EE 344 Project Proposal - Mini - AFG

Abhin Shah - 140070013
Karan Chadha - 140070014
Kalpesh Krishna - 140070017

20 January 2016

Contents

1 Abstract

Arbitrary function generators are used to generate baseband analog signals. Besides being a costly equipment, it cannot be interfaced directly with our PC/GNU-Radio. As of now we need to first synthesize the signals, transfer them to the USB stick and then to the AFG before transmission using the IQ Modulator board. To achieve this end goal, we generally do not need the complicated functionality offered by an AFG.

In our project we wish to build a low-cost and portable mini-AFG, which can generate waveforms with a given voltage and frequency parameters. These waveforms would then directly be used as input to IQ modulator board. Effectively the combo of of the mini-AFG and the IQ-modulator board will make the Made-in-IITB "RTL-SDR Transmit Dongle".

2 Project Description

2.1 Background and Motivation

Arbitrary Function Generators (Arbitrary Waveform Generators) are generally complicated and expensive devices which offer a large variety of useful functions. Unlike function generators AFGs can generate any arbitrarily defined waveshape as their output. The waveform is usually defined as a series of "waypoints" (specific voltage targets occurring at specific times along the waveform) and the AFG can either jump to those levels or use any of several methods to interpolate between those levels.
Today, a number of excellent AFGs are available in the market designed by Textronics, Keysight and other companies. Most of the modern AFGs have proprietary software which can interface with the AFG hardware to generate arbitrary waveforms. There are some portable AFGs as well, such as the Hantek DDS3X25. A few AFGs have an in-built Windows operating system as well!
However, these devices suffer from the following problems -

1. **Proprietary** - The software for these products is proprietary and cannot be dissected easily by an end user. This makes it difficult to interface this with open source software like GNURadio.
2. **Expensive** - The complicated hardware is very costly.
3. **Overkill for EE - 340** - A complicated AFG is an overkill for EE - 340. It would be much more convenient to directly supply signal inputs to hardware using GNURadio.

This project was pursued last year in EDL (see the report here) and achieved some success. More specifically,

1. **Communication** - They succeeded in taking samples from a custom GNURadio block and transferring them via USB to a Tiva microcontroller. The Tiva code was written using the Energia interface.
2. **AFG Waves** - They obtained standard waveforms such as sine, square, triangle and sawtooth at certain frequencies less than 30 KHz.

The limitations in their version of the project were -

1. **Wrong Sampling** - While transmitting from GNU-Radio 8 samples were obtained per wave when sampling rate is 10 times the frequency. Ideally 10 samples per period should have been obtained. Besides this, the product only worked at certain discrete frequencies.
2. **Slow Speed** - Since USB 2.0 was used, they could not achieve frequencies higher than 30 KHz.
3. **Irregular Waveform** - There were spikes in the waveform and it is difficult to filter out high frequency glitches.

4. **TIVA** - Since Energia was used, the full functionality of TIVA was not used (such as Port C and JTAG). Also, a number of TIVA boards were lost in the process of building this.

After their project, the WEL Lab RAs have worked on this project. They were able to fix some of the problems, especially # 2 and # 4. They have written new non-Energia TIVA code which can transfer data via Ethernet at higher speeds. However, they are yet to achieve perfect sampling or a regular waveform.

2.2 Project Goal

Very broadly, our project goal is to achieve the following - **A low cost, portable, open source (compatible with GNURadio) mini AFG which can be used for the communication lab at the very least. We hope to produce signals up to 100 KHz.**
The solution to this problem can be divided into three larger components -

1. **Open Source Software** - Generation of digital samples using Python/GNURadio. This software is meant to be compatible with any PC / OS.
2. **Communication** - Communication between PC/GNU-Radio and Microcontroller via USB/Ethernet.
3. **Buffering and DAC** - The microcontroller will store a copy of these samples (buffer) and finally output them onto a DAC, which will convert them to an analog signal.

Our project will have the following advantages:- Very broadly, our project goal is to achieve the following - **A low cost, portable, open source (compatible with GNURadio) mini AFG which can be used for the communication lab at the very least. We hope to produce signals up to 100 KHz.**
The solution to this problem can be divided into three larger components -

1. **Open Source** - The software is completely open source and can be used by anyone to generate signals as long as he has the attached board.
2. **Low Cost Kit** - The board will have a microcontroller and a DAC. Since it is indigenous, it will be a low cost device and easy to interface.
3. **EE - 340** - Using this kit will eliminate the use of an AFG for many of the EE-340 lab experiments. It will be much easier now, since one can directly use GNURadio to do the trick.

2.3 Project Specifications

1. **Customer Specifications**: Low cost, portable, mini-AFG which can be used as a support to the IQ Modulator board by generating arbitrary analog signals (to be supplied to the I and Q channels). This board is powered using freely available, popular, open source software.
2. **Technical Specifications**: A TIVA microcontroller powered board which acts like an interface between GNURadio and the IQ Modulator board. The board will transfer signals with frequencies up to 100 KHz with a short time interval for buffering the signal. Data transfer will be done using Ethernet via a TCP connection. Hence a computer with support for GNURadio and Ethernet is essential for this product to work.

3 Technical Design Description

3.1 Possible Solutions and Design Alternatives

There is no proper documentation for the TIVA C series programming in Code Composer Studio. So we plan to contact the Texas Instruments guys regarding this.

3.2 System-level overview

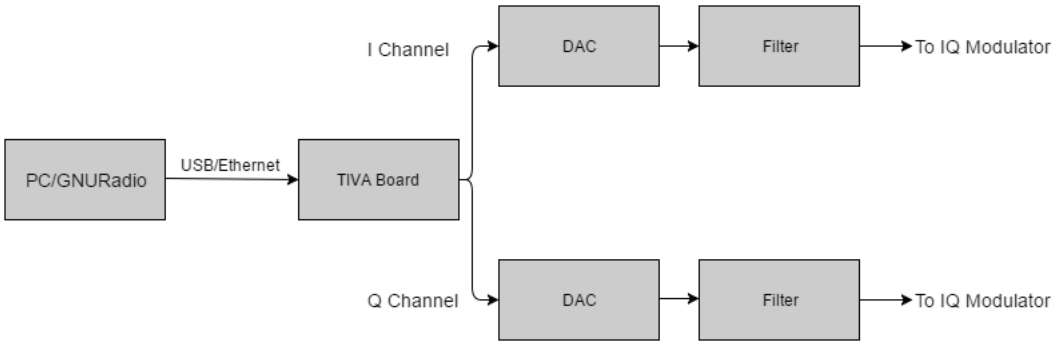

Figure 1: Block Diagram

Description of each block is given below:

1. **PC/GNURadio** - The desired signal can be generated using a python script on the PC or using GNURadio and transferred to the TIVA Board using a USB connection. An ethernet connection can also be used for better speed.
2. **TIVA Board** - The Tiva Board(TM4C1294NCPDT with TI-RTOS) will be used to store the samples coming from the PC and transmitting them at regular time periods to the DAC. The I Channel and Q Channel will be separated after this component.
3. **DAC** - DAC(Digital to Analog Converter) will be used to create an analog signal from the received samples. We will need a fast DAC which works upto the desired frequency range.
4. **Filter** - Finally, we will be using a filter to remove all the unwanted noise components that might have crept in during the processing.

3.3 Performance Validation

We will give a standard input from the computer for instance a sine wave and then check whether we get the same output of same frequency. This would be done initially on the breadboard circuit. Then we would do the same testing on our fabricated PCB.

4 Work Plan

4.1 Work distribution

Given that the project is a continuation of past work, here are a list of tasks -

1. **Literature Survey** - This is an extremely important step, and we need to find out the different existing tools and tutorials that can help us achieve this task. Here are some must-reads-
   - **GNURadio OOT Modules** - The procedure to build Out of Tree Modules in python in GNURadio(see this)
   - **TIVA Programming** - The datasheet of the TIVA microcontroller, guidelines to use Energia, writing C / Assembly programs in the TIVA microcontroller and finally TI-RTOS
   - **DAC integrated circuits** - The datasheet of the TIVA microcontroller, guidelines to use Energia, writing C / Assembly programs in the TIVA microcontroller and finally
   - **Filtering circuits** - This part will come in the latter part of the project, where we must decide the best way to process the analog output to get a continuous waveform.

2. **Previous Work** - We need to get up-to-date with the work done by last year's EDL team and the work done by the WEL RAs'. More specifically,
   - **GNURadio Blocks** - Understand the code written in GNURadio to transfer samples produced by GNURadio to the USB / Ethernet.
   - **TIVA** - Understand the code written by the last year's team in Energia and the WEL RA's for receiving Ethernet samples.
   - **Testing DAC** - Finally, we need to put together last year's work and figure out the issues they were facing. This section includes any filtering circuits after the DAC.
3. **Embedded System Programming** - Once we are completely familiar with the existing code, we hope to use TI-RTOS to start from scratch and correct the code. This will involve setting up a reliable connection between GNURadio and the TIVA microcontroller, saving the samples in a buffer, and transferring them at a constant rate to a DAC. This code will NOT be written in Energia, to achieve higher performance.
4. **Choice of DAC and Filters** - We hope to try a few readily available DACs and compare the performance across them. We might need some additional filters after the DAC, so we will try a few circuits on a breadboard before fixating on a particular design.
5. **PCB Design** - Once the software design and components are finalized, we hope to design a PCB containing the microcontroller, digital to analog converter and additional filters (if any).
6. **Testing and Final Specifications** - Once the PCB is ready, we hope to extensively test it and find out how well it's actually performing. We need to calculate the final specifications of the device and document them for future use.

4.2 Gantt chart

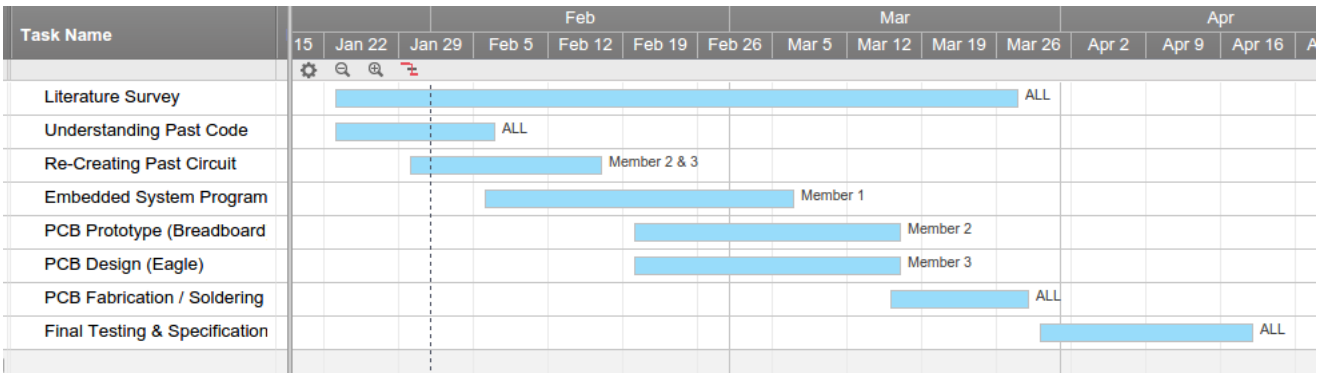

Figure 2: Gantt Chart

See an HTML version here

5 Project Implementation

5.1 Components Needed (Bill of Materials)

1. **Tiva-C LaunchPad** - Available in WEL Lab to the best of our knowledge.
2. **GNURadio** - Free, Open Source Software.
3. **Digital to Analog Converters** - Basic converters are available in the lab. The more advanced converters might need to be purchased (To be finalized in literature survey).
4. **PCB Fabrication Cost**

5.2 Testing Procedures

1. **DSO Testing** - We intend to keep the frequency of the GNURadio signal as the independent variable. We will try to provide different inputs to the microcontroller and see the output on a DSO.
2. **Testing with IQ Modulator** - We intend to interface this with an IQ Modulator board and verify whether the produced signal is being transmitted correctly. This will ensure that our device has a working final application and can help us identify additional problems with the circuit.

6 Deliverables

- **First Evaluation (mid February)** - Get a working prototype, by recreating most of the past work. Besides this, we hope to have understood TIVA programming well.
- **Second Evaluation (mid March)** - Half complete PCB design, final working prototype
- **Final Evaluation (mid April)** - Working PCB chip, interfaced with the IQ Modulator board.