Loop Filter

The digital filter is implemented with a first order low pass filter described by the following transfer function:

\[ H_{LP} = \left( \frac{1 - \alpha}{2} \right) \frac{z + 1}{z - \alpha} \]

The 3-dB cutoff is determined by:

\[ \Omega_c = \cos^{-1} \left( \frac{2\alpha}{1 + \alpha^2} \right) \quad \Omega_c = 2\pi f_c T_s \]

Where \( f_c \) is the cutoff frequency and \( T_s \) is the sampling period ([2], pg. 650). If we choose \( f_c = 15 \) kHz then we will be able to demodulate the mono audio signal in an FM station as shown in figure 4.

![Figure 4 – FM Radio Spectrum of a Single Channel [3]](image)

With sampling rate \( T_s = 1.25\text{MHz} \) and \( f_s = 15\text{kHz} \) \( \alpha \) is found as follows:

\[ \Omega_c = 2\pi (15\text{kHz}) \left( \frac{1}{1.25\text{MHz}} \right) = 0.0754 \]

\[ \cos(0.0754) = 0.99716 \]

\[ \Omega_c = \cos^{-1} \left( \frac{2\alpha}{1 + \alpha^2} \right) \rightarrow \alpha^2 \cos(\Omega_c) - 2\alpha + \cos(\Omega_c) \]
\[
\alpha = 0.9273
\]

Therefore, our transfer function looks like:

\[
H_{LP}(z) = \left(\frac{1 - 0.9273}{2}\right) \left(\frac{z + 1}{z - 0.9273}\right) = 0.03635 \left(\frac{1 + z^{-1}}{1 - 0.9273z^{-1}}\right)
\]

Now taking the inverse Z transform,

\[
y(n) = 0.03635x(n) + 0.03635x(n - 1) + 0.9273y(n - 1)
\]

However, we are not going to use floating point values on our FPGA, instead we will multiply all the coefficients by \(2^{16}\), round to the nearest integer, and then shift the result to the right by 16 bits to effectively divide by \(2^{16}\). 16 is an arbitrary number of bits that gave a reasonable resolution to the floating point values calculated.

\[
y(n) = (2^{-16})(2382x(n) + 2382x(n - 1) + 60771y(n - 1))
\]
4.1.3 Numerically Controlled Oscillator (NCO)

The NCO used is of the standard accumulation type. That is, every clock cycle you add a word to the accumulator that corresponds to the output frequency. The accumulator is then used to index into a cosine ROM, which then produces the sinusoidal output. This is shown in the diagram in figure 5. For this technique higher frequency signals are produced with lower resolution than lower frequency signals. The Offset signal is used to get NCO oscillating around 360kHz. The tune signal comes from the filtered error voltage of the phase detector.

![Figure 5 - Accumulator Type NCO](image)
4.1.4 Channel Selection

There will be some logic required to increase or decrease the voltage at the second DAC output so that the DAC output can be used to tune the VCO for channel selection. From the VCO design section we know that the VCO frequency increases 6KHz per count. The Channel section circuit is designed so that if you press and hold button 1 the frequency will increase 60KHz every second. If you press and hold button 2, the frequency will decrease 60KHz every second. This allows the user to scan for the radio station they wish to listen to.

4.2 FM Front End Design

The front end design used is a dual conversion superheterodyne front end. We must use two stages because the best ceramic filters for channel selection are at an IF of 10.7MHz, however the A/D converter used has built in filters with a cut-off at 500kHz and the maximum sampling rate is 1 MSPS. To meet the Nyquist criteria we will use a second stage IF of 360KHz. A block diagram with all the components that are required is shown in figure 6.

*Figure 6 - FM Front End Block Diagram*
4.2.1 RF Amplifier

High frequency monolithic amplifiers are used for the RF amplifier stage. The Mini-Circuits MAV-11 amplifiers were chosen for the following specifications: 50 Ohm Input and Output Impedance, Operate between 50 and 1000MHz, and 12dB of gain at 100MHz. The maximum output power is 18.2dBm with an IP3 of 34 dBm. Because it is possible to receive up to 0dBm of power if close to a radio station, only two stages of MAV-11 amplifiers are used, for a total of 24dB of gain. The circuit diagram is shown in figure 7. The biasing resistors R1 and R2 were chosen to be 100Ohms based on the suggested biasing resistors in the datasheet for the MAV-11s.

![RF Amplifier Schematic](figure_7)

*Figure 7 - RF Amplifier Schematic*
4.2.2 RF Filter

FM broadcast stations are between 88MHz and 108MHz. Therefore, any signal that is received on the antenna that is not in this range should be rejected. To accomplish this, a 5\textsuperscript{th} order maximally flat filter with a lower cutoff frequency at 88MHz and an upper cutoff frequency of 118MHz is used. The upper cutoff is 118MHz instead of 108MHz, because when building the actual filter the cutoffs tend to be much lower than the theoretical cutoffs. Agilent Advanced Design System software is used for the design. The filter design and magnitude response are shown in figures 8 and 9 respectively.

![Figure 8 - RF Filter Simulated Magnitude Response](image-url)
Figure 9 - RF Filter Schematic
4.2.3 Mixers

Gilbert cell mixers are used in this design. The advantage of using Gilbert cell mixers is that there is RF to IF gain instead of insertion loss. HFA3101 transistor packages were used. These transistors are already configured in a gilbert cell configuration, they just need biasing. These gilbert cells are biased for about 10dB of gain. The biasing circuit is the circuit suggested in the datasheet. However, for the mixer with an IF of 360KHz a choke of 66uH was used instead. The circuit is shown in figure 10.

![Gilbert Cell Mixer Schematic]

*Figure 10 - Gilbert Cell Mixer Schematic*
4.2.4 10.7MHz IF Bandpass Filter

Each FM channel has 200kHz between its center frequency and the center frequency of the next channel. The bandwidth of each channel is 100KHz. Therefore, we must use a ceramic filter to filter at the 10.7MHz IF. The filter chosen is the TDK107MS ceramic filter. The datasheet reports a bandwidth of 180KHz. An image of the filter is shown in figure 11.

![Image of TDK107MS Filter](image)

*Figure 11 - TDK107MS Filter*
4.2.5 Automatic Gain Control

The input power for the selected station can vary widely. In order for the design to work with varying power levels we need to amplify the input signal to a constant power level. The Analog Devices AD603 variable gain amplifiers, along with a half-wave power detector were used for this task. The design is a reference design from the datasheet for the AD603. R2 and R5 are set to 10kOhm to give each stage a maximum gain of 46dB resulting in a total maximum gain of 92dB. The BJTs used for the detector are the BJTs on the CA3086F chip, however just about any BJT will work. The AGC schematic is shown in figure 12.

Figure 12 - Automatic Gain Control Schematic
4.2.6 11.06MHz Oscillator

The output of the 11.06MHz oscillator will be the LO for the second stage mixer. It will convert the 10.7MHz IF down to 360kHz. A Colpitts oscillator design was used. The BC548 transistor was used because of its relatively high gain-bandwidth product of 100MHz. It is not the ideal transistor, a transistor with a higher gain-bandwidth product would be ideal. The inductor values and capacitor values for resonance at 11.06MHz were calculated as follows:

\[ f_o = \frac{1}{2\pi \sqrt{LC_T}} \]

\[ C_T = \frac{1}{(2 \times \pi f_o)^2 L} \]

If we choose \( L = 2\mu H \),

\[ C_T = \frac{1}{(2\pi(11.06 \times 10^6))^2 (2 \times 10^{-6})} = 104pF \]

If \( C_2 = 470pf \), \( C_1 \) is calculated as follows,

\[ C_T = \frac{C_1C_2}{C_1 + C_2} \]

\[ C_1 = \frac{C_T C_2}{C_2 - C_T} = \frac{(104pF)(470pF)}{104pF + 470pF} = 133pF \]

The final values were optimized in ADS. The transistor was biased for a collector current of 10mA. C8 and C7 are used as an impedance transformer with \( N = 1/5.7 \). These values were optimized in ADS to reduce distortion but maximize power to the output. The output is then buffered by a NLB-310-T1 amplifier, which is just like the MAV-11 only it works from DC to 10GHz. The schematic is shown in figure 13. The simulated results in figure 14 show an output power of -1.8dBm at 11.06MHz.
Figure 13 - Colpitts Oscillator Schematic

Figure 14 - Colpitts Simulated Output, Harmonix Index 1.0=11.06MHz
4.2.7 Voltage Controlled Oscillator

The POS-200 was selected for the voltage controlled oscillator. The output of the POS-200 will be mixed with the RF to generate the 10.7MHz IF. The POS-200 generates signals between 100MHz and 200MHz, which is the right range for high side injection. The control voltage of the VCO is controlled by a summing amplifier. The tuning voltage from the NEXYS board is summed with an offset voltage. The summer op-amp circuit is then followed by an inverter, that output is then tied to the control pin of the VCO. Any general purpose op-amp will work in this design. The POS-200 frequency versus control voltage was measured; the results of the measurements are shown in figure 15.

<table>
<thead>
<tr>
<th>Voltage (V)</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>88</td>
</tr>
<tr>
<td>1</td>
<td>95</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
</tr>
<tr>
<td>3</td>
<td>111</td>
</tr>
<tr>
<td>4</td>
<td>119</td>
</tr>
<tr>
<td>5</td>
<td>126</td>
</tr>
<tr>
<td>6</td>
<td>134</td>
</tr>
<tr>
<td>7</td>
<td>141</td>
</tr>
<tr>
<td>8</td>
<td>149</td>
</tr>
<tr>
<td>9</td>
<td>157</td>
</tr>
<tr>
<td>10</td>
<td>165</td>
</tr>
<tr>
<td>11</td>
<td>173</td>
</tr>
<tr>
<td>12</td>
<td>181</td>
</tr>
<tr>
<td>13</td>
<td>189</td>
</tr>
<tr>
<td>14</td>
<td>197</td>
</tr>
<tr>
<td>15</td>
<td>205</td>
</tr>
</tbody>
</table>

Note: Output power was constant as a function of tune voltage

Figure 15 - VCO Frequency versus Control Voltage Measurements
The D/A converter has a full scale range between 0 and 3.3V. The resolution of the D/A converter is 12 bits therefore,

\[ 1 \text{ Count} = \frac{3.3V}{4096} = 0.8057\text{mV} \]

The frequency is linearly proportional to the control voltage, where the constant of proportionality is read from the slope in figure 15.

\[ K_{\text{VCO}} = 7.8044\text{MHz/V} \]

Therefore, the smallest change in frequency possible, \( \Delta F \) is as follows,

\[ \Delta F = \left( \frac{7.804\text{MHz}}{V} \right) (0.8054\text{mV}) = 6\text{kHz/Count} \]

With zero volts at the control voltage the VCO oscillates at 88MHz. The first FM channel requires that the VCO oscillates at 98.6MHz. However, to leave room for error we will use an offset voltage that brings the VCO up to 98.14MHz.

\[ 98.14\text{MHz} - 88\text{MHz} = K_{\text{VCO}} V_o \]

\[ V_o = \frac{98.14\text{MHz} - 88\text{MHz}}{7.8044} = 1.3V \]

The 1.3V needed for the offset voltage of the VCO is generated by a voltage divider from the 3.4V regulator. The schematic for the voltage controlled oscillator circuit is shown in figure 16.
Figure 16 - Voltage Controlled Oscillator Schematic
4.2.8 Anti-Aliasing Filter and Amplifier

A 3 pole Sallen Key Butterworth active lowpass filter was designed as an anti-aliasing filter. The A/D converter already includes a 2-pole anti-aliasing filter with a cutoff of 500kHz, this filter is for additional anti-aliasing.

![Figure 17 - General 3-Pole Sallen Key Amplifier Circuit Diagram](image)

After writing nodal equations and finding the transfer function of the Sallen key filter circuit, shown in figure 17, the transfer function can be compared to the general form of a 3-pole Butterworth lowpass filter. Using this method we then have the values of resistors and capacitors for a 1Hz cutoff which are as follows. $R_{1o} = 1.292k\Omega$, $R_{2o} = 2.093k\Omega$, $R_{3o} = 3.698k\Omega$, $C_{1o} = 1mF$, $C_{2o} = 1mF$, $C_{3o} = 100\mu F$. $R_4 = 0\Omega$ and $R_5 = \infty \Omega$, due to stability issues we will avoid any gain in this design. The values of resistors are then scaled as follows.

$$F_c = \frac{1}{2\pi xy} \text{ Hz}$$

Where the resistor values are scaled by multiplying each value by $x$ and the capacitors are scaled by multiplying each capacitance by $y$. [4]
The 3dB cutoff is designed to be at 700KHz to avoid additional attenuation near 500kHz while still providing additional antialiasing from the higher order mixing terms of the mixer proceeding the anti-aliasing filter stage.

\[ xy = \frac{1}{2\pi F_c} = \frac{1}{2\pi (700 \times 10^3)} = 2.27 \times 10^{-7} \]

If we choose C2=100pF then,

\[ y = \frac{C2}{C2_o} = \frac{100\text{pF}}{1\text{mF}} = 1 \times 10^{-7} \]

\[ x = \frac{2.27 \times 10^{-7}}{1 \times 10^{-7}} = 2.27 \]

Now scaling up the resistor and capacitor values we find,

R1=2.94kΩ, R2=4.76kΩ, R3=8.41kΩ, C1=100pF, C2=100pF, C3=10pF

The lowpass filter is then followed by an amplifier with a gain of 12dB. The op-amps used are LF351s. LF351s have a gain-bandwidth product of 4MHz. The Anti-Aliasing Filter and Amplifier schematic are shown in figure 18. The simulated magnitude response is shown in figure 19.

![Figure 18 - Anti-Aliasing Filter and Amplifier Schematic](image)
Figure 19 - Anti-Aliasing Filter Simulated Magnitude Response
5. Implementation and Construction

This section will be broken into two portions, the first is the implementation and testing of the demodulator, the second part is the implementation and testing of the front end.

5.1 FM Demodulator

A more detailed overview of the components that will need to be implemented in VHDL is shown in figure 20. Each component will be discussed in detail in this section.

![Figure 20 - FM Demodulator Block Diagram](image-url)
5.1.1 A/D Converter

The AD Converter board used is the PmodAD1 from Digilent, shown in figure 21. These boards contain two ADCS7476MSPS 12-bit A/D Converters, along with two 2-pole Sallen-Key anti-aliasing filters. The full scale voltage range is from 0V to 3.3V. We use one of these A/D converts to sample the 455 kHz frequency modulated input signal. The maximum sampling rate of the A/D is 1 MSPS, therefore, we will just meet the nyquist criteria to sample our 455KHz signal.

The A/D converter uses the serial peripheral interface. Instead of using SPI directly, Digilent provides a VHDL component to interface with the A/D converter.

**VHDL PmodAD1 Interface Component**

The PmodAD1 interface component is shown in figure 22, and a description of the signals and usage follow.
**General Signals:**

CLK – System clock that the component will use to generate SCLK.

RST – Asynchronous Reset

**PmodAD1 Interface Signals:**

SDATA1 - The signal for the serial data retrieved from the A/D Converter.

SDATA2 – The signal for the serial data retrieved from the second A/D converter.

SCLK – The clock for the ADCS7476MSPS A/D converter chips.

nCS – The chip select signal to enable the A/D converter chips.

**User Interface Signals:**

DATA1(11:0) – The 12 bit digitized sample of the input waveform for the first A/D chip.

DATA2(11:0) – The 12 bit digitized sample of the input waveform for the second A/D chip.

START – The bit to set to tell the A/D to start the conversion process. Start must remain high during the conversion process.

DONE – This bit is set high when the A/D converter is ready for another conversion.
Component Usage

In order to properly use the component the state diagram in figure 23 must be followed.

We start off in the *Idle* state. To start a conversion we change START to ‘1’ and wait until the *ShiftIn* state is complete and we are in the *SyncData* state. Once we want to sync the data we flip START back to 0 and then the conversion will be pushed on to the DATA Buses.

The way this state diagram is implemented in VHDL code is to use a counter and invert START at the appropriate clock counts. That is, count to 1 then flip START to ‘1’.

The conversion will begin and take 18 clock cycles to complete. Therefore, when we count to 19 we flip START back to ‘0’ to reach the *Idle* state, where we will then have the data on the DATA bus and will then restart the counter. A full conversion takes 20 clock cycles (SCLK) to complete. If START flips back to 0 before the *SyncData* state is reached, then no conversion will take place.

A full description on how to use the A/D reference component is at [5]

http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,499&Prod=PMOD-AD1
5.1.2 D/A Converter

The D/A Converter board used is the PmodAD2 from Digilent, shown in figure 24. This board contains two DAC121S101 12 bit D/A converters. The full scale voltage range is 0V to 3.3V. Our requirements on the D/A converter are not so demanding since our output is an audio signal.

We will be operating the D/A at 12.5MHz. 20 clock cycles are required to complete a conversion. Therefore, our output will be sampled at 625kHz.

The D/A converter uses the serial peripheral interface (SPI). Instead of using SPI directly, Digilent provides a VHDL component to interface with the A/D converter.
VHDL PmodDA2 Interface Component

The PmodAD2 interface component is shown in Figure 25, and a description of the signals and usage follow.

**General Signals:**
CLK – System clock that the component will use to generate SCLK.
RST – Asynchronous Reset

**PmodDA2 Interface Signals:**
D1 – The signal for the serial output data for the PmodDA2
D2 – The signal for the second D/A converter on the PmodDA2
CLK_OUT – The clock for the DAC121S101 chips
NSYNC – The bit to latch the data inside the PmodDA2 after the data has been shifted into it.

**User Interface Signals:**
DATA1(11:0) – The 12 bit word that is to be converted by the D/A converter.
DATA2(11:0) – The 12 bit word that is to be converted by the second D/A converter.
START – The bit to set to start the conversion process.
DONE – The bit that is set high when the conversion process is complete.
Component Usage

In order to properly use the component, the state diagram in figure 26 must be followed.

Starting in the Idle state, we set START to ‘1’ to begin the conversion process. We wait until the ShiftOut is complete and we are in the SyncData state. We then set START to ‘0’ and the PmodAD2 will complete a conversion.

The way this state diagram is implemented in VHDL code is to use a counter and invert START at the appropriate clock counts. That is, count to 1 then flip START to ‘1’. The data will be shifted into the D/As and will take 18 clock cycles to complete. Therefore, when we count to 19 we flip START back to ‘0’ and reset the counter to complete a conversion. Therefore, a full conversion takes 20 clock cycles (CLK_OUT) to complete. If START flips back to 0 before SyncData is reached, then no conversion will take place.

A full description on how to use the D/A Converter is found at [6]

http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,487&Prod=PMOD-DA2
5.1.3 Phase Detector

The phase detector component is called MultPhaseDet in the VHDL model. The phase detector component is shown in figure 27. The phase detector is implemented by taking the two input signals, VCO(11:0) and REF(11:0) and multiplying them to product SOUT(11:0). The result is shifted to the right by 12 bits, to produce a 12bit output at SOUT(11:0).

entity MultPhaseDet is
  Port ( REF : in  STD_LOGIC_VECTOR (11 downto 0);
         VCO : in  STD_LOGIC_VECTOR (11 downto 0);
         SOUT : out  STD_LOGIC_VECTOR (11 downto 0));
end MultPhaseDet;

architecture Behavioral of MultPhaseDet is
  signal OUT_TEMP : STD_LOGIC_VECTOR (23 downto 0);
begin
  OUT_TEMP <= VCO * REF;
  SOUT <= (OUT_TEMP(23 downto 12));
end Behavioral;
5.1.4 Lowpass Filter

The lowpass filter component, shown in figure 28, is called Lowpass in the VHDL model. The component implements the recursive equation that was designed in the design section.

\[ y(n) = (2^{-16})(2382x(n) + 2382x(n - 1) + 60771y(n - 1)) \]

The design requires one previous value for the output and one previous value for the input. Registers are implemented in order to keep track of the previous values. The VHDL model should be redesigned now that I have learned more about VHDL. It is not necessary to keep track of two different last_inputs and last_outputs. When changing the value of a register, it does not take effect till the end of that clock cycle, therefore, you can read the old value of a register as you change it to a new value. The \((2^{-16})\) multiplication in the difference equation is accomplished with a shift to the right by 16 bits.
entity Lowpass is
  Port ( CLK : in  STD_LOGIC;
         RST : in  STD_LOGIC;
         FILTER_IN : in  STD_LOGIC_VECTOR (11 downto 0);
         FILTER_OUT : out STD_LOGIC_VECTOR (11 downto 0));
end Lowpass;

architecture Behavioral of Lowpass is
  signal CURRENT_INPUT : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal LAST_INPUT0 : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal LAST_INPUT1 : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal LAST_OUTPUT0 : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal LAST_OUTPUT1 : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal CURRENT_OUTPUT : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal COUNT : STD_LOGIC := '0';
begin
  CURRENT_INPUT(11 downto 0) <= FILTER_IN;
  process(CLK,RST)
  begin
    if (RST = '1') then
      LAST_INPUT0 <= "000000000000";
      LAST_INPUT1 <= "000000000000";
      LAST_OUTPUT0 <= "000000000000";
      LAST_OUTPUT1 <= "000000000000";
      CURRENT_OUTPUT <= "000000000000";
    elsif rising_edge(CLK) then
      COUNT <= NOT COUNT;  --To keep track of which last_input and last_output to use.
      if COUNT = '0' then
        LAST_INPUT1 <= CURRENT_INPUT;
        LAST_OUTPUT1 <= CURRENT_OUTPUT;
        CURRENT_OUTPUT <= SHR("100101001110" * CURRENT_INPUT + "100101001110" * LAST_INPUT0 +
                                "111011010110011" * LAST_OUTPUT0), "10000");  --2382,2382,60771
      elsif COUNT = '1' then
        LAST_INPUT0 <= CURRENT_INPUT;
        LAST_OUTPUT0 <= CURRENT_OUTPUT;
        CURRENT_OUTPUT <= SHR("100101001110" * CURRENT_INPUT + "100101001110" * LAST_INPUT1 +
                                "111011010110011" * LAST_OUTPUT1), "10000");
      end if;
    end if;
  end process;
  FILTER_OUT <= CURRENT_OUTPUT(11 downto 0);
end Behavioral;
5.1.5 Numerically Controlled Oscillator

The NCO component, shown in figure 29, is called NCO in the VHDL model. The component contains a cosine rom within it. The cosine rom has pre-calculated values for a cosine function of 1Hz up to $\pi/2$. Reconstructing the cosine wave from the first quadrant allows us to produce a higher resolution cosine wave with a smaller ROM. All the other portions of the cosine wave are constructed as follows:

For $0 \leq \text{PHASE\_ADDRESS} \leq 65536$ \( \rightarrow \) NCO\_OUT $\leq 2096 + \text{COS\_DATA(\text{PHASE\_ADDRESS})}$

For $65536 \leq \text{PHASE\_ADDRESS} \leq 131072$ \( \rightarrow \) NCO\_OUT $\leq 2096 - \text{COS\_DATA(131072-\text{PHASE\_ADDRESS})}$

For $131072 \leq \text{PHASE\_ADDRESS} \leq 196608$ \( \rightarrow \) NCO\_OUT $\leq 2096 - \text{COS\_DATA(\text{PHASE\_ADDRESS}-131072)}$

For $196608 \leq \text{PHASE\_ADDRESS} \leq 262144$ \( \rightarrow \) NCO\_OUT $\leq 2096 + \text{COS\_DATA(262144 - \text{PHASE\_ADDRESS})}$

Every clock cycle we add the OFFSET word and TUNE word to the PHASE\_ADDRESS, therefore, sampling our cosine rom. The larger the OFFSET and TUNE words are, the higher the frequency the output is.
entity NCO is
  Port ( CLK : in  STD_LOGIC;
         RST : in  STD_LOGIC;
         OFFSET : in  STD_LOGIC_VECTOR (23 downto 0);
         TUNE : in STD_LOGIC_VECTOR (11 downto 0);
         NCO_OUT : out  STD_LOGIC_VECTOR (11 downto 0));
end NCO;

architecture Behavioral of NCO is

  component CosRom
    port(
      ADDRESS : in STD_LOGIC_VECTOR (15 downto 0);
      COS_DATA : out STD_LOGIC_VECTOR (11 downto 0));
  end component;

  signal PHASE_ACCUM : STD_LOGIC_VECTOR(23 downto 0) := "000000000000000000000000";
  signal PHASE_ADDRESS : STD_LOGIC_VECTOR(17 downto 0) := "000000000000000000";
  signal ADDRESS : STD_LOGIC_VECTOR(15 downto 0);
  signal COS_DATA : STD_LOGIC_VECTOR(11 downto 0);
  signal FREQ : STD_LOGIC_VECTOR(23 downto 0);

  begin
    COSROM1: CosRom port map(ADDRESS, COS_DATA);
    FREQ <= OFFSET + TUNE;
    process(CLK,RST)
    begin
      if(RST = '1') then
        PHASE_ACCUM <= "000000000000000000000000";
      elsif(rising_edge(CLK)) then
        PHASE_ACCUM <= PHASE_ACCUM + FREQ;
        PHASE_ADDRESS <= PHASE_ACCUM(23 downto 6);

        if(PHASE_ADDRESS <= "100000000000000000") then
          ADDRESS <= PHASE_ADDRESS(15 downto 0);
          NCO_OUT <= "100000000000"+COS_DATA;
        elsif(PHASE_ADDRESS <= "100000000000000000" - PHASE_ADDRESS(15 downto 0)) then
          NCO_OUT <= "100000000000"-COS_DATA;
        elsif(PHASE_ADDRESS <= "110000000000000000") then
          ADDRESS <= PHASE_ADDRESS(15 downto 0) - "100000000000000000";
          NCO_OUT <= "100000000000"-COS_DATA;
        else
          ADDRESS <= "100000000000000000" - PHASE_ADDRESS(15 downto 0);
          NCO_OUT <= "100000000000"+COS_DATA;
        end if;
      end if;

      NCO_OUT <= NCO_OUT;
    end if;
  end process;
end Behavioral;
5.1.6 Channel Selection

Channel selection was implemented with an additional process instead of another component.

CLK1Hz is generated by dividing the main clock down by 25e6.

```
CHANNEL: process(RST, CLK1HZ)
begin
  if(RST = '1') then
    DATA2_DA <= "000000000000";
  elsif rising_edge(CLK1HZ) then
    if BTN2 = '1' then
      DATA2_DA <= DATA2_DA + 10;
    elsif BTN1 = '1' then
      DATA2_DA <= DATA2_DA - 10;
    end if;
  end if;
end process;
```
5.2 FM Front End

The construction and testing of each RF Frontend component will be presented in this section.

5.2.1 RF Amplifier

The RF amplifier circuit was built as shown in figure 7 in the design section. The constructed RF amplifier is shown in figure 30.

*Figure 30 - RF Amplifier Picture*
From figure 31 and figure 32, the spectrum analyzer shows a 17dB gain from the amplifier stage, this is 7dB lower than expected. The two scope captures are taken with the following spectrum analyzer settings: 101MHz center frequency, 2dBm reference, 50MHz frequency span, 10dB/Div, and a 30kHz resolution bandwidth.
5.2.2 RF Filter

The RF filter was built as shown in figure 9 of the design section. However, Variable capacitors were used for the three sections with capacitors to ground. These capacitors allow tuning of the RF filter. Additional tuning was accomplished by compressing the inductors. After iterating through capacitor tuning and inductor tuning, the response shown in figure 34 was achieved.

The picture of the constructed RF Filter is shown in figure 33.
As seen in figure 34 the magnitude response is larger than 0 dB around 91MHz. This is a passive circuit, so a gain greater than 0 dB should not be possible. However, this may be due to the inductors picking up RF signals.

The lower 3dB point of the magnitude response is at 88.5MHz the upper 3dB point is at 106MHz.
5.2.3 Mixers

The two mixers are constructed as they were designed in design section in figure 10. Figure 35 shows the constructed first stage mixer (10.7MHz IF), figure 36 shows the constructed second stage mixer (360KHz IF). The mixers were tested by using a frequency synthesizer on the RF and LO port and monitoring the output at the IF port with a spectrum analyzer.

*Figure 35 - 10.7MHz IF Gilbert Cell Mixer Picture*

*Figure 36 - 360KHz IF Gilbert Cell Mixer Picture*
5.2.4 10.7MHz Bandpass Filter

The 10.7MHz ceramic filter was characterized as shown in figure 37. The insertion loss was measured at 9dB. There are not enough points of resolution to determine the 3dB points however, they appear to be close to 10.55MHz and 10.8MHz.

![10.7MHz Filters Magnitude Response](image)
5.2.5 Automatic Gain Control

The AGC circuit was built as shown in figure 12 of the design section. The performance of the AGC circuit is shown in figure 38. The output is held at a constant -7dBm as long as the input is at least -60dBm. Figure 39 shows the constructed AGC circuit.

![Automatic Gain Control Circuit Response, RSB=100kHz, F=10.7MHz](image)

*Figure 38 - Automatic Gain Control Measured Response*
Figure 39 - Automatic Gain Control Circuit Picture
5.2.6 11.06MHz Oscillator

The Colpitts oscillator circuit was built as shown in the design section in figure 13. The oscillator output was verified to be at 11.06MHz with an oscilloscope. The crystal used is a 11.06MHz crystal. Figure 40 shows the constructed oscillator circuit.

*Figure 40 - 11.06MHz Colpitts Oscillator Picture*
5.2.7 Voltage Controlled Oscillator

The VCO control circuit and biasing were built as shown in the design section in figure 16. The first iteration of this design did not include the 100uF decoupling capacitor at the control node of the VCO. Initially the node was only decoupled with a 1uF capacitor, after decoupling with the 100uF capacitor the phase noise of the VCO got considerably better and thus was added to the design. The op-amps used are LF351s. However, any general purpose op-amp will work. Figure 41 shows the constructed VCO circuit.

Figure 41 - Voltage Controlled Oscillator Picture
5.2.8 Anti-Aliasing Filter and Amplifier

The Anti-Aliasing Filter and Amplifier were built as shown in figure 18 of the design section. Instead of using one resistor for the feedback resistor of the amplifier, a 10KOhm in series with a 50Kohm potentiometer was used. The potentiometer allows the user to control the gain so that, given that the input signal is high enough, we can adjust the peak voltage of the output to fit within our 3.3V range on the A/D converter. Note that this circuit is followed by a capacitor and resistor circuit that operates as a level shifter, DC shifting the output by 1.65V. The open nodes of the LF351 op-amps were decoupled to ground since they were picking up RF signals. The constructed anti-aliasing filter and amplifier is shown in figure 42.

Figure 42 - Anti-Aliasing Filter and Amplifier Picture
5.2.9 Additional Components

Initially this voltage regulator was going to be used on a rail for a direct digital synthesis oscillator chip that was going to be used instead of the VCO. However, this regulator was repurposed to supply the supply voltage required for the mixers and is used as the voltage source for other voltage dividers that are used throughout the circuit.

\[
\begin{align*}
V_i &= 10V \\
V_{out} &= 3.4V \\
R_1 &= 1k\Omega \\
R_2 &= 1.67k\Omega
\end{align*}
\]

*Figure 43 - Voltage Regulator*

The 1.3V for the offset in the VCO control circuit is achieved by using dividing the 3.4V output of the voltage regulator in figure 43. Also the 1.6V needed to DC shift the output of the anti-aliasing filter and amplifier is achieved with a voltage divider from this regulator as well.
6. Integration and System Testing

Each component was integrated together as shown in figure 44. A 75Ohm impedance FM antenna was plugged into the Antenna input of the system. The receiver is a 50Ohm system so there is a slight mismatch which may account for some signal attenuation of the RF signal. An audio amplifier was connected to the audio output, and then to a speaker. Initially, only faint whispers of music made it through the system, although it was mostly noise. However, after I discovered a bad amplifier and replaced it I was able to pick up the following stations at my house: 88.5, 89.3, and 91.3MHz. Using the large rooftop antenna on campus I was able to pick up many more stations although they were all noisier than the three stations I picked up with the small dipole antenna.
7. Conclusion

The FM receiver was a success. While not perfect, you can tune to FM radio stations and listen in. A further analysis of this prototype would need to be done to see why the output is noisy and why the receiver has to have very good reception in order to hear anything at all. Some suggestions for a better receiver are as follows:

- Additional gain at the RF stage
- Use balanced mixers
- Increase the output power of the automatic gain control circuit
- Shield RF sensitive components such as the inductors
- Use a faster transistor for the 11.06MHz oscillator
- Improve the phase lock loop to handle larger signals.

This FM receiver is not practical, that is, it is expensive, large, and noisy. There are $10 chips on the market that could replace most of this circuit. However, this method does give more experience in RF design than using a single chip. Also, implementing the phase lock loop on the FPGA gives good VHDL experience.
8. Bibliography


# 9. Appendix

## A. Parts Lists and Costs

<table>
<thead>
<tr>
<th>Project Parts</th>
<th>Notes</th>
<th>Cost Estimate</th>
<th>Quantity</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEXY Board</td>
<td></td>
<td>$90.00</td>
<td>1</td>
<td>$90.00</td>
</tr>
<tr>
<td>NEXYS A/D Converter</td>
<td>PmodAD1</td>
<td>$25.00</td>
<td>1</td>
<td>$25.00</td>
</tr>
<tr>
<td>NEXYS D/A Converter</td>
<td>PmodDA2</td>
<td>$25.00</td>
<td>1</td>
<td>$25.00</td>
</tr>
<tr>
<td>Voltage Controlled Oscillator</td>
<td>POS-200</td>
<td>$12.00</td>
<td>1</td>
<td>$12.00</td>
</tr>
<tr>
<td>Mixers</td>
<td>HFA3101</td>
<td>$6.00</td>
<td>2</td>
<td>$12.00</td>
</tr>
<tr>
<td>Variable Gain Amplifiers</td>
<td>AD603</td>
<td>$10.00</td>
<td>2</td>
<td>$20.00</td>
</tr>
<tr>
<td>RF Amplifier</td>
<td>MAV-11</td>
<td>$3.00</td>
<td>2</td>
<td>$6.00</td>
</tr>
<tr>
<td>RF Amplifier</td>
<td>689-1003-1-ND</td>
<td>$3.44</td>
<td>1</td>
<td>$3.44</td>
</tr>
<tr>
<td>11.06MHz Crystal</td>
<td></td>
<td>$0.59</td>
<td>1</td>
<td>$0.59</td>
</tr>
<tr>
<td>10.7MHz IF Filter</td>
<td>490-4710-ND</td>
<td>$0.36</td>
<td>1</td>
<td>$0.36</td>
</tr>
<tr>
<td>Op-Amps</td>
<td>LF351</td>
<td>$0.50</td>
<td>4</td>
<td>$2.00</td>
</tr>
<tr>
<td>Transistor Array</td>
<td>CA3096</td>
<td>$0.50</td>
<td>1</td>
<td>$0.50</td>
</tr>
<tr>
<td>Voltage Regulator</td>
<td>LM317</td>
<td>$0.30</td>
<td>1</td>
<td>$0.30</td>
</tr>
<tr>
<td>Audio Amplifier</td>
<td>LM386</td>
<td>$2.00</td>
<td>1</td>
<td>$2.00</td>
</tr>
<tr>
<td>Speaker</td>
<td></td>
<td>$5.00</td>
<td>1</td>
<td>$5.00</td>
</tr>
</tbody>
</table>

|                             |                 |               |          |        |
| Total                       |                 | $204.19       |          |        |
| Total W/Out NEXYS           |                 | $64.19        |          |        |
### B. Time Table

<table>
<thead>
<tr>
<th>Date</th>
<th>Time (Hours)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/25/10-3/1/2010</td>
<td>20</td>
<td>Researching digital PLLs, phase detectors and loop filters. Implemented parts of the PFD design in VHDL</td>
</tr>
<tr>
<td>3/2/2010</td>
<td>7</td>
<td>Scrapped PFD Design, Got ADC and DAC working properly</td>
</tr>
<tr>
<td>3/3/2010</td>
<td>7</td>
<td>Started 1st Order LP Filter</td>
</tr>
<tr>
<td>3/4/2010</td>
<td>4</td>
<td>Completed 1st Order LP Filter</td>
</tr>
<tr>
<td>3/7/2010</td>
<td>2</td>
<td>Started NCO</td>
</tr>
<tr>
<td>3/8/2010</td>
<td>3</td>
<td>Working on NCO</td>
</tr>
<tr>
<td>3/9/2010</td>
<td>6</td>
<td>Completed NCO</td>
</tr>
<tr>
<td>3/10/2010</td>
<td>6</td>
<td>Got it demodulating</td>
</tr>
<tr>
<td>3/12/2010</td>
<td>2</td>
<td>Demoing FM demodulator</td>
</tr>
<tr>
<td>3/14/2010</td>
<td>2</td>
<td>Writing EE463 Report</td>
</tr>
<tr>
<td>3/15/2010</td>
<td>4</td>
<td>Writing EE463 Report</td>
</tr>
<tr>
<td>3/30/2010</td>
<td>5</td>
<td>Implemented Moving Average Filter</td>
</tr>
<tr>
<td>4/5/2010</td>
<td>4</td>
<td>Receiver Frontend</td>
</tr>
<tr>
<td>4/15/2010</td>
<td>4</td>
<td>Found parts to buy for front end</td>
</tr>
<tr>
<td>4/26/2010</td>
<td>2</td>
<td>AGC Research</td>
</tr>
<tr>
<td>4/27/2010</td>
<td>3</td>
<td>AGC Construction</td>
</tr>
<tr>
<td>4/28/2010</td>
<td>3</td>
<td>AGC Construction</td>
</tr>
<tr>
<td>4/30/2010</td>
<td>5</td>
<td>AGC Measurements, DDS for LO</td>
</tr>
<tr>
<td>5/9/2010</td>
<td>5</td>
<td>DDS</td>
</tr>
<tr>
<td>5/11/2010</td>
<td>4</td>
<td>DDS</td>
</tr>
<tr>
<td>5/15/2010</td>
<td>6</td>
<td>DDS</td>
</tr>
<tr>
<td>5/16/2010</td>
<td>6</td>
<td>DDS</td>
</tr>
<tr>
<td>5/17/2010</td>
<td>4</td>
<td>DDS</td>
</tr>
<tr>
<td>5/18/2010</td>
<td>4</td>
<td>DDS</td>
</tr>
<tr>
<td>5/19/2010</td>
<td>4</td>
<td>Mixer and 10.7MHz Filter</td>
</tr>
<tr>
<td>5/20/2010</td>
<td>6</td>
<td>DDS, Mixer and 10.7MHz Test</td>
</tr>
<tr>
<td>5/21/2010</td>
<td>3</td>
<td>DDS</td>
</tr>
<tr>
<td>5/22/2010</td>
<td>6</td>
<td>DDS</td>
</tr>
<tr>
<td>5/23/2010</td>
<td>6</td>
<td>DDS (Gave up on it)</td>
</tr>
<tr>
<td>5/24/2010</td>
<td>3</td>
<td>Built Second Mixer</td>
</tr>
<tr>
<td>5/25/2010</td>
<td>3</td>
<td>RF Filter Design and Build</td>
</tr>
<tr>
<td>5/26/2010</td>
<td>2</td>
<td>Second Mixer Test</td>
</tr>
<tr>
<td>5/27/2010</td>
<td>2</td>
<td>RF Filter Test</td>
</tr>
<tr>
<td>5/28/2010</td>
<td>3</td>
<td>Design and build of VCO Control Circuit</td>
</tr>
<tr>
<td>5/29/2010</td>
<td>6</td>
<td>11.06Filter Design and Test</td>
</tr>
<tr>
<td>5/30/2010</td>
<td>7</td>
<td>500MHz Sallen Key LPF and Amplifier Design and Build</td>
</tr>
<tr>
<td>Date</td>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>5/31/2010</td>
<td>5</td>
<td>500MHz and amplifier redesign</td>
</tr>
<tr>
<td>6/1/2010</td>
<td>7</td>
<td>System Integration and Testing</td>
</tr>
<tr>
<td>6/7/2010</td>
<td>3</td>
<td>Project Report</td>
</tr>
<tr>
<td>6/8/2010</td>
<td>2</td>
<td>Project Report</td>
</tr>
<tr>
<td>6/9/2010</td>
<td>5</td>
<td>Project Report</td>
</tr>
<tr>
<td>6/10/2010</td>
<td>8</td>
<td>Project Report</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>199</strong></td>
<td></td>
</tr>
</tbody>
</table>
C. VHDL Code Listing

FMADPLL – Main Component

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FMADPLL is
  Port ( CLK : in  STD_LOGIC;
         RST : in  STD_LOGIC;
         SDATA1 : in  STD_LOGIC;
         SDATA2 : in  STD_LOGIC;
         OUTPUT_FILTER_SWITCH : in STD_LOGIC;
         BTN0 : in STD_LOGIC;
         BTN1 : in STD_LOGIC;
         BTN2 : in STD_LOGIC;
         SCLK : out  STD_LOGIC;
         nCS : out  STD_LOGIC;
         CLK_OUT : out  STD_LOGIC;
         NSYNC : out STD_LOGIC;
         D1 : out  STD_LOGIC;
         D2 : out  STD_LOGIC);
end FMADPLL;

architecture Behavioral of FMADPLL is
  component AD1RefComp
  Port ( --General usage
          CLK      : in std_logic;
          RST      : in std_logic;
  --Pmod interface signals
          SDATA1   : in std_logic;
          SDATA2   : in std_logic;
          SCLK     : out std_logic;
          nCS      : out std_logic;
  --User interface signals
          DATA1    : out std_logic_vector(11 downto 0);
          DATA2    : out std_logic_vector(11 downto 0);
          START    : in std_logic;
          DONE     : out std_logic
          );
  end component;

  component DA2RefComp
  Port ( --General usage
          CLK      : in std_logic;
          RST      : in std_logic;
    ...
--Pmod interface signals
D1       : out std_logic;
D2       : out std_logic;
CLK_OUT  : out std_logic;
nSYNC    : out std_logic;

--User interface signals
DATA1    : in std_logic_vector(11 downto 0);
DATA2    : in std_logic_vector(11 downto 0);
START    : in std_logic;
DONE     : out std_logic
);
end component;

cOMPONENT Lowpass IS
  PORT ( CLK : IN  STD_LOGIC;
         RST : IN  STD_LOGIC;
         FILTER_IN : IN  STD_LOGIC_VECTOR (11 downto 0);
         FILTER_OUT : OUT  STD_LOGIC_VECTOR (11 downto 0));
END COMPONENT;

COMPONENT MultPhaseDet IS
  PORT ( REF : IN  STD_LOGIC_VECTOR (11 downto 0);
         VCO : IN  STD_LOGIC_VECTOR (11 downto 0);
         SOUT : OUT  STD_LOGIC_VECTOR (11 downto 0));
END COMPONENT;

COMPONENT NCO IS
  PORT ( CLK : IN  STD_LOGIC;
         RST : IN  STD_LOGIC;
         OFFSET : IN  STD_LOGIC_VECTOR (23 downto 0);
         TUNE : IN  STD_LOGIC_VECTOR (11 downto 0);
         NCO_OUT : OUT  STD_LOGIC_VECTOR (11 downto 0));
END COMPONENT;

COMPONENT MAF IS
  -- Moving Average Filter
  PORT ( CLK : IN  STD_LOGIC;
         RST : IN  STD_LOGIC;
         MAF_IN : IN  STD_LOGIC_VECTOR (11 downto 0);
         MAF_OUT : OUT  STD_LOGIC_VECTOR (11 downto 0));
END COMPONENT;

Signal VREF : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
Signal DATA2_AD : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
Signal START_AD : STD_LOGIC := '0';
Signal DONE_AD : STD_LOGIC := '0';
Signal LOOPFILTER_IN : STD_LOGIC_VECTOR (11 downto 0) := "100000000000";
Signal LOOPFILTER_OUT : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
Signal MAFFILTER_OUT : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
Signal DATA1_DA : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
Signal DATA2_DA : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
Signal START_DA : STD_LOGIC := '0';
signal DONE_DA : std_logic;
signal nCS_S : std_logic;
signal NSYNC_S : std_logic;
signal CLK_OUT_S : std_logic;
signal SCLK_S : std_logic;
signal DA_TIMER : std_logic_vector(4 downto 0) := "00000";
signal AD_TIMER : std_logic_vector(4 downto 0) := "00000";
signal OFFSET : STD_LOGIC_VECTOR(23 downto 0) := "00000011001011001011010100";
--208053, 310kHz
signal NCO_OUT : STD_LOGIC_VECTOR(11 downto 0);
signal COUNT1HZ : STD_LOGIC_VECTOR(24 downto 0) := "0101111110111100001000000";
--25e6
signal CLK1HZ : STD_LOGIC := '0';

begin
  AD1: AD1RefComp port map (CLK, RST, SDATA1, SDATA2, SCLK_S, nCS_S, VREF, DATA2_AD, START_AD,
                           DONE_AD);
  Lowpass1: Lowpass port map (nCS_S, RST, LOOPFILTER_IN, LOOPFILTER_OUT); --Using nCS_s as the clock, since
                   its frequency is the same as the sample rate.
  DA1: DA2RefComp port map (CLK, RST, D1, D2, CLK_OUT_S, NSYNC_S, DATA1_DA, DATA2_DA, START_DA,
                           DONE_DA);
  MPD1: MultPhaseDet port map(VREF,NCO_OUT,LOOPFILTER_IN);
  NCO1: NCO port map(CLK,RST,OFFSET,LOOPFILTER_OUT,NCO_OUT);
  MAF1: MAF port map(nCS_S,RST,LOOPFILTER_OUT,MAFFILTER_OUT);

  SCLK <= SCLK_S;
  NSYNC <= NSYNC_S;
  nCS <= nCS_S;
  CLK_OUT <= CLK_OUT_S;

  DATA1_DA <= MAFFILTER_OUT when OUTPUT_FILTER_SWITCH = '1' else
              LOOPFILTER_OUT;

ADConversion: process(SCLK_S,RST)
begin
  if(RST = '1') then
    START_AD <= '0';
    AD_TIMER <= "00000";
  elsif(rising_edge(SCLK_S)) then
    AD_TIMER <= AD_TIMER + 1;
    if(AD_TIMER = "00001") then --Was in the IDLE state, go to SHIFTIN state
      START_AD <= '1';
    elsif(AD_TIMER = "10011") then --Should now be in the SYNCDATA state
      AD_TIMER <= "00000";
      START_AD <= '0';
    end if;
  end if;
end process;

DAConversion: process(CLK_OUT_S,RST)
begin
  if(RST = '1') then
    START_DA <= '0';
  elsif(rising_edge(CLK_OUT_S)) then
    ...
DA_TIMER <= DA_TIMER + 1;
if(DA_TIMER = "00001") then
    START_DA <= '1';
elsif(DA_TIMER = "10011") then
    START_DA <= '0';
    DA_TIMER <= "00000";
end if;
end if;
end process;

-- Purpose: Increase the base tuning word when a button is held down --
-- Decrease the base tuning word when a different button is held down --
-------------------------------------------------------------------

DDSFreq: process(CLK, RST)
beg
if(RST = '1') then
    OFFSET <= "000001110011011110110110"; --208053, 310kHz
    COUNT1HZ <= "010111111011111000100000"; --25e6
elsif rising_edge(CLK) then
    if COUNT1HZ = "0000000000000000000000000" then
        CLK1HZ <= NOT CLK1HZ;
        COUNT1HZ <= "010111111011111000100000"; --25e6
        if BTN0 = '1' then
            OFFSET <= OFFSET + "0000000000000000000000000"; --671, 1kHz
            --elsif BTN1 = '1' then
            --    OFFSET <= OFFSET - "0000000000000000000000000"; --671
            end if;
        else
            COUNT1HZ <= COUNT1HZ - 1;
        end if;
    end if;
end if;
end process;

-- Purpose: Increment the voltage on DA2, for VCO channel selection --
-------------------------------------------------------------------

CHANNEL: process(RST, CLK1HZ)
beg
if(RST = '1') then
    DATA2_DA <= "000000000000";
elsif rising_edge(CLK1HZ) then
    if BTN2 = '1' then
        DATA2_DA <= DATA2_DA + 10;
    elsif BTN1 = '1' then
        DATA2_DA <= DATA2_DA - 10;
    end if;
end if;
end process;
end Behavioral;
AD1RefComp - A/D Reference Component

The A/D Reference Component used is copyright of Digilent. The component can be found at www.Digilentinc.com [5]

DA2RefComp – D/A Reference Component

The D/A Reference Component used is copyright of Digilent. The component can be found at www.Digilentinc.com [6]

Lowpass – Lowpass Filter Component

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Lowpass is
  Port ( CLK : in  STD_LOGIC;
         RST : in  STD_LOGIC;
         FILTER_IN : in  STD_LOGIC_VECTOR (11 downto 0);
         FILTER_OUT : out  STD_LOGIC_VECTOR (11 downto 0));
end Lowpass;

architecture Behavioral of Lowpass is
  signal CURRENT_INPUT : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal LAST_INPUT0 : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal LAST_INPUT1 : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal LAST_OUTPUT0 : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal LAST_OUTPUT1 : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal CURRENT_OUTPUT : STD_LOGIC_VECTOR (11 downto 0) := "000000000000";
  signal COUNT : STD_LOGIC := '0';
begin
  CURRENT_INPUT(11 downto 0) <= FILTER_IN;
  process(CLK,RST)
  begin
    if (RST = '1') then
      LAST_INPUT0 <= "000000000000";
      LAST_INPUT1 <= "000000000000";
      CURRENT_OUTPUT <= "000000000000";
    elsif rising_edge(CLK) then
      COUNT <= NOT COUNT;
      -- To keep track of which last_input and last_output to use, this is necessary since the code under each condition executes at the same clock edge.
      if COUNT = '0' then
        LAST_INPUT1 <= CURRENT_INPUT;
        LAST_OUTPUT1 <= CURRENT_OUTPUT;
      else
        LAST_INPUT0 <= CURRENT_INPUT;
        LAST_OUTPUT0 <= CURRENT_OUTPUT;
      end if;
  end process;
end Behavioral;
CURRENT_OUTPUT <= \text{SHR}(("100101001110" \ast \text{CURRENT\_INPUT} + "100101001110" \ast \text{LAST\_INPUT0} + "11101101100111" \ast \text{LAST\_OUTPUT0}), "10000");

elsif COUNT = '1' then
    \text{LAST\_INPUT0} <= \text{CURRENT\_INPUT};
    \text{LAST\_OUTPUT0} <= \text{CURRENT\_OUTPUT};
    \text{CURRENT\_OUTPUT} <= \text{SHR}(("100101001110" \ast \text{CURRENT\_INPUT} + "100101001110" \ast \text{LAST\_INPUT1} + "11101101100111" \ast \text{LAST\_OUTPUT1}), "10000");
end if;
end if;
end process;

\text{FILTER\_OUT} <= \text{CURRENT\_OUTPUT}(11\ \text{downto}\ 0);

end Behavioral;

\textbf{MultPhaseDet} – Phase Detector Component

library IEEE;
use IEEE.STD\_LOGIC\_1164.ALL;
use IEEE.STD\_LOGIC\_ARITH.ALL;
use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity MultPhaseDet is
    Port ( \text{REF} : \text{in} \ \text{STD\_LOGIC\_VECTOR} (11\ \text{downto}\ 0);
    \text{VCO} : \text{in} \ \text{STD\_LOGIC\_VECTOR} (11\ \text{downto}\ 0);
    \text{SOUT} : \text{out} \ \text{STD\_LOGIC\_VECTOR} (11\ \text{downto}\ 0));
end MultPhaseDet;

architecture Behavioral of MultPhaseDet is
    signal \text{OUT\_TEMP} : \text{STD\_LOGIC\_VECTOR} (23\ \text{downto}\ 0);

begin
    \text{OUT\_TEMP} <= \text{VCO} \ast \text{REF};
    \text{SOUT} <= (\text{OUT\_TEMP}(23\ \text{downto}\ 12));
end Behavioral;

\textbf{MAF} – Moving Average Filter (Not used)

library IEEE;
use IEEE.STD\_LOGIC\_1164.ALL;
use IEEE.STD\_LOGIC\_ARITH.ALL;
use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity MAF is
    Port ( \text{CLK} : \text{in} \ \text{STD\_LOGIC};
    \text{RST} : \text{in} \ \text{STD\_LOGIC};
    \text{MAF\_IN} : \text{in} \ \text{STD\_LOGIC\_VECTOR} (11\ \text{downto}\ 0);
    \text{MAF\_OUT} : \text{out} \ \text{STD\_LOGIC\_VECTOR} (11\ \text{downto}\ 0));
end MAF;
architecture Behavioral of MAF is
  type SAMPLE_ARRAY is array (0 to 64) of std_logic_vector (11 downto 0);
signal SAMPLE_ARRAY_CONTENTS : SAMPLE_ARRAY := (others => (others => '0'));
signal SCOUNTER : std_logic_vector (6 downto 0) := (others => '0');
signal WRITE_ADDRESS : std_logic_vector (5 downto 0) := (others => '0');
signal READ_ADDRESS : std_logic_vector (5 downto 0) := (others => '0');
signal LAST_OUTPUT : std_logic_vector(17 downto 0) := (others => '0');
begin
  process(CLK,RST)
  begin
    if rising_edge(CLK) then
      SAMPLE_ARRAY_CONTENTS(conv_integer(unsigned(WRITE_ADDRESS))) <= MAF_IN;
      LAST_OUTPUT <= LAST_OUTPUT + MAF_IN - SAMPLE_ARRAY_CONTENTS(conv_integer(unsigned(WRITE_ADDRESS+1)));
      WRITE_ADDRESS <= WRITE_ADDRESS + 1;
    end if;
  end process;
  MAF_OUT <= "100"*LAST_OUTPUT(17 downto 6)-"100000000000";
end Behavioral;

**CosRom - Cosine Rom**

entity CosRom is
  port( ADDRESS : in STD_LOGIC_VECTOR (15 downto 0);
        COS_DATA : out STD_LOGIC_VECTOR (11 downto 0));
end CosRom;

architecture behavior of CosRom is
  type ROM is array (0 to 65535) of STD_LOGIC_VECTOR (11 downto 0);
  constant COS_ROM : ROM := (
    0 => "011111111111",
    1 => "011111111111",
    2 => "011111111111",
    65533 => "000000000000",
    65534 => "000000000000",
    65535 => "000000000000";
begin
  COS_DATA <= COS_ROM(conv_integer(unsigned(ADDRESS)));
end behavior;

**Constrains file**

NET "CLK" LOC = A8;
NET "D1" LOC = R11;
NET "D2" LOC = P8;
NET "NSYNC" LOC = T12;
NET "CLK_OUT" LOC = T10;
NET "RST" LOC = K12;
NET "SCLK" LOC = R12;
NET "SDATA1" LOC = R13;
NET "SDATA2" LOC = T13;
NET "nCS" LOC = T14;
NET "OUTPUT_FILTER_SWITCH" LOC = N15;
NET "BTN0" LOC = J13;
NET "BTN1" LOC = K14;
NET "BTN2" LOC = K13;
#NET "BTN2" CLOCK_DEDICATED_ROUTE = FALSE;