iFPGA - Intermittent Intelligent FPGA Platform

Design Document

sdmay20-38

Client: Henry Duwe

Advisers: Henry Duwe

Team Members/Roles

Justin Sung - Embedded Systems Engineer

Zixuan Guo - Systems Diagram Expert

Jake Meiss - Electrical Engineer

Andrew Vogler - FPGA Design Engineer

Jake Tener - Software Technician

Team Email: sdmay20-38@iastate.edu

Team Website: http://sdmay20-38.sd.ece.iastate.edu

Executive Summary

Development Standards & Practices Used

Hardware and software we will use in this project:

- Powercast P2110b RF Energy Harvester
    - RF to DC Converter
    - Boost Converter
    - Voltage Monitor
- FPGA Circuit Board
    - MicroSemi's Igloo nano AGLN250
- Microphone
    - MEMS microphone
- Capacitor
    - Electrostatic double-layer capacitor
- Regulators
    - Boost and Buck converters to manage voltage to the load
- Neural Network
    - Tensorflow Lite Model with 3 fully connected layers
- Microcontroller
    - TI's MSP430FR5994

Engineering Standards we are applying to our project from the IEEE Code of Ethics:

- Honesty about the functionality and usefulness (#'s 3 & 6)
    - Intellectual integrity for previous work is necessary for eventual published research on the platform
- Emphasis on Teamwork (#'s 7, 8, & 9)
- To make the highest quality product within our capability (#'s 5 & 6)

Summary of Requirements

- Design batteryless PCB-based FPGA system
- FPGA performing computation on low power
- Design application that can be accelerated onto the FPGA
- Accurate neural network predictions
- Data exportable by UART
- Ability to checkpoint progress in a program

    - Intermittent execution on an FPGA platform with frequent power cycling

Applicable Courses from Iowa State University Curriculum

- CPRE488
- CPRE381
- EE330
- CPRE281
- CPRE288

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project:

- Using Libero (an FPGA design tool)
- How to do independent research
- IP-Block research
- Machine Learning
- Python
- Neural Networks
- Sound Parsing and Analysis

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

Henry Duwe - Advised and set concrete goals for the team to work towards.

Narayanan Vishak - Assisting in the HW design development.

Sahu Rohit - Assisting in the SW design and development.

1.2 Problem and Project Statement

Develop a hardware and software solution to intermittently execute a program targeted on a low power FPGA platform that can withstand multiple power cycling events.

As IoT applications become more prolific and integrated into society, the demand for more efficient and powerful devices grow. Addressing these extreme resource requirements and computational demand results in stress on the power supply. Batteries are the predominant source of power for IoT devices. However, batteries are unsustainable and requires maintenance and replacement relatively often in the life span of devices. To confront these inadequate power sources, Self-harvesting power technology shows promise. They require little to no human intervention upon deployment and have significantly longer lifespans compared to batteries. Widespread adoption of this technology can lead to reduced battery usage and mitigate the unattractive qualities that are associated with it.

The iFPGA is a low power designed FPGA platform powered completely by the powercast harvester device to intermittently execute an audio recognition program with resilience against frequent power cycling events. If successful, the iFPGA prototype will uncover novel design solutions to address unreliable power sources, and broaden the feasible areas where IoT can be applied. The prototype will lead to more advanced designs that will build and improve upon any shortcomings of the final prototype. For our research, the iFPGA is targeted at performing audio recognition and classification.

1.3 Operational Environment

The iFPGA will be used within the lab environment, so designing for environmental resilience is a low priority.

1.4 Requirements

Functional Requirements

- Batteryless
    - Power provided by means of RF Energy Harvesting
- Data transmission off-chip
    - UART
- Program execution that can pause and continue while power toggling
    - Checkpointing in the software

Non-Functional Requirements

- Low Power
    - Must be able to detect, perform computations, and transmit data at low power
- Sensor Range
    - Sensor must be capable of capturing nearby audio
- Little to no human intervention after deployment

1.5 Intended Users and Uses

This is the product designed for collecting and analyzing audio based data on an FPGA with an emphasis on low power design. It can be applicable to areas such as IoT and a prototype for developers to expand the study and project.

1.6 Assumptions and Limitations

Assumptions

- This project will be used by Dr. Duwe and his research team.
- The end result of the project is meant to be a prototype design for research, not a market-able project.

Limitations

- Cost of prototype shall not exceed 200 USD
- Battery-less

1.7 Expected End Product and Deliverables

1) A PCB prototype that can accelerate a portion of the audio recognition pipeline that demonstrates the ability to complete the program successfully with frequent power cycling events throughout its execution. Powered by a radio frequency harvester feeding into a capacitor device. A demo to show these capabilities suffice in a successful end product.
2) Comprehensive documentation describing and explaining parts and how to interface with the platform. Justification of our design choices will be included with the end product prototype.

2. Specifications and Analysis

2.1 Proposed Design

Our target solution is an FPGA powered by RF harvesting that can perform simple computations within a low power cycle. We will be using the Powercast RF energy harvester which will provide enough power to run the FPGA and the microcontroller. The FPGA, Microsemi's Igloo Nano, will be able to operate on low power in order for this to be possible. Our FPGA will perform part of our software computation: audio classification; and speed it up. We will do this through accelerating sound analysis onto the FPGA. The FPGA will receive a new sound, generate a spectrogram image of it (a combination of the sound's amplitude, frequency, and change in time) and transmit the data to a connected microcontroller (MSP430). The rest of the software will be uploaded to the MSP430, which will compute an inference of the sound's spectrogram with a loaded neural network, and classify the given sound. Some of the specifics of each technical area of the project are handled below:

Power Management:

- In order to fulfill the requirement for our project to be batteryless, we must begin with energy harvesting. We will be using a 3-watt, 915 MHz wifi transmitter that will be providing the power to our system. This radio frequency signal will be received by a directional patch antenna and then converted to DC power by a Powercast P2110B RF energy harvester. At this point, the converted DC power will be stored in capacitor. The capacitor node will be connected to both a boost converter and a voltage monitor. The voltage monitor will supervise the voltage at the capacitor node, checking for an upper threshold of 1.25V and a lower threshold of 1.02V. The capacitor will begin charging up until the upper voltage threshold is reached, triggering the voltage to be amplified by a boost converter which in turn will provide power to the MSP430 as well as 2 discrete voltage regulators. The MSP430 will control the two regulators that will then provide power to the Igloo Nano FPGA when computations are necessary. The capacitor will be discharging as it provides power to the load (Microcontroller and FPGA) until the capacitor voltage hits the lower threshold of 1.02V in which power to the load is lost and the storage device begin to recharge. Since this process will be continuous, power to the load will be intermittent which must be taken into account in the remainder of the design.

FPGA/Microcontroller Design:

- When powered on by the MSP430, the Igloo Nano will power on and begin computation. Once the MSP430 receives analog audio data, the codec converts the data to be sent to the nano for spectrogram generation. Second, this digital signal will send into the software processor and go through six steps to generate the MFCC: Optimizing the frame, Calculate the periodogram estimate of the power spectrum, Apply the mel filterbank to the power spectra, Sum the energy in each filter, Take the logarithm and DCT for the energies from filter (this step is to let the sound to better represent the human ear), finally, Keep DCT coefficients 2-13 and discard the rest to let the spectrum. The MSP430 will send relevant information to the nano when ready, then it will perform the computation and write back the result to the MSP430. Once the data has returned to the MSP430, it will be sent off to an external server.

- The choice to use the MSP430 as our microcontroller was from a suggestion from Dr. Duwe. Although we could have searched for others, this was his recommendation given his experience in the area and so we thought that to be the best microcontroller for the job. Our FPGA, the Igloo Nano, was chosen after looking at the different products between Lattice and Microsemi. We first came to the conclusion to use a Microsemi product because we verified that we could use Libero, an FPGA design tool, for free. We also found that Lattice was more vague on the product descriptions, making Microsemi a better company for the job. We then narrowed down to the Nano because it is low power and because it still had the most computation power given it’s low power.

Software:

- Since our project entails audio classification, we developed a neural network for our device to use in inference with each new sound. In order to do this, we used UrbanSound 8k’s dataset of 8,732 sounds with 10 unique classifications. Then we sample them into a digital time series and generate spectrograms of each sound (an array of floats indicating its frequency, amplitude, and change in time). With this data, we are able to train a model to know what type of sound correlates to a given label.

- Upon receiving a new sound, our software will sample the sound, then generate its spectrogram, and then use this in inference with our model's weights in order to predict the class of the sound (Fig 2.3). This inference is then going to be transmitted off of the device to a target system.
- When accelerating and uploading the software to the devices, the sound analysis (sampling and MFCC generation) will be our target acceleration for the Igloo Nano. The MSP430 will run the model and inference with the sound data from the Nano.

Output:

- The Igloo Nano and MSP430 will conduct this software on a given segment of sound, then transmit a resultant string including the sound's classification, and a date time object of when the sound was heard.

Top Level Diagram

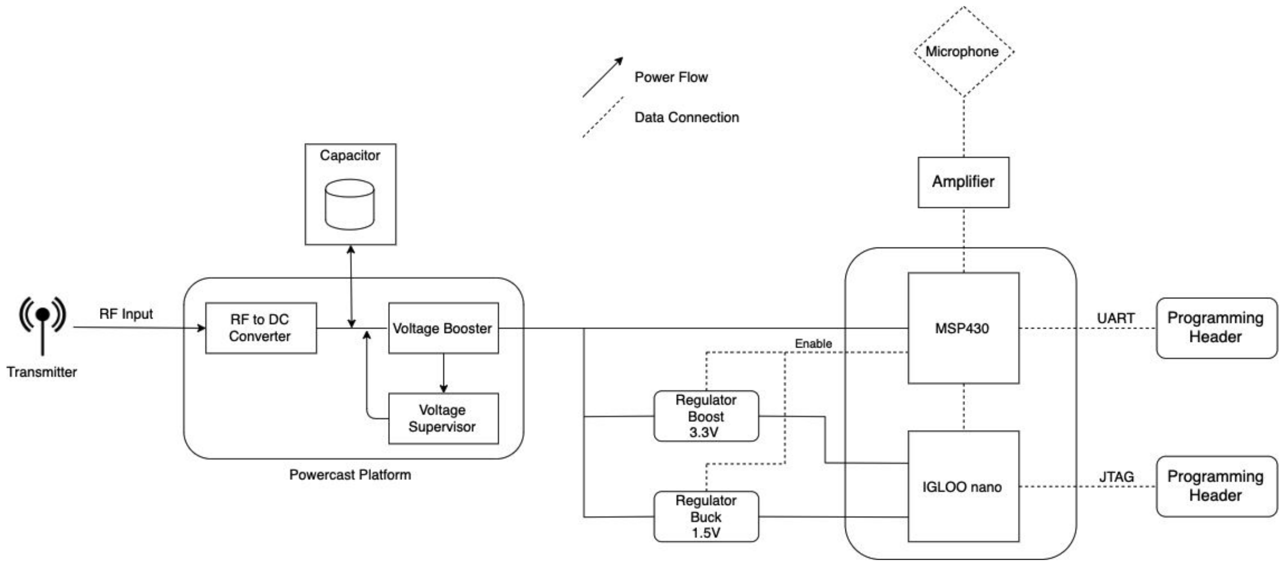

Figure 2.1 Top level design diagram

<u>Embedded System Architecture Diagramt</u>

Figure 2.2 Embedded System Architecture design diagram

Software Process

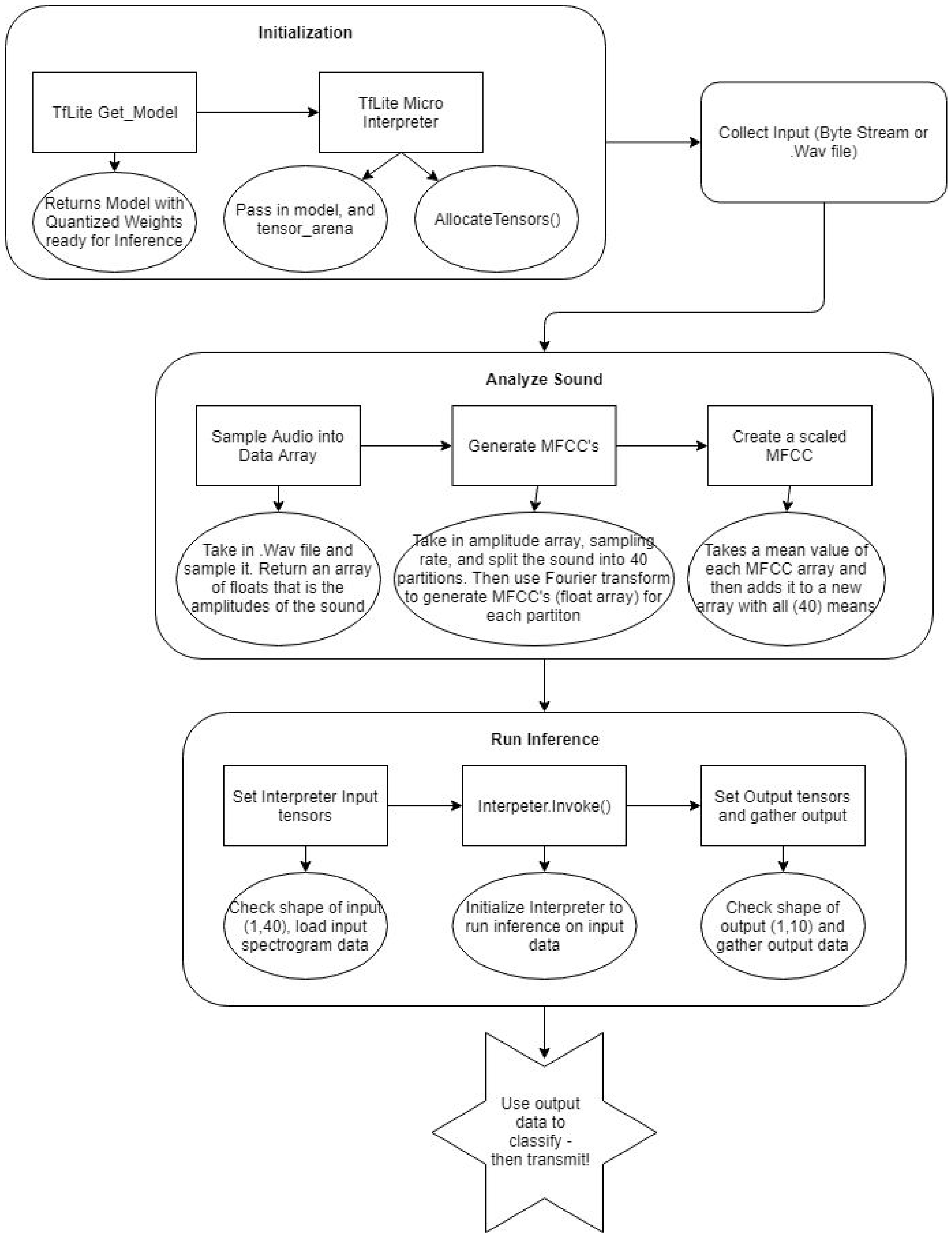
Initialization
TfLite Get_Model
TfLite Micro Interpreter
Returns Model with Quantized Weights ready for Inference
Pass in model, and tensor_arena
AllocateTensors()
Collect Input (Byte Stream or .Wav file)
Analyze Sound
Sample Audio into Data Array
Generate MFCC's
Create a scaled MFCC
Take in .Wav file and sample it. Return an array of floats that is the amplitudes of the sound
Take in amplitude array, sampling rate, and split the sound into 40 partitions. Then use Fourier transform to generate MFCC's (float array) for each partiton
Takes a mean value of each MFCC array and then adds it to a new array with all (40) means
Run Inference
Set Interpreter Input tensors
Interpeter.Invoke()
Set Output tensors and gather output
Check shape of input (1,40), load input spectrogram data
Initialize Interpreter to run inference on input data
Check shape of output (1,10) and gather output data
Use output data to classify - then transmit!

Figure 2.3 Software design diagram

2.2 Design Analysis

Strengths

- Performing computations on an indirect powered platform
- Experience with Test Plans

Weaknesses

- Possibility of insufficient energy harvested resulting in an inability to provide power to the load
- Low power design provides limitations to computational intensity and memory capabilities
- Lack of data security on the platform
- Hardware specifically targeted to perform acceleration on one applications, not flexible for different applications

2.3 Development Process

We are using a waterfall-agile mix for our workflow. Since many of the tasks in the early phase we are walking into with a lot of unknowns, we have to go back and forth with decisions as we learn more about FPGA design. We carry out a new set of tasks each week. Some of the smaller decisions don't have a specific due date (hence the waterfall method) but the bigger decisions, such as choosing an FPGA have a more concrete deadline (more of the agile mindset).

2.4 Design Plan

Once the technology and materials have been secured, we plan to employ a design processes similar to integration by parts and modular design. For both the hardware and software, the first step will be to create and design the modular parts of the project (*e.g.* read-and-write functionality module, hardware accelerator module, FPGA-to-microcontroller interface module, etc.) and verify that they work as intended through rigorous testing. Once each module has been created and verified for functionality, the next step would be to integrate these modules together and verify functionality of the combined modules. Repeating this integration and verification until all modules are combined into a single, fully integrated design.

Modular design will streamline the progress made since errors will be corrected upon discovery rather than allowing them to propagate further into the design and complicating the debugging process. Integration by parts will also facilitate this streamlined progress since logical errors will be caught upon testing the modules and ensures that errors will be corrected at the source and not be allowed to propagate downstream.

3. Statement of Work

3.1 Previous Work And Literature

Embedded System for Acquisition and Enhancement of Audio Signals Paper

This paper describes previous research into an FPGA-based accelerator targeted at audio capture and filtering. It describes the embedded system architecture, signal processing steps and signal enhancement that can be accelerated using hardware. The system architecture was helpful in formulating the eventual hardware diagram. The link to the paper is provided in the appendix.

Efficient Processing of Deep Neural Networks: A Tutorial and Survey Paper

This paper went into the details of current efforts and strategies in accelerating neural network computations on FPGAs and resources-exhaustible hardware platforms. It states that the portion with the highest potential to be hardware accelerated in a neural network pipeline is the multiply-and-accumulate (MAC) operations. These operations have also been shown to result in significant reduction in energy consumption.

Market survey comparisons of components for the platform explained in the previous sections in detail.

Self energy producing devices exists, but none that try to achieve something as costly, both in power and computational power, such as what the iFPGA proposes. No other device in its class can sustain itself solely by radio-frequency derived power while attempting costly computations.

3.2 Technology Considerations

Low power and budget are the two main factors that guided the technology choices made. Since the end product is a prototype to assess the feasibility of such a platform, the budget was restricted and considerations were made in light of it. The low power constraint also affected the technology choices.

3.3 Task Decomposition

- Decide on components that fit the project constraints and functional/non-functional requirements.
    - Perform market survey with limitation considerations.
- Design the hardware platform.
    - Research into existing hardware architecture targeted at audio acceleration.
    - Design and verify portions of the completed hardware architecture and integrate together until the full system has been created and verified.
- Design the software to handle the audio pipeline acceleration and intermittent programming.
    - Design and develop the software to be integrated with the hardware platform.
    - Modify the software to handle intermittent programming.
- Integrate (2) and (3).
- Perform system-level test plan

3.4 Possible Risks And Risk Management

- Platform maximum power consumption
    - The in-rush power and the dynamic power requirements for the platform is bounded by the capacitor storage. A full discharge of the capacitor needs to be able to account for these power demands.
- Intermittent execution handling
    - In the case where the nano is in the middle of its spectrogram computation and the power is insufficient to continue to power it, the intermediate data will be lost and re-execution of the spectrogram must be queued again.
    -
- The Data-pipeline on FPGA and the I/O resources
    - Data go through the Data-pipeline on the FPGA needs to assign each I/O and IP resources, we need to make sure we have a good management on using these resources and optimize the data

pipeline or its structure.

3.5 Project Proposed Milestones and Evaluation Criteria

Key Milestones:

- Component finalization list
- FPGA component integration
- Power analysis and component integration
- Intermittent programing design
- Completed hardware and software integration

3.6 Project Tracking Procedures

Weekly group meetings, weekly advisor meetings, and weekly work period meetings.

3.7 Expected Results and Validation

Deploy the platform in the field and be able to: (1) capture audio from its environment, (2) perform computation on the data, and (3) transmit the results from the computation to an external device. Frequent demonstrations will guarantee that each milestone is functioning as expected and that progress is being made in the right direction and in the right way.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

Task Name	***Start Date***	***End Date***
Semester 1		
All Groups		

Explore previous work	Aug 30th, 2019	Sept 6th, 2019
Embedded System		
Research and choose FPGA	Sept 6th, 2019	Sept 27th, 2019
Prototype FPGA Programs	Sept 27th, 2019	Oct 18th, 2019
IP-Block Research Analysis	Oct 18th, 2019	Nov 15th, 2019
Matrix Multiplication Research and Prototyping	Nov 15th, 2019	Dec 6th, 2019
Software		
Research Possible Applications	Sept 13th, 2019	Sept 27th, 2019
Research Neural Networks and Machine Learning	Sept 27th, 2019	Oct 3rd, 2019
Develop and Train UrbanSound 8k Neural Network (Keras Model)	Oct 3rd, 2019	Oct 17th, 2019
Develop Testing Script for Neural Network	Oct 17th, 2019	Oct 25th, 2019
Quantize UrbanSound 8k Keras Model for Acceleration (Converts to TfLite Model)	Oct 25th, 2019	Nov 8th, 2019
Develop Testing Script for TfLite Model	Nov 8th, 2019	Nov 15th, 2019
Upload TfLite Model onto MSP430 and Run Test Case Inference	Nov 15th, 2019	Current
Research Sound Analysis with Intention to Convert Python Libraries to C Code	Nov 29th, 2019	Current
Power		
Research of RF Energy Harvesting Platform	Sept 6th, 2019	Sept 27th, 2019
FPGA Power Consumption Analysis	Sept 16th, 2019	October 25th, 2019
Power Management Design	Oct 18th, 2019	Current
System Level Architecture Diagram	Nov 25th, 2019	Current

Other:		
Presentation & Design Doc Preparation	Nov 22nd, 2019	Dec 9th, 2019
Semester 2		
Integration		
PCB Design Completion and Order	Jan 13th, 2020	Feb 7th, 2020
Transfer Python Code to MSP430	Jan 13th, 2020	Jan 24th, 2020
Fourier Transformations (FPGA Software)	Jan 13th, 2020	Jan 24th, 2020
Sound Spectrogram Generation	Jan 24th, 2020	Feb 14th, 2020
Data Transfer	Jan 24th, 2020	Feb 14th, 2020
Change to Microphone Input	Feb 14th, 2020	Mar 13th, 2020
Data Retention	Feb 14th, 2020	Mar 13th, 2020
Testing		
Input .wav files for hardware/electrical testing	Feb 14th, 2020	Mar 13th, 2020
Microphone to .wav file testing	Mar 13th, 2020	Apr 3rd, 2020
Refining		
Tweak hardware/electrical based on testing	Mar 13th, 2020	Apr 3rd, 2020
Tweak software based on testing	Apr 3rd, 2020	Apr 17th, 2020
Other		
Buffer Time / Documentation Updating	Apr 17th, 2020	Apr 27th, 2020
Final Project Presentation Preparation	Apr 27th, 2020	May 1st, 2020

This schedule will keep our project on target while also allotting us enough time to sufficiently accomplish each important part. Our goal is to have each piece working well enough to be able to convince our client that we can build a working prototype during semester 2 by the end of the first semester.

4.2 Feasibility Assessment

Our realistic projection of the project will be a custom PCB with the Igloo nano and the MSP430. Given audio data, it will be able to generate a scaled spectrogram, pass it through the neural network to obtain the inference data, and be able to classify the audio data. The spectrogram generation step should be handled by the FPGA while the microcontroller handles the rest.

4.3 Personnel Effort Requirements

Our personal effort will be very high throughout the year. This has been deemed a difficult senior project by our advisor, so it will be tough for us. We will be spending a very high volume of time during the fourth cycle (Build individual pieces, test, and demo) as we get ready to show off our project ideas in order to begin prototyping.

Name	**Overall Contributions to the Project**
Jacob Tener	Neural Network Research, Analysis and Application. Constructed quantized Urban Sound 8k Model. Sound Parsing and Analysis. Applied CNN to microcontroller.
Jake Meiss	Energy harvesting research, power consumption measurements, power management design, system level architecture
Andrew Vogler	FPGA Market Research, Building FPGA example projects, IP-Block Research, Requirements, Schedule Flow
Zixuan Guo	FPGA Testing research and deal with the data pipeline, Hardware design flow
Justin Sung	FPGA Market Research, FPGA Designing and Research, HW/SW design flow

4.4 Other Resource Requirements

- Microsemi's Igloo nano AGLN250v2
- Libero SoC IDE for Microsemi's FPGAs
- TI's MSP430FR5994
- Powercast P2110b RF Energy Harvester

4.5 Financial Requirements

The financial requirements for our project is to keep everything we need to buy under $200.

5. Testing and Implementation

5.1 Interface Specifications

- Libero SoC V11.9
- ModelSim
    - Verifies that the HW works as intended.
- Synplify PRO
    - Verifies the power constraints are satisfied with the HW design
- Code Composer Studio 9.2.0
    - Verifies software implementation on the hardware

5.2 Hardware and software

- Igloo Nano ALGN250 Starter Kit
    - This was a reasonably priced FPGA and had all the specs we were looking for.
- Libero SoC V11.9
    - This is a tool that MicroSemi offers for free to use on all their FPGA products
- Powecast P2110B
    - A 915 MHz RF energy harvester that will be used to provide power to downstream devices
- Lab Instrumentation
    - Includes measurement devices such as multimeters and oscilloscopes used for testing and analysis
- Code Composer Studio 9.2.0

- IDE with microcontroller support allowing us to upload software to MSP430

5.3 Functional/Non-Functional Testing

Test ID:	Test Type:	Grouping:	Given/When/Then	Verified?	Additional Comments
	Functional:	Unit Tests:			
F-U-101			**Given** an RF Harvester Unit, **When** running, **Then** it shall produce the power as specified in our power analysis.		Add % error when better defined...
F-U-102			**Given** our sound classification software (training set and input), **When** the program executes, **Then** the input sound shall be accurately classified into the appropriate category.		See Test No 203 & 204 for Accuracy Percentage
F-U-103			**Given** the results of computation in memory, **Then** the results shall be exportable by an offline UART.		
F-U-104			**Given** a microcontroller, **When** a sound is given as an input to the microcontroller, **Then** the software embedded on the microcontroller shall output a classification for the input sound.		
F-U-105			**Given** an FPGA, **When** a sound classification result sent as an input to the		

			FPGA, **Then** the result shall be stored in the FPGA's memory.		
		Integration Tests:			
F-I-200			**Given** an FPGA and a Microcontroller, **When** the microcontroller has executed it's computation, **Then** the result shall be sent as an input to the FPGA.		
F-I-201			**Given** an RF Harvester and an FPGA, **When** the RF Harvester is running, **Then** the RF Harvester shall apply sufficient power to keep the FPGA running in steady, computation, and data storage states.		
		System Tests:			
F-S-300			**Given** an iFPGA system, **Then** it should not be placed under mild/extreme weather conditions but rather kept at room temperature.		
F-S-301			**Given** an iFPGA system, **When** and IF the FPGA shuts off mid-computation, **Then** the FPGA should not need to be reprogrammed to continue the computation.		
F-S-302			**Given** an iFPGA system, **When** the FPGA is toggled on->off->on, **Then** the		

			computation should continue run as well as data transfer once turned on again.		
		Acceptance Tests:			
F-A-400			**Given** an iFPGA system, **Then** sufficient documentation shall be provided enabling the user to fix and understand the entire system.		
	Non-Functional Tests	Performance Tests:			
NF-P-500			**Given** the system's FPGA (Igloo Nano) , **Then** it should not run above ___ Watts		
NF-P-501			**Given** the system's FPGA (Igloo Nano) , **Then** the voltage applied to the device shall not exceed its maximum threshold (based on it's datasheet).		
NF-P-502			**Given** the system's microcontroller (MSP430) , **Then** the voltage applied to the device shall not exceed its maximum threshold (based on it's datasheet).		
NF-P-503			**Given** our sound classification software, **Then** the Training Accuracy shall be a minimum of 50%.		

NF-P-504			**Given** our sound classification software, **Then** the Testing Accuracy shall be a minimum of 50%.		

5.4 Process

Requirements are verified by the Requirement Verification Matrix specified in section 5.3.

The general flow of testing will follow the diagram below.

Test Plan Flow Diagram

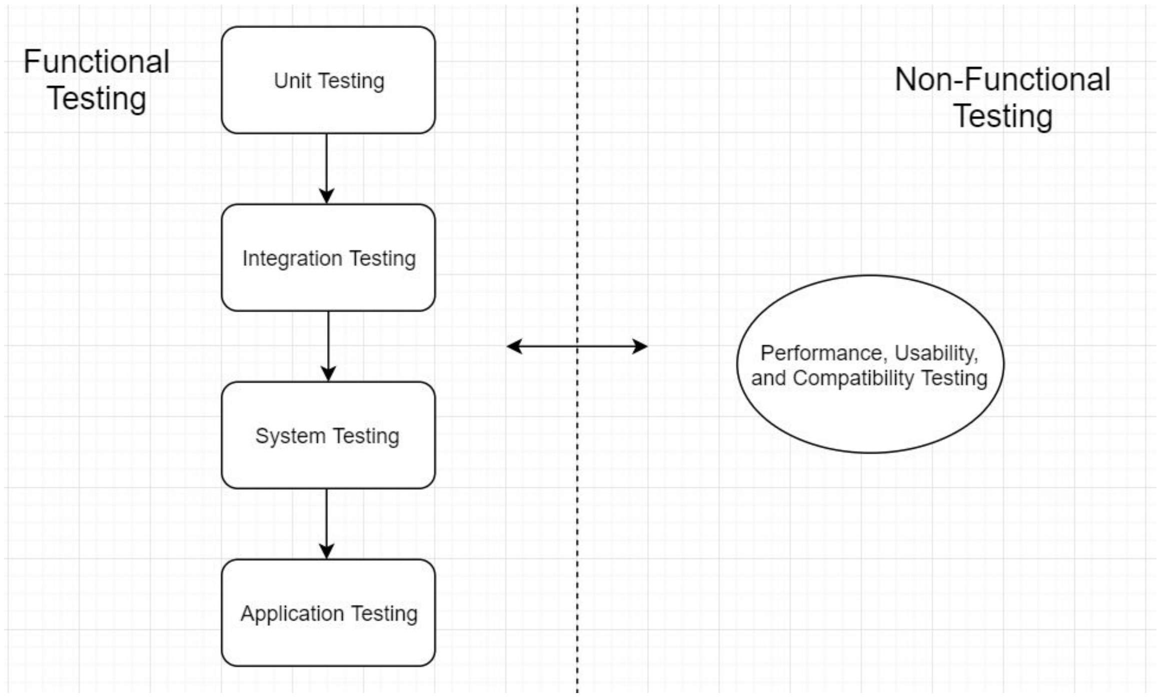

Figure 5.1 Test plan flow diagram

6. Closing Material

6.1 Conclusion

So far, we have gotten through most of the thorough description of the design. Our software can do sound classification, we know which part of the software we are going to accelerate, and we have an understanding of how we are going to attach it to our embedded hardware. On the electrical side, we have done power analysis for the FPGA as well as done the number crunching so we know what electrical components we are going to buy. For the Embedded System, we have picked out a board as well as the ip-blocks we are going to use when programming, and the channels of communication between devices and to external devices. Our goal for this project is to create PCB- based battery-less FPGA platform that can accelerate software computations. Most of this goal is achieved in simply making the physical device and programming it. Making a PCB is one of the tasks to be taken care of in the beginning of the semester. The battery-less capability should be feasible based on our power calculations but it may need some tweaking as we learn more about the software. The acceleration will be feasible as long as we give ourselves enough time to test and make tweaks to our design as necessary. Since we won't know if the part of the software we choose to accelerate will actually accelerate, it is important that we start the next semester running. We are finishing the semester with an in-depth analysis of how to integrate the individual components of the project. The best way to make sure this whole design can actually accelerate a computation is to build it as quickly as possible up front so in the case the software doesn't actually accelerate, we have time to figure out why and make appropriate changes to our design so that it does.

6.2 References

Embedded System for Acquisition and Enhancement of Audio Signals Paper:

https://ieeexplore.ieee.org/document/7763589

Microsemi Libero SoC Documentation:

https://www.microsemi.com/product-directory/libero-soc/5507-libero-soc-v11-9-archive#documents

Efficient Processing of Deep Neural Networks: A Tutorial and Survey
https://arxiv.org/pdf/1703.09039.pdf

6.3 Appendices

N/A