

Open-Source Radio Microcontroller for Fabrication

DESIGN DOCUMENT

Team Number: 27

Client/Adviser: Dr. Henry Duwe

Team Members/Roles:

Ibrahim Shenouda – Analog Architecture Developer

Noah Thompson – Analog Architecture Developer

Nathan Stark – Digital Architecture Developer

Nolan Eastburn – Digital Architecture Developer

Ethan Kono – Security Architecture Developer

Will Custis – Security Architecture Developer

Team Website: <https://sdmay25-27.sd.ece.iastate.edu/>

Revised: 12/4/2024/v1

Executive Summary

Our goal with this project is to create a radio microcontroller unit (MCU) with all created artifacts being open-source. Then, if an individual desired to learn about how our radio MCU works, they could simply read our design and implementation artifacts. This makes our radio MCU unlike all others we found on the market, which are closed source. Closed source designs make it quite difficult for a user to understand how the unit works, which is what makes our design important for users who want to learn how a radio MCU works. Based upon this description, our design must meet the following key requirements:

- All created artifacts (design documents, test document, code, etc.) must be open-source
- All testing and design documentation must be understandable to an individual with a basic knowledge of circuits and embedded systems
- The design must allow for wireless communication using an open standard wireless communication protocol
- The design must be fabricated
- The design must be intuitive to use
- The design must interface to peripherals (servos, sensors, motors, etc.)

As required from our client, we will be using Caravel harness provided by the Efabless corporation to implement our design. The Caravel harness has a user area that can have a custom digital and analog design put in. The Caravel harness and the design we put in the user area will be laid out in the Skywater 130nm PDK, which Efabless has the means to fabricate in. This means we can fabricate the Caravel harness with our custom design in it to implement this MCU. For the design, we split it into two primary sections: digital and analog. The digital section contains all of the digital components that will execute instructions, transfer data, and interface with peripherals. The analog section will contain all the analog circuits required for RF communication. For the digital section, we are going to be implementing a RISC-V core, a DMA controller, an SRAM block, a wishbone crossbar to arbitrate all inter-component wishbone bus traffic, and a digital interface to the analog section. For the analog section, we will be implementing a PLL. These components alone will not be enough to enable RF communication, which violates one of our requirements. There is simply too much work for a single senior design team to implement a fully-functional radio MCU. However, implementing and fabricating these components will provide a good basis for future senior design teams to work on and the artifacts related to these components can still be studied by our users.

Currently, we have a design for the PLL, have researched SRAM generation, RISC-V core generation, have a prototype for the wishbone crossbar, and have created a simple project on the Caravel harness that is currently being fabricated. This provides us a sufficient baseline to begin implementation next semester in CPR E 4920. For our next steps on this project, we plan to take our research and start to implement the digital and analog components into the user area of the Caravel harness. Once we have some prototypes for the components implemented in the user area, we plan on running pre-fabrication tests and running the design through the OpenLane software, which will give us a layout that can be fabricated by Efabless. Based upon this summary, we believe that we are ready to implement our proposed design next semester and send it off for fabrication.

Learning Summary

Development Standards & Practices Used

We considered several engineering standards from IEEE. These include IEEE 802.15.1, 802.15.4, and 1481-2019. IEEE 802.15.1 outlines the requirements for the Bluetooth communication protocol, IEEE 802.15.4 outlines the requirements for the Zigbee communication protocol, and IEEE 1481-2019 outlines standards for verifying integrated circuit designs. While our project implementation will not contain everything necessary to make a functioning radio, the standards will help provide guidance when implementing the system to make sure that it can be easily extended to make a functional radio. In addition to these standards, we will also utilize best practices taught in courses and gained from experience in industry, such as self-checking testbenches to confirm digital designs work as intended. This will make verification easier, making it more likely that we will succeed in making a functional final product.

Summary of Requirements

- The MCU shall be implemented using the Efabless Caravel Harness (constraint).
- The MCU shall implement the Zigbee communication stack.
- The MCU shall contain a radio subsystem.
- The MCU shall contain two independent RISC-V cores to execute user programs.
- The MCU shall contain two DMA engines.
- The MCU shall contain the following standard peripherals:
- processors.
- The MCU shall have software libraries that provide access to hardware functions.
- The MCU shall have a programming interface.
- The RF module shall be able to transmit over the 915MHz ZigBee broadcast band with a 1MHz channel width.
- All Wishbone masters and slaves will use a 50 MHz reference clock which will be shared with the management SoC.
- 4 KiB of SRAM shall be provided for data storage.
- 4 KiB of SRAM shall be provided for instruction memory for the RISC-V processor.
- The MCU shall have a testing interface that enables a user to fully test the unit.
- The digital MCU peripherals shall be tested in a computer simulator to ensure correct behavior prior to fabrication.
- The MCU shall have a test plan to evaluate the characteristics of the system.
- Each piece of the RF module will be tested by sending their outputs to analog GPIO pads to read the waveforms. Some internal signals will be connected to the pads via transmission gates for testing.

- All the artifacts produced throughout the design of the MCU shall be open source (constraint).
- All the documentation shall be intuitive and understandable by individuals with a basic understanding of circuits, digital logic, and MCU usage (constraint).
- The design will fit into a die area of 2.92 mm x 3.52 mm (constraint).

Applicable Courses from Iowa State University Curriculum

The following courses taught at Iowa State are relevant to this project.

Course	Relevance
CPR E 281	Basics of digital logic design and HDLs
CPR E 288	Embedded systems programming in C
CPR E 381	More advanced digital hardware design, CPU architecture
E E 201	Basics of electrical circuits
E E 230	More advanced electrical circuits and systems
E E 330	Integrated circuit design, testing, and layout
E E 465	Digital VLSI
CYB E 331	Cryptography

New Skills/Knowledge acquired that was not taught in courses

Skill	Relevance
PLL design	Key component in analog portion of the design
ASIC fabrication	Design is going to be fabricated, design of ASICs was taught, but not fabrication specific
NGSpice	Simulator used for analog components
OpenLane	Hardening tool to convert HDL code into digital logic layout

Table of Contents

1.	Introduction	13
1.1.	Problem Statement	13
1.2.	Intended Users	13
1.2.1.	ChipForge Co-Curricular	13
1.2.2.	Faculty	13
1.2.3.	Radio Hobbyists	14
2.	Requirements, Constraints, And Standards	14
2.1.	Requirements & Constraints	14
2.1.1.	Functional Requirements	14
2.1.2.	Testing Requirements	15
2.1.3.	Non-Functional Requirements	16
2.1.4.	Resource Requirements	16
2.2.	Engineering Standards	16
2.2.1.	Importance on Engineering Standards	16
2.2.2.	IEEE 802.15.1 - Bluetooth Standard	16
2.2.3.	IEEE 802.15.4 - LR-WPAN Standard (Zigbee)	16
2.2.4.	IEEE 1481-2019: Integrated Circuit (IC) Open Library Architecture (OLA)	16
2.2.5.	Incorporation into Project Design	17
3.	Project Plan	17
3.1.	Project Management/Tracking Procedures	17
3.2.	Task Decomposition	17
3.2.1.	Research and Design	17
3.2.2.	Hardware Implementation	18
3.2.3.	Software Implementation	21
3.2.4.	Testing	21
3.3.	3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria	22
3.3.1.	Fall 24 Milestones	22
3.3.2.	Spring 25 Milestones	22
3.4.	Project Timeline/Schedule	23
3.4.1.	Gantt Chart for Fall 24	24
3.4.2.	Gantt Chart for Spring 25	25

3.5. Risks and Risk Management/Mitigation	26
3.5.1. Unknown Design Tools	26
3.5.2. Limited Die Space	26
3.5.3. High Frequency Components	26
3.6. Personnel Effort Requirements	27
3.7. Other Resource Requirements	29
4. Design	29
4.1. Design Context	29
4.1.1. Broader Context	29
4.1.2. Prior Work/Solutions	30
4.1.3. Technical Complexity	34
4.2. Design Exploration	35
4.2.1. Design Decisions	35
4.2.2. ZigBee Wireless Protocol	35
4.2.3. Reducing Scope	35
4.2.4. AES Encryption	35
4.2.5. PLL Divider	35
4.2.6. PLL Phase Frequency Detector	36
4.2.7. PLL Voltage Controlled Oscillator	36
4.2.8. PLL Charge Pump	36
4.3. Ideation	36
4.3.1. Wireless Protocols	36
4.3.2. Zigbee Wireless Protocol	36
4.3.3. BLE Wireless Protocol	36
4.3.4. Wi-Fi Wireless Protocol	37
4.3.5. LoRa Wireless Protocol	37
4.3.6. NB-IoT Wireless Protocol	37
4.3.7. PLL Divider Designs	37
4.3.8. First Order Delta Sigma	38
4.3.9. Dual Modulus	39
4.3.10. PLL PFD Designs	40
4.3.11. PLL VCO Designs	42

4.3.12.	PLL Charge Pump Designs	43
4.3.13.	Decision-Making and Trade-Off	45
4.3.14.	Wireless Protocols	45
4.3.15.	PLL Divider Designs	46
4.3	Proposed Design	47
4.3.16.	Overview	47
4.3.17.	Detailed Design and Visual(s)	48
4.3.18.	Analog Subsystem	49
4.3.18.1.	DAC	49
4.3.18.2.	PLL: Low Frequency Oscillator	49
4.3.18.3.	PLL: Phase Frequency Detector (PFD)	49
4.3.18.4.	PLL: Charge Pump and Filtering	50
4.3.18.5.	PLL: Voltage Controlled Oscillator (VCO)	50
4.3.18.6.	PLL: Fractional N Divider	50
4.3.18.7.	Modulator	50
4.3.18.8.	Power Amplifier	50
4.3.19.	Digital Subsystem	50
4.3.19.1.	RISC-V Core	50
4.3.19.2.	SRAM	51
4.3.19.3.	Security Acceleration	51
4.3.19.4.	DMA Engine	51
4.3.19.5.	Wishbone Crossbar	51
4.4.	Functionality	51
4.4.1.	Areas of Concern and Development	52
4.5.	Technology Considerations	53
4.6.	Design Analysis	53
5.	Testing	53
5.1.	Unit Testing	53
5.1.1.	Digital	53
5.1.1.1.	Wishbone Crossbar	54
5.1.1.2.	VexRISCV Core	54
5.1.1.3.	DMA Engine	54

5.1.1.4.	TX Sample FIFO	54
5.1.1.5.	Security Acceleration	54
5.1.2.	Analog	54
5.1.2.1.	Phase Frequency Detector (PFD)	54
5.1.2.2.	Charge Pump and Filtering	55
5.1.2.3.	Voltage Controlled Oscillator (VCO)	55
5.1.2.4.	Fractional N Divider	55
5.2.	Interface Testing	55
5.2.1.	Digital	55
5.2.2.	Analog	55
5.2	Integration Testing	55
5.3.	System Testing	56
5.3.1.	Digital	56
5.3.2.	Analog	56
5.3	Regression Testing	56
5.3.3.	Digital	56
5.3.4.	Analog	56
5.4.	Acceptance Testing	56
5.4.1.	Digital	57
5.4.1.1.	Wishbone Crossbar pt. 1	57
5.4.1.2.	VexRISCV Core	57
5.4.1.3.	Security Acceleration	57
5.4.1.4.	DMA Engine	57
5.4.1.5.	DMA Engine/Wishbone Crossbar pt. 2	57
5.4.1.6.	System Testing	57
5.4.2.	Analog	58
5.4.2.1.	PLL Testing Architecture	58
5.4.2.2.	PLL Measurements and Characterizations	58
5.5.	Security Testing	59
5.6.	Results	59
6.	Implementation	59
6.1.	Wishbone Crossbar	59

6.2. Seven-Segment Controller	60
7. Ethics and Professional Responsibility	60
7.1. Areas of Professional Responsibility/Codes of Ethics	60
7.2. Four Principles	62
7.3. Virtues	63
7.3.1. Responsibility	63
7.3.2. Respect	63
7.3.3. Flexibility	63
7.3.4. Individual Virtue Assessment	63
7.3.4.1. Ibram Shenouda	63
7.3.4.2. Noah Thompson	64
7.3.4.3. Nathan Stark	64
7.3.4.4. Nolan Eastburn	64
7.3.4.5. Ethan Kono	65
7.3.4.6. William Custis	65
8. Closing Material	65
8.1. Conclusion	65
8.2. References	66
8.3. Appendices	67
8.3.1. Security Analysis	67
8.3.1.1. Attacks & Vulnerabilities in Microcontrollers	67
8.3.1.2. Countermeasures and Solutions for Vulnerabilities in Microcontrollers	68
8.3.1.3. Attacks & Vulnerabilities in Radio Frequency Modules	69
8.3.1.4. Countermeasures and Solutions for Vulnerabilities in Microcontrollers	70
8.3.1.5. Conclusion	70
9. Team	71
9.1. Team Members	71
9.2. Required Skill Sets for Your Project	71
9.3. Skill Sets covered by the Team	72
9.4. Project Management Style Adopted by the team	72
9.5. Initial Project Management Roles	72
9.6. Team Contract	72

Figures

Figure 1: Research and Design Task Decomposition	18
Figure 2: Hardware Implementation Task Decomposition	20
Figure 3: Software Implementation Task Decomposition.....	21
Figure 4: Testing Task Decomposition.....	22
Figure 5: Fall 24 Gantt Chart.....	24
Figure 6: Spring 25 Gantt Chart	25
Figure 7: First Order Delta Sigma	38
Figure 8: Dual Modulus Divider	39
Figure 9: PLL Diagram.....	40
Figure 10: Phase Frequency Detector.....	40
Figure 11: Phase Frequency States	41
<i>Figure 12: PFD gain without dead zones</i>	<i>41</i>
Figure 13: PFD gain with Dead Zone	42
Figure 14: Dead Zones Jitter	42
Figure 15: Ring Oscillator	43
Figure 16: Current-Starved Ring Oscillator	43
Figure 17: PLL Charge Pump & Loop Filter	44
Figure 18: Charge Pump with added buffers	44
Figure 19: Full Implementation	47
Figure 20: Full Implementation.....	48
Figure 21: PLL Architecture.....	49
Figure 22: Minimum Design Implementation	52
Figure 23: PLL Test Diagram.....	58

Tables

Table 1: Personnel Effort Requirements 27

Table 2: Broader Design Context 30

Table 3: Market Research 31

Table 4: Areas of Professional Responsibility and Codes of Ethics 60

Table 5: Four Ethical Principles..... 62

1. Introduction

1.1. PROBLEM STATEMENT

Many radio microcontroller units (MCUs) exist on the market today; however, their designs are closed source. This makes it difficult for users of radio MCUs such as the ISU ChipForge co-curricular, faculty, and radio hobbyists to learn about radio microcontrollers without reverse engineering the unit due to the designs not being publicly available. It is important that they learn about radio communication now as wireless connections are becoming more common compared to their wired counterparts. To address this, our team is designing an open-source radio MCU. This will provide anyone who desires to learn about how a radio MCU works with all the documentation and implementation details, including how we designed and implemented in our unit. In addition, our implementation can be fabricated through the Efabless Corporation, which allows users to physically use and analyze the radio MCU on top of being able to look at design documents. Having this radio MCU be open-source enables users to make their own modifications to the design, which is something an individual cannot do with closed source units. This further promotes learning and creates opportunities for individuals to be innovative with the base radio MCU design.

1.2. INTENDED USERS

1.2.1. ChipForge Co-Curricular

Chip Forge is a co-curricular at Iowa State University (ISU) that primarily focuses on the design and fabrication of chips through the Efabless Corporation. The members of Chip Forge consist of undergraduate and graduate students at ISU as well as faculty advisors. All members of Chip Forge have an interest in chip fabrication and desire to learn more about it through project development and experimentation. Based upon their interests, the ChipForge members need an open-source radio MCU that they can see the designs for, fabricate the design, and then use the MCU in the lab. This will allow them to dive deep into how a radio frequency (RF) subsystem works on an MCU as well as use this MCU has a component in any projects they work on. This provides a much better learning experience than attempting to reverse-engineer an existing closed source radio MCU, which would prove to be quite difficult and not clear. Finally, we hope that the ChipForge members can take the radio MCU design and build upon it to meet their needs. Since the original design can be fabricated, the ChipForge members can make whatever modifications they see fit for their applications and learn tons about radio MCUs along the way.

1.2.2. Faculty

Some courses at Iowa State University utilize microcontrollers for labs, such as CPR E 288. Once fabricated and tested, faculty could use the open-source MCU for lab assignments in place of existing options. Faculty teaching these courses need a reliable way to teach students about MCUs with good documentation and tooling support. Good documentation and tools are essential since this will likely be used for undergraduate students. Documentation and tools would also reduce the time that faculty need to spend changing existing lab experiments should they adopt the new MCU. There are several potential advantages of the open-source MCU over existing ones. These include the ability to tailor the hardware to course requirements, such as adding additional functionality not commonly implemented in other commercially available products. Additionally, for cybersecurity focused courses, faculty could provide students the opportunity to study in detail

a wireless device from the hardware level in the lab, which would be a valuable learning experience not able to be replicated with closed-source designs.

1.2.3. Radio Hobbyists

Radio hobbyists are usually interested in unique features and documentation/tooling and need MCUs that enable them to communicate with other devices in an easy way without a lot of setup. If one examines commercial MCUs that sell well amongst hobbyists, they are typically low-cost, have good tools and documentation, and have several connectivity features that allow hobbyists to connect them to common devices. Due to the low volume of the production, competing on price will be difficult, but tools and documentation is going to be a central focus of the project, and unique features are possible due to the open-source nature of the project. Furthermore, for hobbyists with a larger budget, our design could serve as a starting point to develop their own MCU tailored to their specific application.

2. Requirements, Constraints, And Standards

2.1. REQUIREMENTS & CONSTRAINTS

2.1.1. Functional Requirements

- The MCU shall be implemented using the Efabless Caravel Harness (constraint).
 - All digital and analog components shall be compatible with the Skylake 130nm technology that the Caravel platform supports (constraint).
- The MCU shall implement the Zigbee communication stack.
 - The MCU shall be able to connect to Zigbee-compatible devices.
 - The MCU shall adhere to the security protocols defined in the Zigbee communication stack.
- The MCU shall contain a radio subsystem.
 - The radio subsystem shall support multiple operating frequencies.
 - The radio subsystem shall support multiple modulation schemes.
 - The radio subsystem shall be able to transmit and receive information to/from other radios.
- The MCU shall contain two independent RISC-V cores to execute user programs.
- The MCU shall contain two DMA engines.
- The MCU shall contain the following standard peripherals:
 - GPIO, UART, I2C, configurable timers, and SPI.
 - The peripherals must have an easy interface for use and configuration (constraint).
- The MCU peripherals shall be memory-mapped and accessible to the RISC-V processors.
- The MCU shall have software libraries that provide access to hardware functions.
- The MCU shall have a programming interface.
 - The programming interface shall be over a serial connection to the MCU.
 - The programming interface shall have a stand-alone application that users run on their PCs to program the MCU.

- o The programming interface shall be intuitive to use, such that an individual with basic MCU programming knowledge can easily program the MCU (constraint).
- The RF module shall be able to transmit over the 915MHz ZigBee broadcast band with a 1MHz channel width.
 - o The carrier signal will be generated by an analog PLL frequency synthesizer to 915MHz using a reference oscillator.
 - o The PLL will be able to step 1MHz between 902 and 928MHz to land at every frequency band.
 - o The input signal from the processor will go through a DAC before being modulated with the carrier signal using quadrature phase shift keying
 - o The modulated signal will be sent to a power amplifier and transmitted from an antenna.
 - o The transmit power shall be between -15dBm to 30dBm (less than 1000mW)
 - o The PLL shall have a settling time of less than or equal to 170us [ATSAMR30M18A](#)
 - o The PLL output will have phase noise less than -91dBm/Hz at a 32 MHz offset.
- All Wishbone masters and slaves will use a 50 MHz reference clock which will be shared with the management SoC.
- 4 KiB of SRAM shall be provided for data storage.
- 4 KiB of SRAM shall be provided for instruction memory for the RISC-V processor.

2.1.2. Testing Requirements

- The MCU shall have a testing interface that enables a user to fully test the unit.
 - o The testing interface should provide electrical connections to the following components in the MCU for testing:
 - ♣ RISC-V cores, DMA engines, peripheral interfaces, PLL
- The digital MCU peripherals shall be tested in a computer simulator to ensure correct behavior prior to fabrication.
- The MCU shall have a test plan to evaluate the characteristics of the system.
 - o The test plan shall provide steps to test the electrical characteristics of the system.
 - ♣ Supply voltage, power consumption, output voltages, slew rates
 - o The test plan shall provide steps to test the MCU peripherals via software.
 - ♣ Transmit and receive for communication peripherals (I2C, SPI, UART).
 - ♣ Correct divider, counter, and comparator behavior for timer.
 - ♣ Correct input/output functionality for GPIO.
- Each piece of the RF module will be tested by sending their outputs to analog GPIO pads to read the waveforms. Some internal signals will be connected to the pads via transmission gates for testing.
 - o The PLL will have part if not all of its loop filter outside the chip for area optimization.
 - o The PLL will have its divider, PFD, reference oscillator, and VCO simulated and characterized using the layout derived from the caravel board.

- ♣ The wave forms can then be used to confirm the components behave as expected.

2.1.3. Non-Functional Requirements

- All the artifacts produced throughout the design of the MCU shall be open source (constraint).
- All the documentation shall be intuitive and understandable by individuals with a basic understanding of circuits, digital logic, and MCU usage (constraint).

2.1.4. Resource Requirements

- The design will fit into a die area of 2.92 mm x 3.52 mm (constraint).

2.2. ENGINEERING STANDARDS

2.2.1. Importance on Engineering Standards

- Engineering standards are important to ensure safety, reliability, and compatibility with other products and protocols. Standards also ensure that the creation and manufacturing of products meets sets of requirements that are deemed necessary to meet several different qualifications.

2.2.2. IEEE 802.15.1 - Bluetooth Standard

- This standard defines the physical and medium access control layers for wireless communication. The Bluetooth standard is used worldwide for low power wireless communication devices across short range radio frequency connectivity, which is important for our project as we are implementing a short range and low radio frequency device through the caravel and Efabless process. Its standards will be important to take into consideration since our design implements an RF (radio frequency) module as well.

2.2.3. IEEE 802.15.4 - LR-WPAN Standard (Zigbee)

- This standard specifies low-rate wireless personal area network operations, network stack, model, and foundational network layers for the physical and MAC networking layers. It is the basis for the ZigBee standard that our project group is utilizing. It also specifies communication for low-rate wireless devices and is intended to provide solid foundations in networking for said wireless devices. This standard applies to our project as the ZigBee protocol outlines the details of operating a ZigBee device so that they are compatible with existing devices. This includes details of the physical and MAC layers of the network stack, which are necessary for reliable communication.

2.2.4. IEEE 1481-2019: Integrated Circuit (IC) Open Library Architecture (OLA)

- This standard provides the analysis for designers to analyze timing, signal integrity, logic behavior, and power consumption across different technologies within a certain accuracy. This standard is relevant to our project since it notes proper timing processes which are essential for the correct operation of the digital components of our microcontroller unit and the RF module.

2.2.5. Incorporation into Project Design

- General principles of power consumption and timing closure will be taken into consideration for our project design and implementation, as it is important for both the MCU and RF module we are designing. The general protocols and principles for Bluetooth and ZigBee will also be utilized and based around because all other low-rate wireless devices follow the same set of requirements and protocols. Incorporating those standards specifically into our project will be necessary across the network stack for connectivity with other devices that use the same protocols and standards.

3. Project Plan

3.1. PROJECT MANAGEMENT/TRACKING PROCEDURES

This project will be managed using a Waterfall approach. This was chosen because we have a small set of deliverables with a lot of interdependence that are required to be completed by a deadline with little flexibility. Waterfall will help make sure that each part of the project (requirements, design, implementation, verification) are all completed in order. Additionally, documentation is a key part of the project, and Waterfall methodology typically results in more comprehensive documentation than Agile, the other approach that was considered. This stems from the fact that Agile focuses on functionality over documentation as defined in the Agile Manifesto. Since the project will span multiple senior design teams, it is more acceptable to trade off functionality for better documentation, since poorly documented functions would likely cause delays and confusion among future design teams.

The project will use Git as the version control system for code created for the project and a repository hosted on the Iowa State Gitlab server. This was chosen since Git provides a lot of functionality for managing different feature development and dealing with conflicts, and the Iowa State Gitlab is already set up for easy collaboration. Issue tracking will be done using Trello, since it is free and allows for custom categories to easily classify issue status. Project communication is currently utilizing a Discord server since it is free, allows different channels to discuss different aspects of the project, and allows for voice and video calls for remote collaboration.

3.2. TASK DECOMPOSITION

The task decomposition for the project was broken into four main categories, with each having several subcategories to further break down tasks so they can be more easily measured. This will allow for actionable tasks that can easily be assigned to team members so everyone knows what they should be doing.

3.2.1. Research and Design

The primary deliverables expected from this portion of the project are the test plan document and the documentation for the components of the system (both hardware and software). The test plan will outline the bring up tests necessary to confirm the chip functions correctly after it is fabricated.

This will need a high level of detail, since these tests will be carried out by people not directly involved with the design process. The documentation for individual components is also important, since the design will be used by a variety of people, including undergraduate students with limited experience, so understanding how each component of the system functions is critical for the chip to be useful.

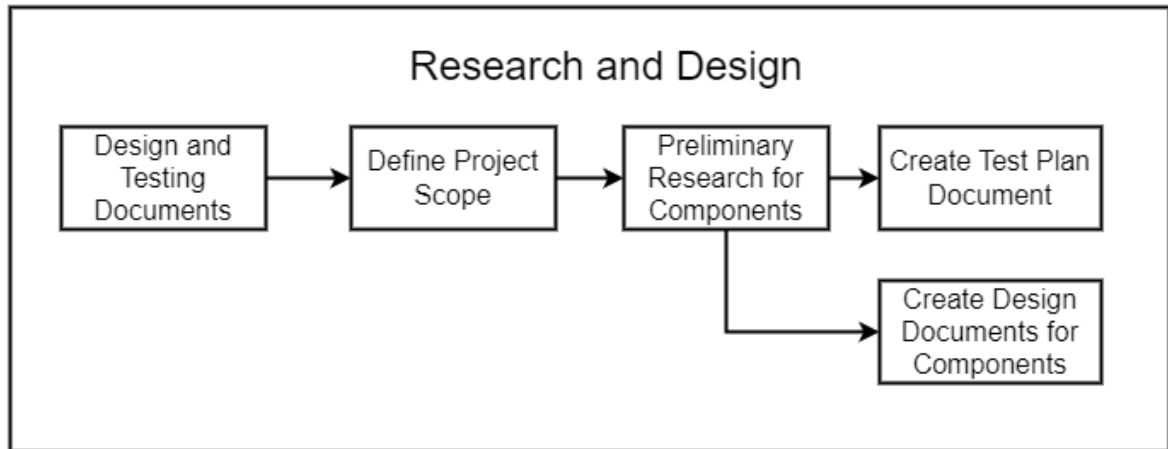


Figure 1: Research and Design Task Decomposition

3.2.2. Hardware Implementation

The hardware implementation is split into two primary categories, analog and digital design. These two will be developed and tested separately and combined during integration to create the final system.

The analog design will primarily consist of a PLL, which in turn is made up of a phase detector, charge pump, low pass filter, voltage-controlled oscillator, and a divider. Each of these components will be designed and tested individually before being integrated together to form the final PLL. In future iterations of the project, the PLL will make up an RF subsystem for transmitting and receiving data.

The digital design will consist of a Wishbone crossbar, RISC-V core, SRAM, DMA controller, security acceleration, and external peripherals. The Wishbone crossbar will arbitrate access to memory mapped peripherals in the design and will be created by hand and may change in the future due to concerns about area usage. The RISC-V core will be generated using a project called VexRISCV that provides optimized cores with good performance along with customization options. This core will be responsible for running user software. The SRAM will be generated using OpenRAM, an open-source tool for creating SRAM layouts. The SRAM will serve primarily as data storage for the RISC-V core, with smaller blocks being used for instruction memory or FIFOs to send/receive data. The DMA controller will help to offload work from the RISC-V core when transferring data to/from various peripherals. This will allow more processing power to be available for user applications. The security acceleration will allow users to encrypt and decrypt data more efficiently than doing it in software by providing hardware capable of performing these operations. Finally, the external peripherals will provide a way for the user application to

communicate with other devices using protocols such as I2C, SPI, or UART. This will let users interact with other devices, such as sensors, nonvolatile storage, or other microprocessors.

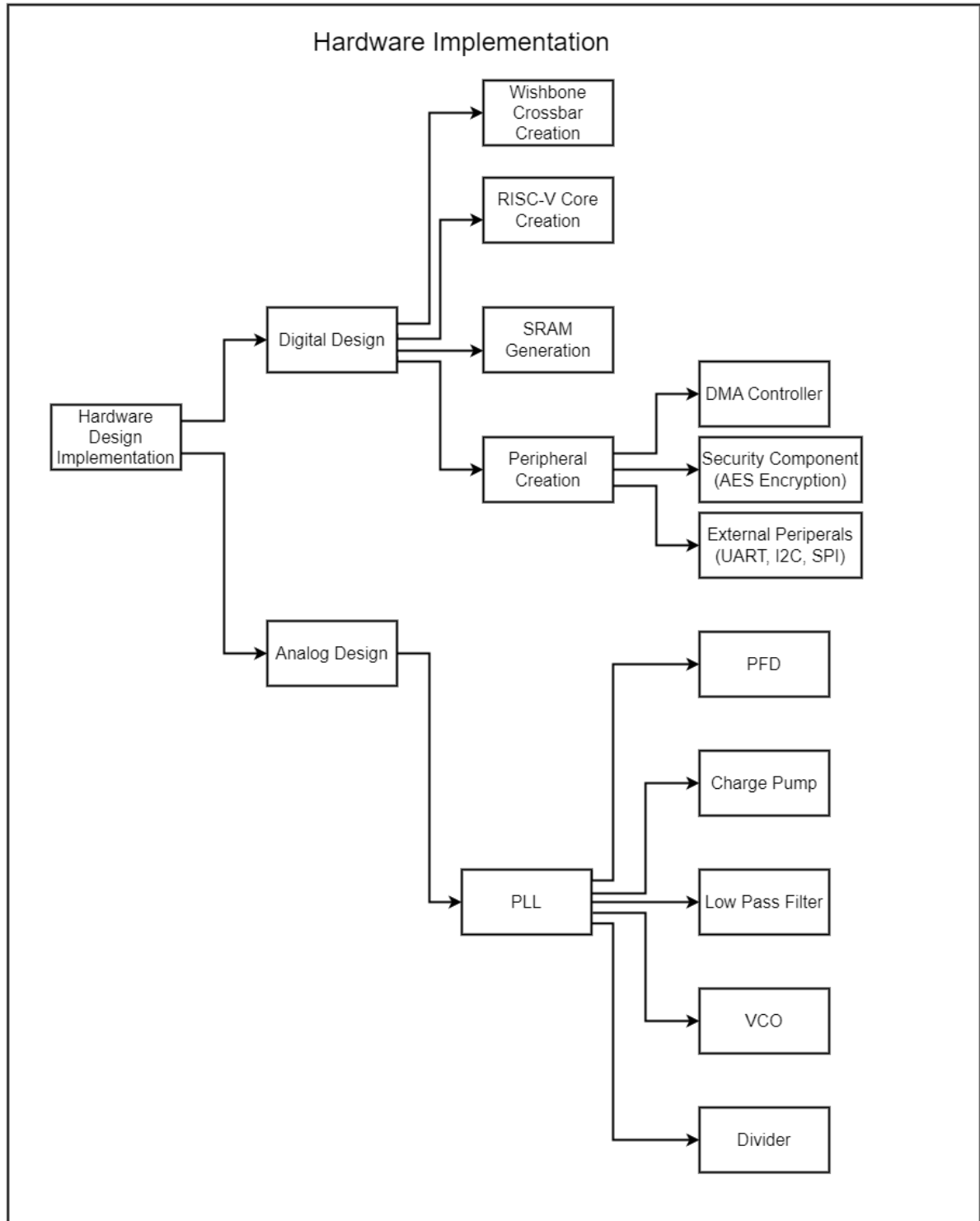


Figure 2: Hardware Implementation Task Decomposition

3.2.3. Software Implementation

There are three primary software deliverables for the project. These are the peripheral firmware, the Zigbee protocol library, and the flashing application. The peripheral firmware will provide easy access to the digital peripherals described in the previous section. This will help make the design more accessible to users, since they will not have to directly interact with memory-mapped registers and can instead use high level functions in their application. The Zigbee protocol library will help make user application development easier by providing functionality that implements lower levels of the Zigbee protocol so users can focus on the high-level transfer of data. Finally, the flashing application will provide an easy way for users to upload their compiled programs to the RISC-V core for execution.

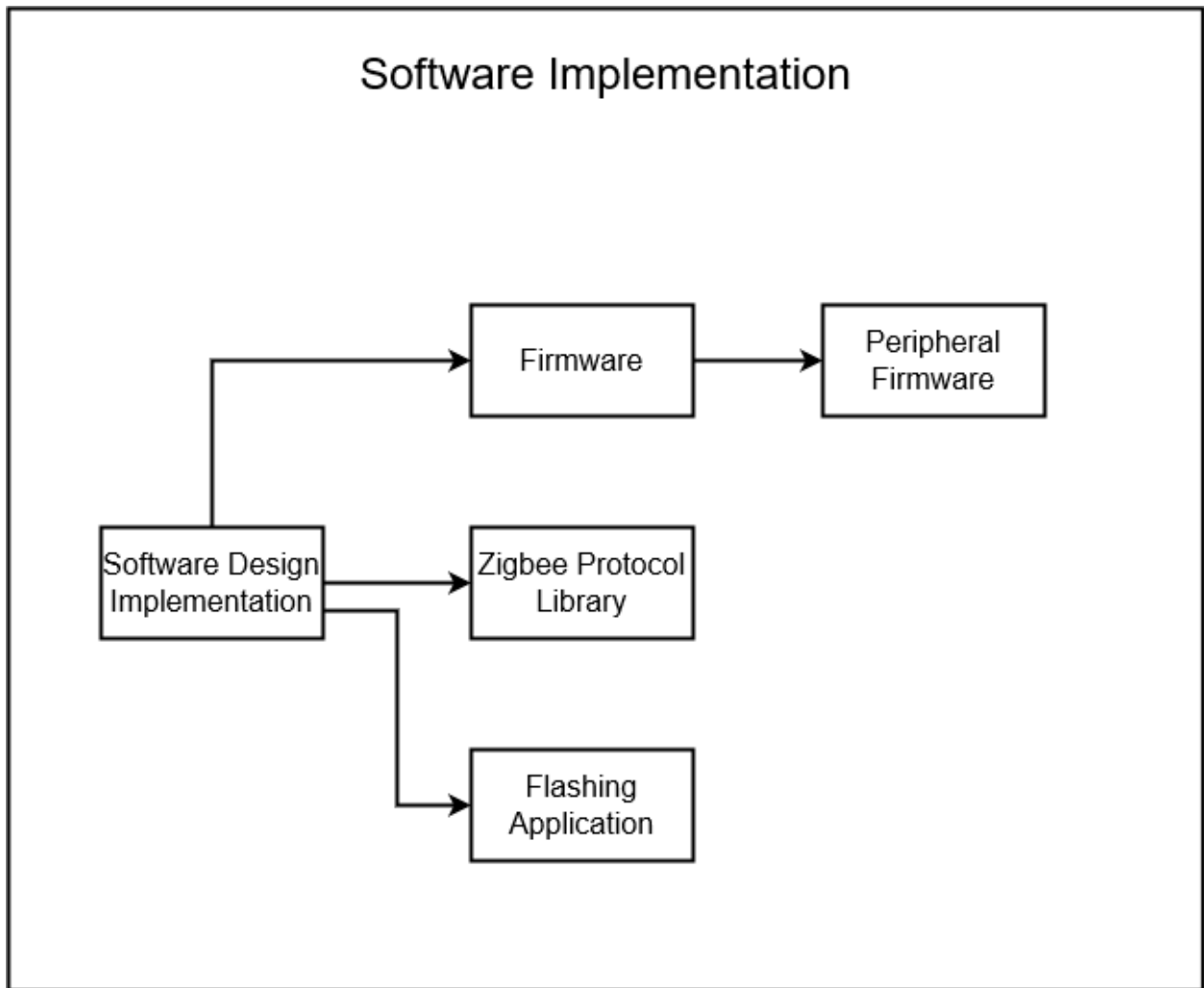


Figure 3: Software Implementation Task Decomposition

3.2.4. Testing

Testing of the various components of the design will be essential to ensuring that the chip functions properly. There are three primary types of testing that will be performed. Verilog

testbenches for the digital peripherals will help to ensure that the blocks work correctly in isolation, which will help to eliminate sources of error during integration. Test C programs for the RISC-V processor will provide a means of system-level testing and will make sure that the system functions as intended. Finally, analog component testing in simulation for the PLL will make sure that the PLL is functioning as expected, and can be tested after fabrication to ensure the fabricated version meets specifications.

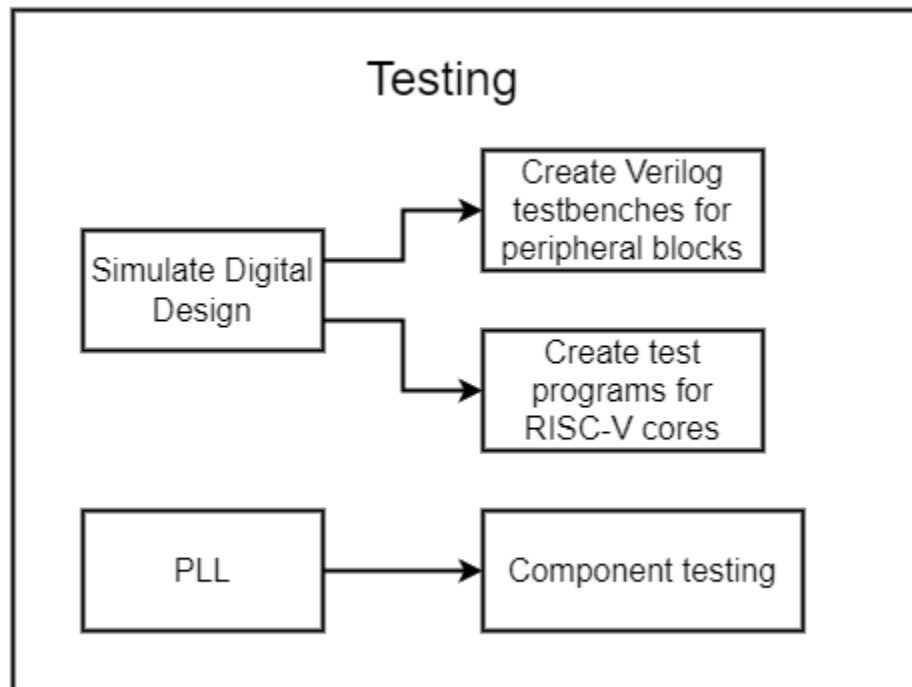


Figure 4: Testing Task Decomposition

3.3.3.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

3.3.1. Fall 24 Milestones

- Design document finished
- Initial hardware designs finalized
 - Specific Implementations of each PLL components
 - Baseline implementations of peripherals
 - RISC-V core design
- Initial test plan
 - Defined expected behavior of each component
 - System to test component behavior

3.3.2. Spring 25 Milestones

- Hardware components created
 - All baseline implementations created and building

- Initial testing complete
 - Hardware components tested as a system in simulation/on FPGA
- Test plan created
 - Test cases for after fabrication to evaluate electrical characteristics and behavior of the device
 - Test cases should cover all peripherals and electrical characteristics

3.4. PROJECT TIMELINE/SCHEDULE

Our full project timeline and schedule are captured in our Gantt charts shown on the next pages. Due to the complexity of our project, many of the tasks will need to be shifted to the end of Fall or Spring of next year. We will only have a single deliverable after the Spring semester, which will contain a partial implementation of our design as well as all of the design and testing documentation.

3.4.2. Gantt Chart for Spring 25

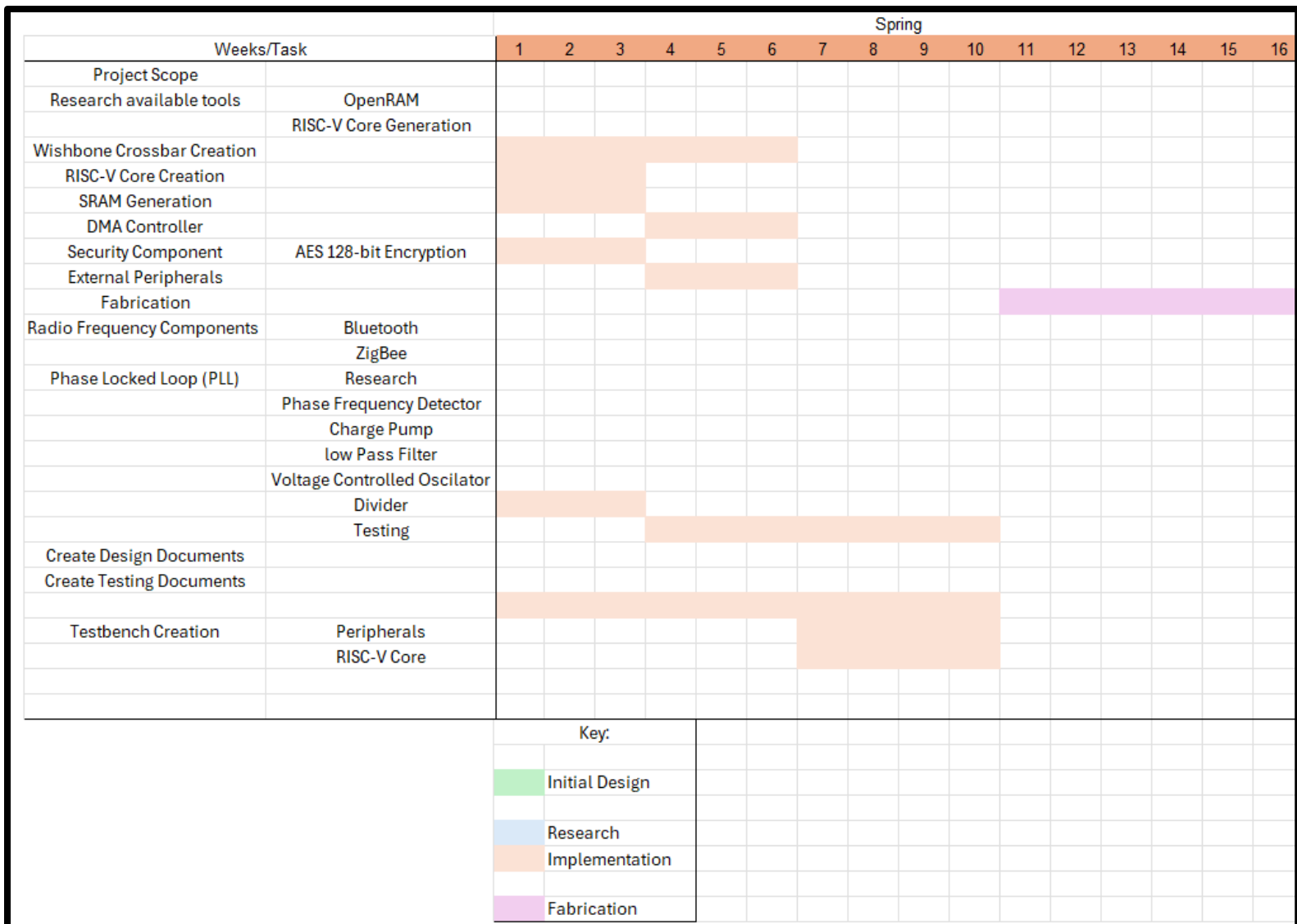


Figure 6: Spring 25 Gantt Chart

3.5. RISKS AND RISK MANAGEMENT/MITIGATION

3.5.1. Unknown Design Tools

Risks

- Our team has no experience with Caravel and the Efabless process.
- Some tools have known issues that may cause delays.

Mitigation

- We are Currently working with members of ISU Chip Forge to learn how to design for the Caravel Board and avoid known issues.
- We will use available IP from the Efabless marketplace when possible.

3.5.2. Limited Die Space

Risks

- The die is 2.92mm x 3.52mm, this means we may not be able to fit everything we want on the chip.
- Challenging to accurately evaluate the amount of space on the die.

Mitigation

- We have created a list of components for a minimum viable product that must be included.
- We can remove lower priority components from the design should die space become an issue.
- If necessary, some components can be accessed via a breakout board.
- Based on our testing, about 300KB of RAM would fill the chip.

3.5.3. High Frequency Components

Risks

- High frequency component behavior is difficult to predict prior to simulation.
- Simulation may reveal that divider creates too much spurring for the PLL to create a usable signal.

Mitigation

- We plan to characterize each individual component through simulation and use that data to characterize the system. This will allow us to quickly pinpoint a component that is not meeting its requirements and revise its design.
- As a backup, we have investigated an alternative divider design that eliminates spurring at the cost of greater noise.

3.6. PERSONNEL EFFORT REQUIREMENTS

Below is an effort analysis for each of our tasks listed in our Gantt chart:

Table 1: Personnel Effort Requirements

Task	Hours (estimate)	Justification
Define Project Scope	80	The scope of this project is quite large and there are many different components we can implement. This will take substantial effort.
Research OpenRAM	15	There exists good documentation for OpenRAM and it utilizes generation scripts. This will only take about a week of effort.
Research RISC-V Core Generation	15	There exists good documentation for the Vex RISC-V generator and the generation is done via scripts. This will only take around a week of effort.
Wishbone Crossbar Creation	50	This component is inherently complicated since it must multiplex many different wishbone interfaces. This will take a large amount of effort.
RISC-V Core Creation	50	Although the core itself will be generated, an interface must be defined for the core such that it can interact with the rest of the design and be programmed. This is complex since we have many components in our design, and we must find a way to supply the core with a program. This will require a large amount of effort.
SRAM Generation	40	The SRAM will be generated, which is not too difficult, but it will need to be integrated into our design and have an interface over the wishbone bus, which makes this complex. This will require a large amount of work.
Create DMA Controller	60	The DMA controller is an inherently complex component that will take a decent amount of research to understand how to create. In addition, we will have to create an interface for this component so it can interact with the rest of our design, which is complex. This will take a substantial amount of work.
Create AES 128-bit Encryption Hardware	40	This hardware will require research to implement, but there is good documentation and known hardware solutions. The main difficulty will be incorporating this into the rest of the design, which is not trivial. This will take a large amount of effort.
Create External Peripherals	80	We are including many peripherals in our final design, each of which have well documented designs, but can be complicated. Since we are implementing many peripherals and need to create an interface for them so they can interact

		with the rest of the design, this will require a substantial amount of effort.
Fabrication	30	The fabrication itself won't be done by us since Efabless will be doing the fabrication, but getting the design ready for fabrication will be challenging. We have to make sure that all of our hardware designs are compatible with the 130nm process, which can be checked using tools provided by Efabless. Depending on how many issues we have when we run the checker tools, this may take a large amount of effort.
Research PLL	20	The PLL is inherently complex, but there exists good documentation of hardware implementations. So, this will require a few weeks of effort.
Research Phase Frequency Detector	20	analog research complexity note The analog design for the RF module will be complicated since many different analog components are required to create each component. Good documentation exists, but many of our team members have not had prior experience designing these analog components. Therefore, it will take a few weeks to fully research each component.
Research Charge Pump	20	See "analog research complexity note"
Research Low-Pass Filter	20	See "analog research complexity note"
Research Voltage Controlled Oscillator	20	See "analog research complexity note"
Research Divider	20	See "analog research complexity note"
Create a Testing Environment for the PLL	50	Since the PLL is a complicated component, creating the testing environment for it will be complicated as well. There are many different waveforms we need to see and analyze and it is not trivial to setup an interface to accomplish this. Therefore, this will take a large amount of effort.
Design Each PLL Component	80	Create each component of the PLL on the Caravel board, then extract its parasitics to be used for simulation.
Simulate Each PLL Component	30	Simulate each component, testing expected behavior, loop characteristics and signal characteristics.
Finalize PLL Design	50	Review and revise the design based on simulations. Determine PLL characteristics to be tested after fabrication.
Create Design Documents	80	The design documents are quite large and will be edited throughout the course. This will take a substantial amount of effort.
Create Testing Documents	80	The testing documents will be quite large and must contain detailed instructions to test our

		many components. This will take a substantial amount of work.
Create Peripheral Testbenches	60	Since we have many peripherals, each of which can be complicated, creating testbenches for the peripherals will take a large amount of work.
Create RISC-V Core Testbenches	60	Due to the complexity of a RISC-V core, creating a solid testbench will be very difficult. There are many different signals we would need to capture and analyze in a meaningful way. This will take a large amount of effort.

3.7. OTHER RESOURCE REQUIREMENTS

We currently require a testing platform to test our designs and a means to fabricate our design. Currently, the ISU co-curricular ChipForge provides a fully-functional testing platform, and we have an agreement with Efabless to fabricate our design on a given date. With these resources and documentation, we can find online and at ISU (mainly through professors, classes, and literature), we have all we need to implement our design or parts of our design due to time constraints.

4. Design

4.1. DESIGN CONTEXT

All radio MCUs currently available are closed source and use proprietary architectures. This makes it more difficult for members of ChipForge, a co-curricular at ISU, and professors doing research to understand their inner workings and tailor them to their specific needs. Our design will focus on being easy to understand as well as open-source, which will allow users to look into the implementation and customize it for their own needs if they have access to the fabrication resources. This will also be supplemented with in-depth documentation, to make the design easy to understand for people with a basic level of knowledge of electrical and computer systems.

4.1.1. Broader Context

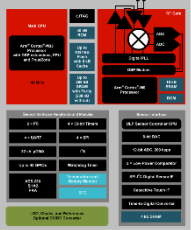

The primary community targeted by our design is college students and professors, since they would likely benefit the most from having access to an open-source radio MCU. Since the product we are creating will likely be lower performance as a tradeoff for being open-source, it is unlikely to impact the microcontroller industry significantly. The biggest societal need being addressed is the need for transparency in microcontroller design, something that is not addressed by many major companies in the space.



Table 2: Broader Design Context

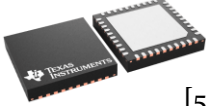
Area	Description	Examples
Public health, safety, and welfare	Our project will benefit people who are college students and professors by providing a learning platform that is capable of performing a variety of tasks often asked of MCUs. It is unlikely to have a significant impact on companies or organizations that already have a foothold in the MCU market.	<ul style="list-style-type: none"> • Extensible design • Documentation allows for students with basic knowledge to use
Global, cultural, and social	The academic community has a strong tradition of sharing resources and information, and the open-source nature of our design helps to continue this tradition by making our work available to be built on by others.	<ul style="list-style-type: none"> • Open-source code • Documentation • Contributing to an ISU co-curricular
Environmental	There are two potential sources of environmental impact for the project, the impact for fabrication and the energy consumption by the device itself after fabrication. Neither of these are likely to be significant, since only a small number will be fabricated and the final design will consume a small amount of power.	<ul style="list-style-type: none"> • Fabrication impact • Power consumption during operation
Economic	Our product will end up being more expensive per unit than existing products for customers, since it is being fabricated in small numbers. This is not something within our control, since small fabrication runs of ASICs are typically expensive regardless of the process or vendor used. However, funding from the university has been secured to cover these costs, so this will not directly impact the primary users at ISU, only other potential users.	<ul style="list-style-type: none"> • Higher unit cost • Fabrication cost covered by university

4.1.2. Prior Work/Solutions

Table 3: Market Research

Product Services and Design What is the product?	Unique Value Proposition What makes this product unique?	Product Advantages What are the things that provide a leg up?	Product Disadvantages Where might drawbacks exist?	User Pros What do users like about the product?	User Cons What do users NOT like about the product?
 <p>TI CC1352P [1]</p>	<ul style="list-style-type: none"> - Thread, Zigbee, Matter - Bluetooth - Low power consumption 	<ul style="list-style-type: none"> - Low power consumption while supporting Bluetooth - Supports multiple radio protocols 	<ul style="list-style-type: none"> - Only use 2.4GHz radio frequencies which can be more than necessary for protocols like Zigbee 	<ul style="list-style-type: none"> - Part of simple link system with common simple development environment 	<ul style="list-style-type: none"> - Not open source - Designed for general use - Specific applications can do better
 <p>Espressif ESP32 [2]</p>	<ul style="list-style-type: none"> - Bluetooth & WIFI - Microcontroller - Multicore - Low cost 	<ul style="list-style-type: none"> - Wide variety of users - Used in low power IoT products - ESP-NOW Protocol 	<ul style="list-style-type: none"> - Proprietary CPU architecture, limited support - Documentation can be lacking - 512kB memory: not enough for some memory-heavy applications 	<ul style="list-style-type: none"> - Very affordable - Easy to use libraries - Various resources - Collaborative online community for hobbyists - Can program with Arduino IDE 	<ul style="list-style-type: none"> - Low Range - Only - High power consumption for some peripherals

 <p>[3]</p> <p>Raspberry Pi Pico W</p>	<ul style="list-style-type: none"> - Wi-Fi and Bluetooth 5.2 support - PIO state machines - MicroPython support - Bootloader allows software to be loaded without a special programmer 	<ul style="list-style-type: none"> - PIO state machines allow for flexible peripheral allocation - Dual core to allow one core to handle radio and one to handle application 	<p>264 kB memory may not be enough for some applications</p>	<ul style="list-style-type: none"> - Cheap - Good documentation - No need for extra tools for programming 	<ul style="list-style-type: none"> - C/C++ SDK setup can be painful relative to other MCUs
<p>STM32WB09KEV6 TR</p>  <p>[4]</p>	<ul style="list-style-type: none"> - Has multiple run modes, notably, for low power - ARM Mo processor (well-known architecture) - Multiple supported frequency bands - Designed for ultra-low power 	<ul style="list-style-type: none"> - This microcontroller is designed for ultra-low power applications, so it is viable to use when power consumption is a concern - Small form-factor 	<ul style="list-style-type: none"> - Has an ARM Mo, which is not super powerful - When in ultra-low power mode, functionality is heavily limited. 	<ul style="list-style-type: none"> - Not generally purchased by individuals for hobbyist use 	<ul style="list-style-type: none"> - Not generally purchased by individuals for hobbyist use

	consumption				
 <p>[5]</p> <p>TI CC2540F256RHAR</p>	<ul style="list-style-type: none"> - Low energy SoC - True Single-Chip BLE Solution that can run both application and BLE protocol stack 	<ul style="list-style-type: none"> - Low-power - True system-on-chip (SoC) - Cost-effective - In-system programmable flash memory 	<ul style="list-style-type: none"> - Limited to BLE only - Less powerful processor compared to competitors - Cannot support complex, higher power applications 	<ul style="list-style-type: none"> - Cost effective - Low power consumption - BLE implementations/support 	<ul style="list-style-type: none"> - Strictly limited to BLE implementations - Not suitable for complex tasks.

The primary advantage our design will have over these others on the market will be documentation and its open-source nature. Due to process constraints, it is unlikely that our design will be capable of competing with many of these designs on performance, since they mostly use newer processes than the one we have available, which is around two decades old. However, since ChipForge typically deals with smaller programs for learning purposes, this is an acceptable tradeoff, since they would be unlikely to use the extra performance of the other solutions.

4.1.3. Technical Complexity

The project consists of three main engineering design systems, analog/RF, digital, and cyber security. Each system consists of multiple subsystems each containing one or more components. We are implementing all of these components using our understanding of complex circuit design principles, computer architecture hierarchies, as well as hardware and software security standards. For example, the PLL is a closed loop system, in which its output depends on the individual gain and noise levels of each subcomponent as well as the closed loop transfer function. The Wishbone crossbar uses interconnected circuitry to enable communication between N number of masters to M number of slaves. The cyber security component design depends on strong foundations of the AES encryption algorithms. Implementing a fully functioning radio microcontroller from scratch mandates collaboration of multiple cross functioning engineering teams and years of design and testing. This is why our team will be scoping the project and implementing some of the radio microcontroller systems. This is why our limited implementation does not enable transmit or receive; however, it layers the foundations for other senior design teams to further contribute to this project. Our limited implementation consists of the following systems:

1. Analog/RF subsystem:
 - a. Phase Locked Loop:
 - i. Phase frequency detector (PFD)
 - ii. Charge Pump (CP)
 - iii. Loop filter
 - iv. Voltage controlled oscillator (VCO)
 - v. Divider
 - b. Digital to analog converter
2. Digital subsystem:
 - a. RISC-V processor
 - b. Wishbone bus crossbar
 - c. SRAM
 - d. DMA
 - e. Peripherals
 - i. Timers
 - ii. GPIO
 - iii. I2C
 - iv. SPI
3. Cyber security subsystem:
 - a. AES 128bit encryption

4.2. DESIGN EXPLORATION

4.2.1. Design Decisions

3 Design Decisions (ZigBee over Bluetooth, which portions of the whole design we will complete, AES encryption)

4.2.2. ZigBee Wireless Protocol

One of the first major decisions our group had to make was what wireless communication protocol we wanted to implement in our design. After researching what protocols some other radio microcontrollers support, we found that Bluetooth and ZigBee were some of the most common with many market options supporting both. After discussion with Professor Duwe and members of the Chip Forge club we concluded that ZigBee would be best for our design. Bluetooth could eventually be added later but we wanted to focus on getting one wireless communication protocol working first. We made this decision due to ZigBee using sub-Gigahertz radio frequencies which will reduce our hardware requirements.

4.2.3. Reducing Scope

When beginning this project with Professor Duwe we did not know the full scope of what would be required to design a radio microcontroller. So, our first couple weeks of work were researching just that. We quickly came to realize that the scope of this project was much larger than we originally anticipated, and that this project would take multiple teams across a couple of years. We then had to decide which components of our complete design we would try to complete before passing the project on to the next group. We chose to start with implementing one of two RISC-V cores, a wishbone bus crossbar, SRAM, one DMA Engine, the flash controller, security acceleration, digital to analog converter, phase-locked loop, reference oscillator, modulator, and some smaller timing components. While we will not have a fully functioning radio microcontroller with just these components, we aim to be able to create a product able to send data wirelessly by the end of our project.

4.2.4. AES Encryption

We knew from the start that we wanted to include some amount of security on our microcontroller, not only to secure it, but also so it can be used to teach about wireless security. Encrypting data is essential to wireless communication as radio waves are very easy to intercept and read. By making the data unreadable without a key we largely negate this risk. There are many encryption methods that could be used but we decided on using AES as it is what is recommended by the ZigBee protocol as well as being one of the most popular and secure encryption methods.

4.2.5. PLL Divider

The design for the PLL divider was more complicated than we initially thought. Because the division is required to operate at 930MHz, a simple programable flip-flop divider won't work. Our research resulted in two potential designs, fractional N, implemented as a first order sigma delta divider, or integer N, implemented as a dual modulus divider. We decided to use the fractional N divider because it creates less in-loop noise.

4.2.6. PLL Phase Frequency Detector

The PFD design is the simplest subcomponent of the PLL yet there exist different design implementations. The widely common circuit topology of the PFD uses the XOR, JK flops, or D-flip flops. The XOR implementation is sensitive to the input duty cycle and will lock with phase error if both inputs are not exactly 50% duty cycle. Unlike the exclusive OR implementation, the DFF implementation is not sensitive to the input duty cycle. It is also easier to implement and less noisy than the JK flip-flop implementation. This is the main reason that it has become the industry standard in PFD design.

4.2.7. PLL Voltage Controlled Oscillator

Various VCO design circuits are used in PLLs. The main two circuit families are ring and LC oscillators. The ring oscillator is generally easier to implement, requires less area, and has a more tunable frequency range than the LC oscillator but has more phase noise. This is the main reason we decided to implement a current-starved ring oscillator VCO.

4.2.8. PLL Charge Pump

The charge pump circuit could be implemented with 4 transistors. However, because it controls the voltage on the VCO, any mismatching in current source/sink will cause the VCO to set a wrong frequency, causing oscillation. The focus of any extra circuitry will be used to charge sharing, and other signal transients.

4.3. IDEATION

The following describes the presented solutions for our wireless protocol and our PLL divider. Each solution is given a summary explaining its functionality and appeal.

4.3.1. Wireless Protocols

With various wireless protocols, we identified potential options based off their widespread use and documentation availability. The ones we found that were most commonly used across a majority of the microcontrollers on the market utilized some form of Wi-Fi, Bluetooth, or Zigbee protocols. The five major options we considered were Zigbee, BLE (Bluetooth Low-Energy), Wi-Fi (Wireless LAN), LoRa, and NB-IoT.

4.3.2. Zigbee Wireless Protocol

Zigbee was one of the first options we considered, since it allows for lower frequency rates than other wireless protocols. Specifically, it supports both 2.4 GHz and 915 MHz frequencies, which was more applicable to our design due to the nature of the RF module we are implementing. Zigbee also has relevant documentation regarding the IEEE standard 802.15.4, (since the protocol itself is built on-top of this standard) which specifies the technical standards of low-rate wireless personal area networks.

4.3.3. BLE Wireless Protocol

BLE or Bluetooth Low Energy was another option we highly considered working with, because of its low power consumption, as well as its ability to support 2.4 GHz and the various amounts of documentation available. The major downside was that BLE does not support lower frequencies,

and although it supports versions of Bluetooth up to Bluetooth 5.0, the lack of low frequency support is ultimately why we choose not to proceed with BLE.

4.3.4. Wi-Fi Wireless Protocol

Wi-Fi as an option was considered, as it operates on 2.4 GHz and 5 GHz frequencies and supports various IEEE standards 802.11a, 802.11b, 802.11ac, and more, proving that there are tons of documentations on implementations, protocols, and specifications for wireless communications, making implementation easier. It is also the most widely used wireless communication protocol. Similar to why BLE was not considered, the lack of low-rate frequency support, and ultimately did not fit our projects goals and specifications that utilize an RF module.

4.3.5. LoRa Wireless Protocol

The LoRa protocol utilizes radio communication techniques to achieve low power and long-range communication. It also supports sub-gigahertz radio frequency bands and enables long-range transmissions. While it contains a lot of the components we would like in our design, it ultimately doesn't work well in data transmission with low data rates, and slow data transmission. It also would require its own network for deployment, which would limit its accessibility and ease of use for our intended users.

4.3.6. NB-IoT Wireless Protocol

The NB-IoT protocol is a low-power wide-area network meant to connect devices that have low bandwidth across long distances. Its low power consumption, ability to connect across long distances, and low frequency support makes it enticing as a design choice, but has limited data rates, latency issues, and coverage limitations (specifically indoors) despite being able to connect across long distances. Its implementation also highly depends on existing network infrastructure. As our microcontroller is likely being utilized in environments with heavy infrastructure such as large buildings and labs throughout the university, it didn't make sense to implement since it wouldn't fit our intended user's needs.

4.3.7. PLL Divider Designs

Of the various options for frequency dividers, we selected the first order delta sigma and the dual modulus dividers because of their ability to operate in the GHz range, their common use in frequency synthesizers, and their relatively simple design.

4.3.8. First Order Delta Sigma

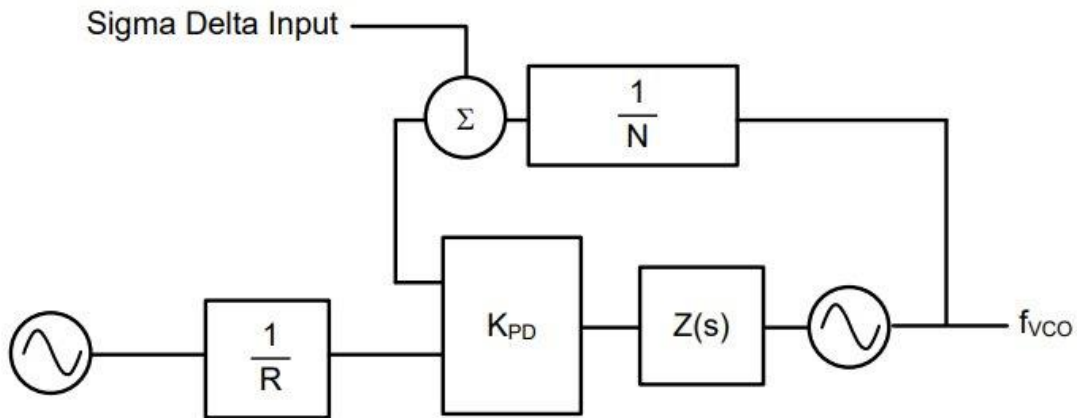


Figure 7: First Order Delta Sigma

The first order delta sigma design uses fractional division. This allows the reference frequency to be higher than the channel width. In our design, it means we can use a 10MHz reference frequency instead of a 1MHz one, which results in a maximum division of 92.8 instead of 928. This reduction in division is what allows this divider to operate at higher frequencies. The fractional division is accomplished by alternating integer division values from a programmable dual modulus divider in a sequence (Sigma Delta Input) that averages to the desired value. For example, dividing by 92.8 would be achieved by dividing by 93 for 4 cycles and 92 for one cycle. This oscillation results in an average division of 92.8, smoothed out by the loop filter.

4.3.9. Dual Modulus

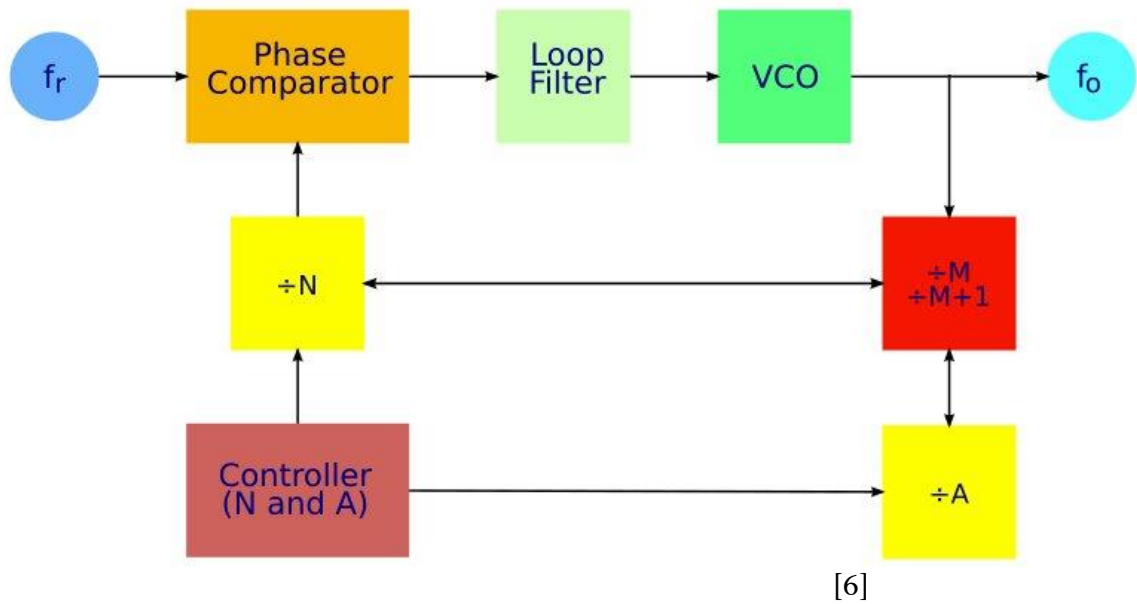
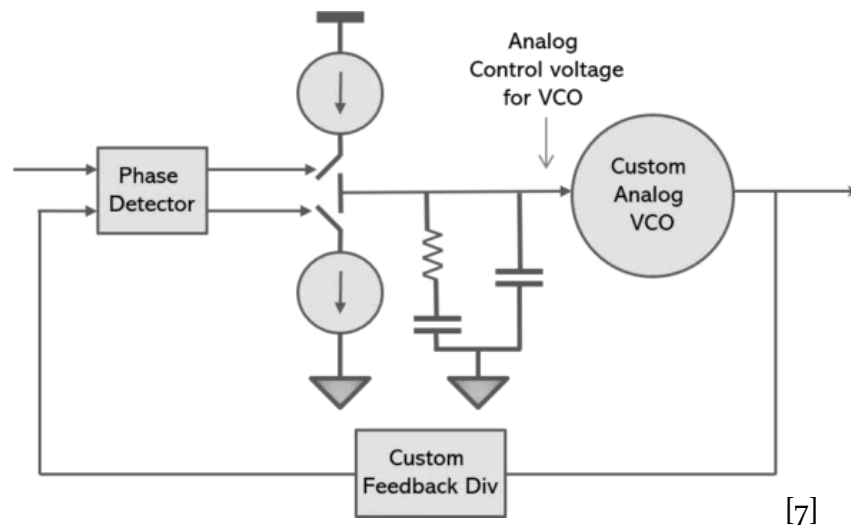


Figure 8: Dual Modulus Divider

The dual modulus design uses integer division which limits the reference frequency to the channel width, 1MHz. To achieve high frequency division, this design uses a fixed prescaler at m and $m + 1$. This prescaler divides the frequency down to a functional operating frequency for a simple programmable flip-flop divider, N , to create the total divisor. Just a prescaler of $M+1$ and N results in a division ratio of $N(M+1)$ because M is fixed the divider would only be create a divisor as a multiple of $M+1$. To allow the divider to create every division ratio instead A second programmable divider A is added where $A < N$. A and N count down at the same time and when $A = 0$ the prescaler is change from $M+1$ to M . This results in a division ratio of $A(M + 1) + (N - A)M$. If $N \geq M$, then every division ratio is possible.

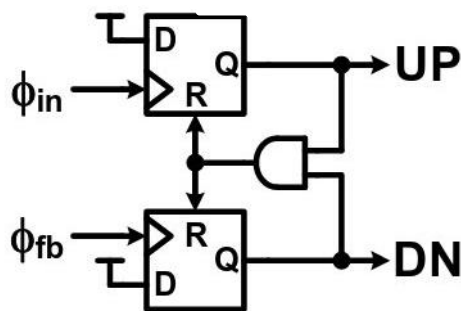
4.3.10. PLL PFD Designs



[7]

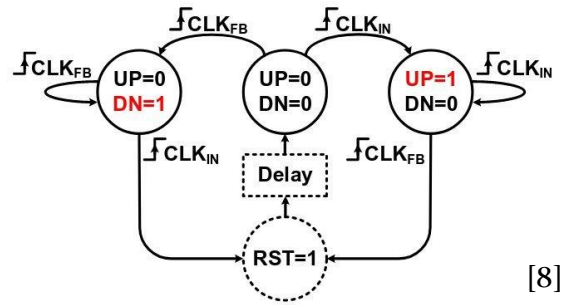
Figure 9: PLL Diagram

Phase frequency detector is the first component of the PLL. It detects the phase error between the reference oscillator signal and the feedback signal from the divider. There are multiple circuit topologies that could be used to make the PFD. However, a D flip-flop implementation been the standard practice in the industry because it is simple to implement, does not depend on the duty cycle, and has a constant gain of $1/2\pi$.



[8]

Figure 10: Phase Frequency Detector

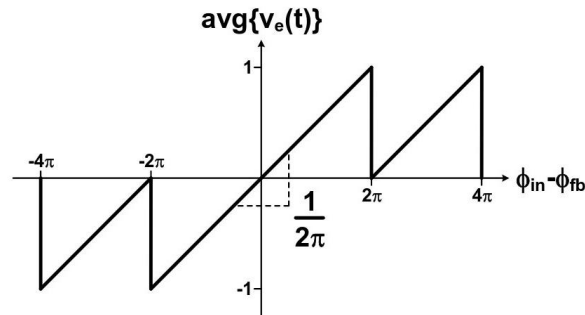


[8]

Figure 11: Phase Frequency States

The UP and DN signals from the PFD will later be used to drive a charge pump, which sources or sinks current to the loop filter, thus dynamically changing the voltage input of the VCO. The UP signal is reflected by a positive 1 because it sources current and the DN signal is reflected by a negative 1 because it is responsible of sinking current from the loop filter.

Because the charge pump and PFD combined will not be able to switch slower than a 150 picoseconds. This causes a defective region formally known as the dead zone, which can cause lower gain and increased jitter. To avoid dead zones, a delay has been inserted between the RST signal of the DFFs and the AND gate. This will ensure that both UP and DN signals will stay high for transistors to switch properly. As both UP and DN are high, the net current (ICP) passing through the loop filter will be zero, causing no net change to the output frequency.



[8]

Figure 12: PFD gain without dead zones

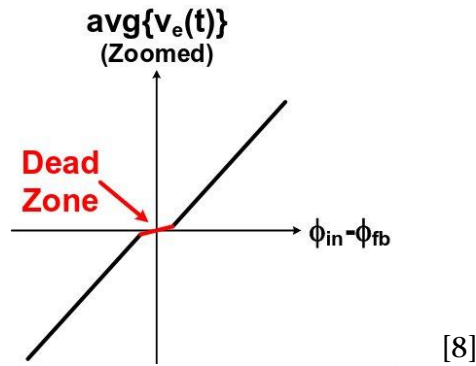


Figure 13: PFD gain with Dead Zone

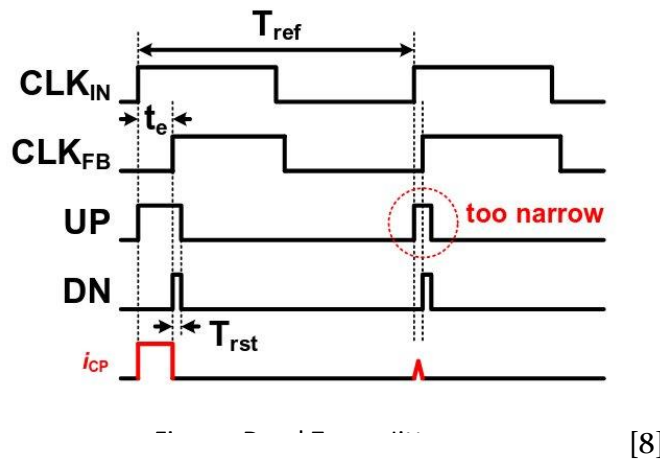
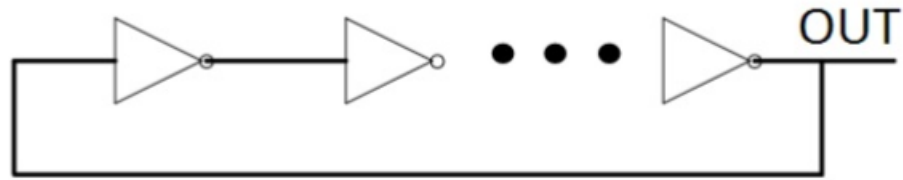


Figure 14: Dead Zones Jitter

4.3.11. PLL VCO Designs

Due to the area restriction of the die and design complexity, we have decided to implement a current starved ring oscillator. This type of VCO is very common among PLL designers because of its small area and wide frequency tuning range.

The current-starved ring oscillator functions the same way as an N stage inverter ring oscillator works. It connects a series of odd number of inverters in series, connecting the output node to the input of the first inverter. Because there are an odd number of inverters, the output and input node constantly change causing an oscillation. This oscillation is directly related to the delay of each of the inverter stages ($f = 1/2NT$) where N is the number of stages and T is the delay per inverter.

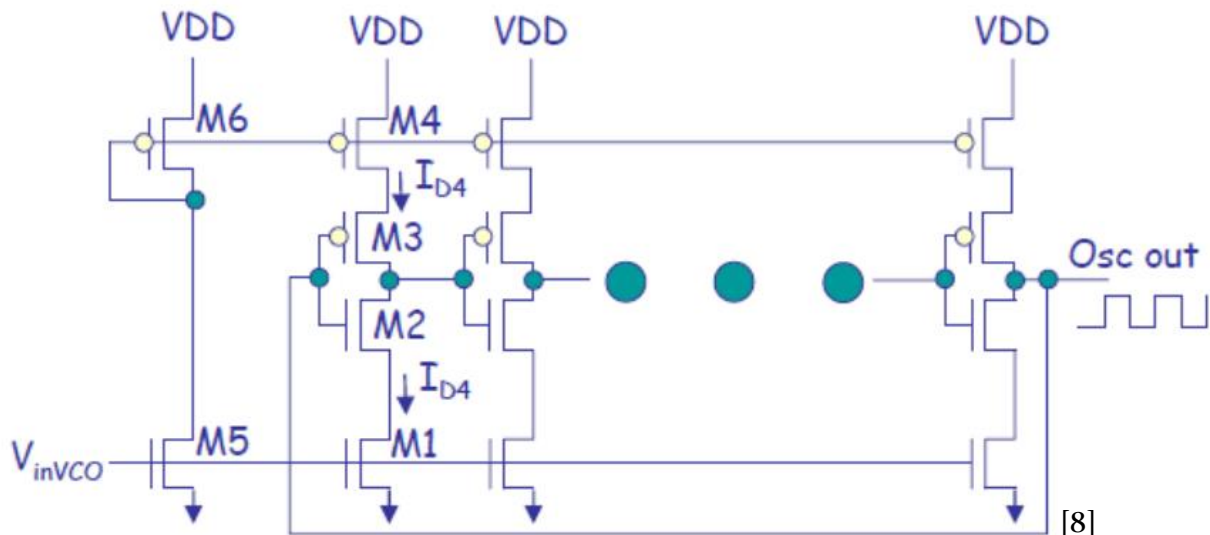


[8]

Figure 15: Ring Oscillator

The current-starved ring oscillator behaves in the same way. However, we add a current mirror, which is controlled by a control voltage to tune the latency of each stage, thus changing the output frequency. The minimum viable operating frequency occurs when the input voltage is less than the threshold voltage of M_5 , while the maximum is when it is set to V_{DD} . The center operating frequency of this circuit is equal to $I_{bias}/(N \cdot V_{DD} \cdot C_{total})$. I_{bias} is the current going through each stage, N is the number of stages, and C_{total} is the total capacitance at the output of each stage.

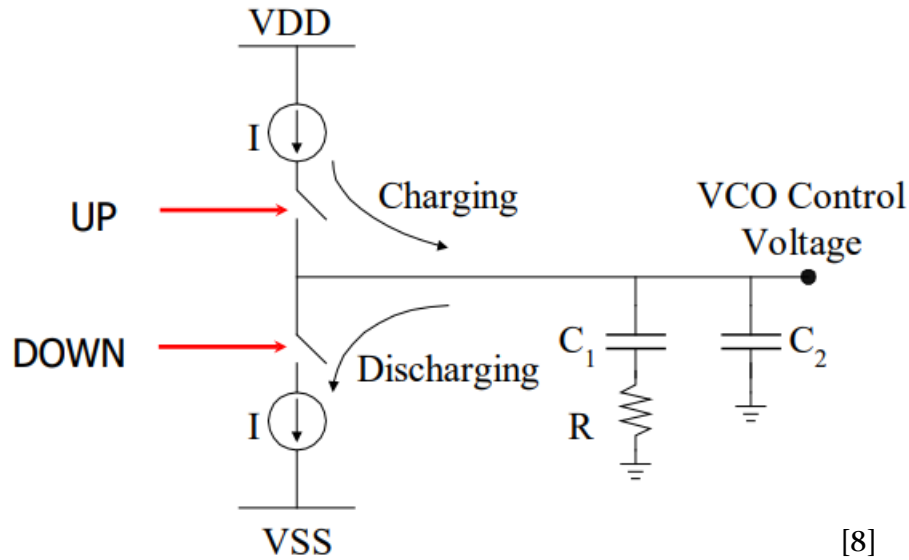
Given these equations, we plan to design a current starved ring oscillator that can operate between 1.6 to 3.2 GHz and scale it down using a basic N divider to 0.8 to 1.6 GHz. This will reduce the effective noise of the VCO. However, if this does not work in simulations properly, we will design the VCO with a 0.8-1.6GHz bandwidth. There have been designs within the online community that were able to achieve such a high frequency using the Skywater 130nm process.



[8]

Figure 16: Current-Starved Ring Oscillator

4.3.12. PLL Charge Pump Designs

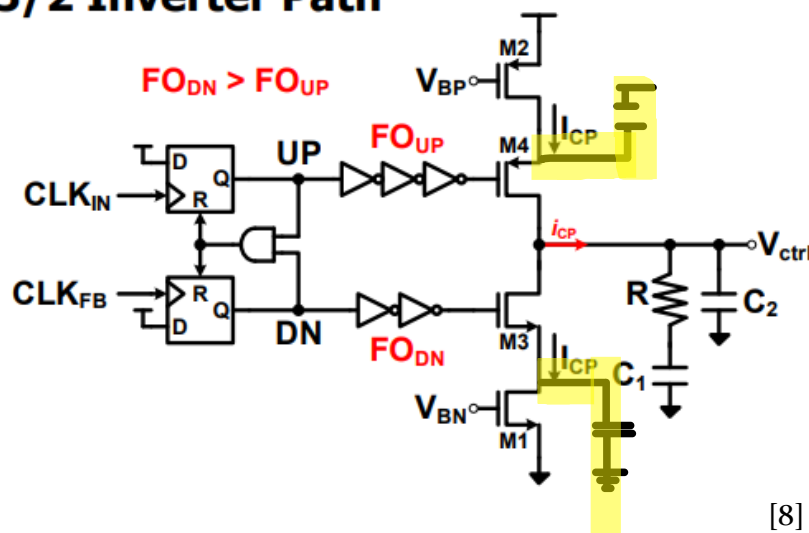


[8]

Figure 17: PLL Charge Pump & Loop Filter

The conceptional design of the charge pump component is fairly simple, yet it requires careful timing analysis. The charge pump receives the UP and DN signal from the PFD component and sources or sinks current to the loop filter, thus changing the VCO Control Voltage. Because PMOS is a better current source than NMOS, the transistor that switches the UP signal will be a PMOS transistor connected by an inverter to invert the PMOS inverter effect. However, the transistor switches the DOWN signal will be an NMOS because it is sinking current. Because the added delay in the UP stage causes a mismatch in timing, a series of buffers will be added to both UP and DOWN stages to match that delay. Both sets of inverters will be designed to have the same delay.

3/2 Inverter Path



[8]

Figure 18: Charge Pump with added buffers

One of the main issues with this design is clock sharing. When DOWN or Up signal triggers when both UP and down were initially off, a zero net current should pass through. However, the highlighted M1 Drain to source capacitors were initially at GND and/or VDD, and is now connected to a different potential (V_{ctrl}). This causes a possible voltage level disturbance at the control voltage node, causing slight fluctuations in output frequency. This results in spurs in the frequency domain.

While this solution is commonly referenced in most online references due to its simplicity, another design that would reduce these effects is a fully differential charge pump which complex circuit that dynamically adjusts the current sources

4.3.13. Decision-Making and Trade-Off

The following discusses the characteristics considered when deciding what protocol and what divider to use. Each criterion is ordered by importance with each option's relevant information listed. After the criterion, there is a discussion about our decision and how it best meets the criteria.

4.3.14. Wireless Protocols

The following characteristics were considered when determining which protocol our microcontroller would follow. Frequency was the largest factor as higher frequencies would be more difficult to implement correctly. Next security was considered, we knew we would be implementing some sort of security for our radio communication so protocols with security requirements that matched our team members' skills sets were prioritized. Finally, the modulation technique is used by the protocol. This was considered last because it is not a part of our scope but will affect teams later on.

Frequency

Zigbee: 2.4GHz worldwide, 902MHz America

BLE: 2.4GHz

Wi-Fi: 2.4GHz, 5GHz

LoRa: 169MHz, 315MHz worldwide, 915MHz America

NB-IoT: Built onto cellular networks, bands between 600MHz and 1700MHz

Security

Zigbee: Advanced Encrypted Standard (AES)

BLE: Encrypted Advertising Data (EAD)

Wi-Fi: Wired Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA)

LoRa: Advanced Encrypted Standard (AES)

NB-IoT: Inherits Security from cellular network

Modulation

Zigbee: Binary Phase Shift Keying

BLE: Gaussian frequency shift modulation

Wi-Fi: Binary Phase Shift Keying, Quadrature Phase Shift Keying, QAM

LoRa: Proprietary spread spectrum modulation

NB-IoT: Orthogonal Frequency-Division Multiplexing

We decided to use the Zigbee protocol. In terms of frequency, Zigbee comes second, allowing the microcontroller to be sub-GHz by using the American bands. For security, Zigbee uses AES, a common security protocol that our team has experience with. Notably, LoRa operates at lower frequencies and uses AES as well. We decided to use Zigbee because of its simpler and open-source modulation scheme. The goal of our project is to be open source and LoRa uses a proprietary form modulation. Lastly, Zigbee's ability to operate at 2.4GHz has the potential for future teams to modify our design into a multiprotocol radio.

4.3.15. PLL Divider Designs

The following characteristics were considered when determining which divider our PLL would use. The most important aspect of the divider was noise. Noise is the defining characteristic of a PLL so minimizing noise through the divider was a priority. Next, we looked at the magnitude of the high-speed division. The greater the divisor at high frequency, the more likely it is for the divider to fail. Lastly, we looked at the complexity of the design. A more complex design takes up more of the limited space on the die.

Noise

First Order Delta Sigma:

$20\log(10)$ less phase noise. Fractional spurring. Instantaneous frequency is not accurate.

Dual Modulus:

$20\log(10)$ more noise. No spurring. Instantaneous frequency is accurate.

High Frequency Division

First Order Delta Sigma:

Divide by 93 at high frequency

Dual Modulus:

Divide by 31 (prescaler only) at high frequency

Complexity

First Order Delta Sigma:

M/M+1 is programmable, range: 90-93. N is programmable, range: 1-10. A is programmable, range 0-9.

Dual Modulus:

M/M+1 is a fixed prescaler, 30/31. N is fixed divider, 30. A is programmable, range: 2-28.

We decided to use the First Order Delta Sigma design. While it has a higher division ratio at high frequency, it is still very low. It is a more complex design, but we decided the lower noise was worth using a more complicated design. The Fractional spurring, caused by the periodic oscillation between the two divisors, is difficult to quantify, however. Because of this, we plan to use the dual modulus design as a backup design in case the spurring results in an unusable signal.

4.4. PROPOSED DESIGN

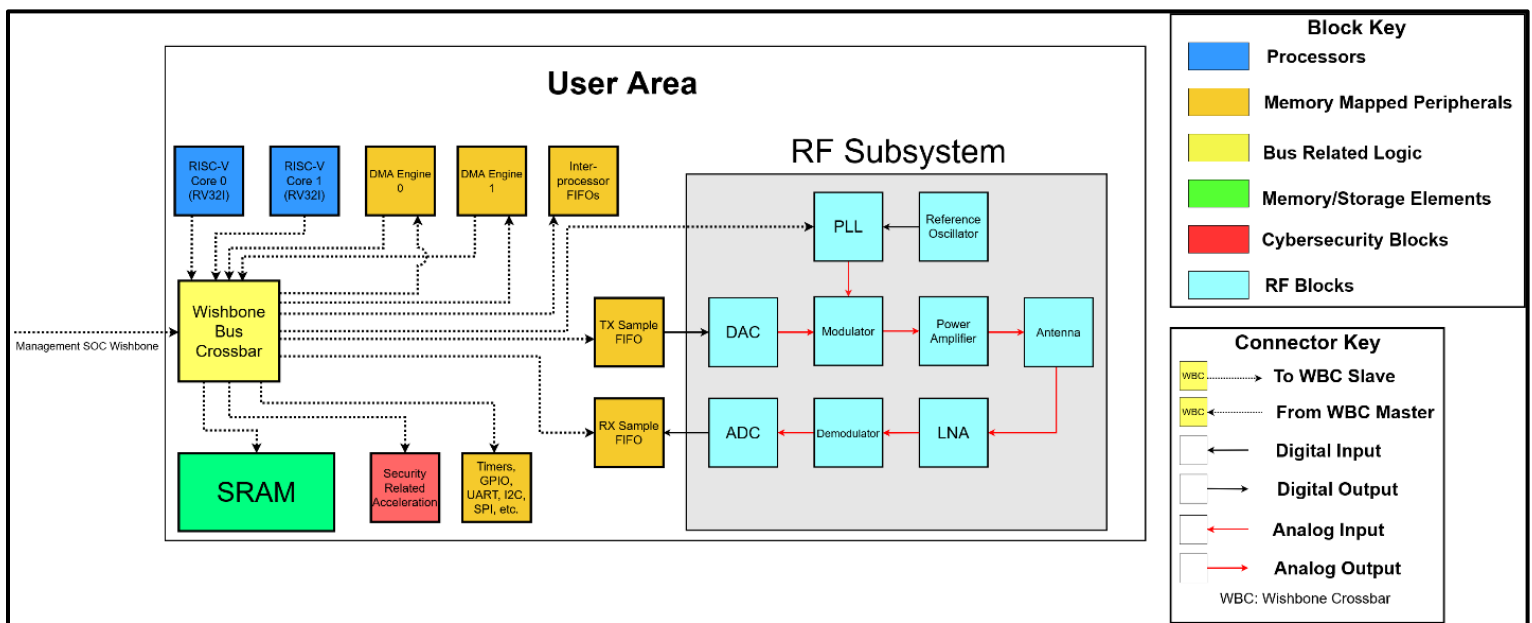


Figure 19: Full Implementation

4.4.1. Overview

At a high level, our design is composed of an analog component and a digital component. The analog component for our part of the project is a PLL capable of generating frequencies between 900 MHz and 928 MHz to transmit and receive using Zigbee. The digital component for our project consists of a processor to run user programs, peripherals to interact with other devices, and RAM to store information. These components are interconnected to form the final design.

4.4.2. Detailed Design and Visual(s)

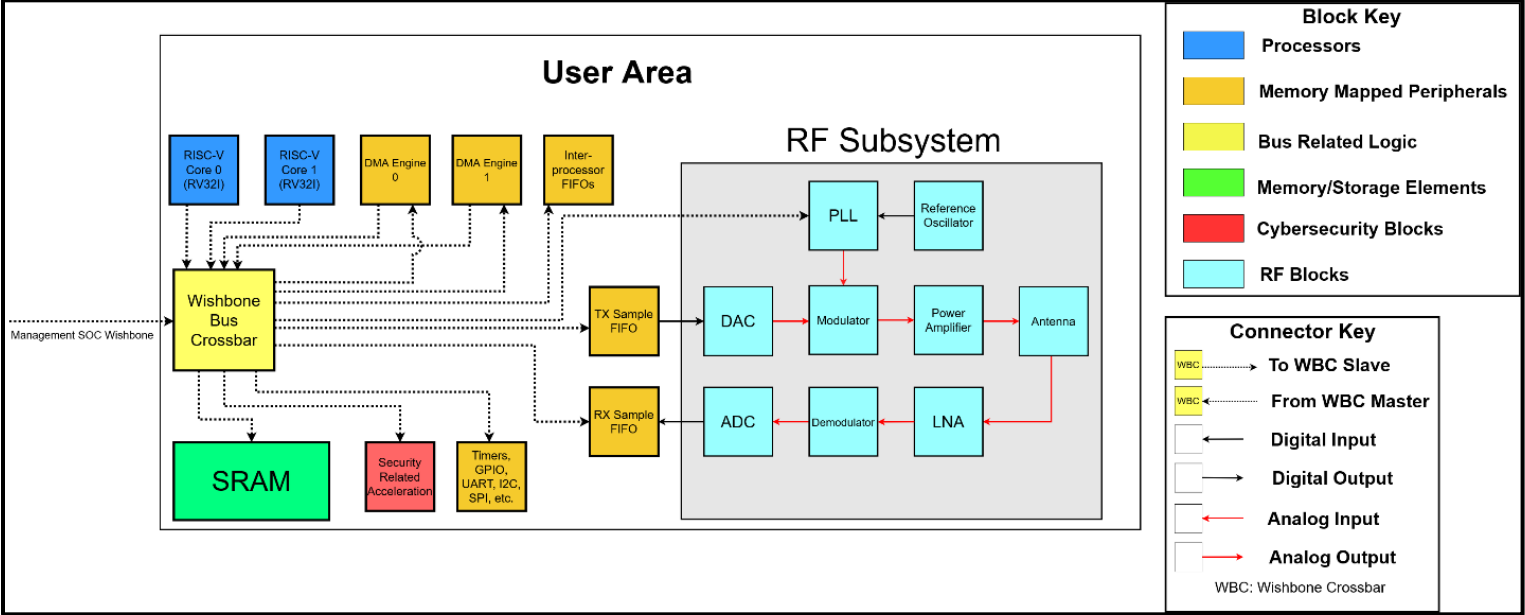


Figure 20: Full Implementation

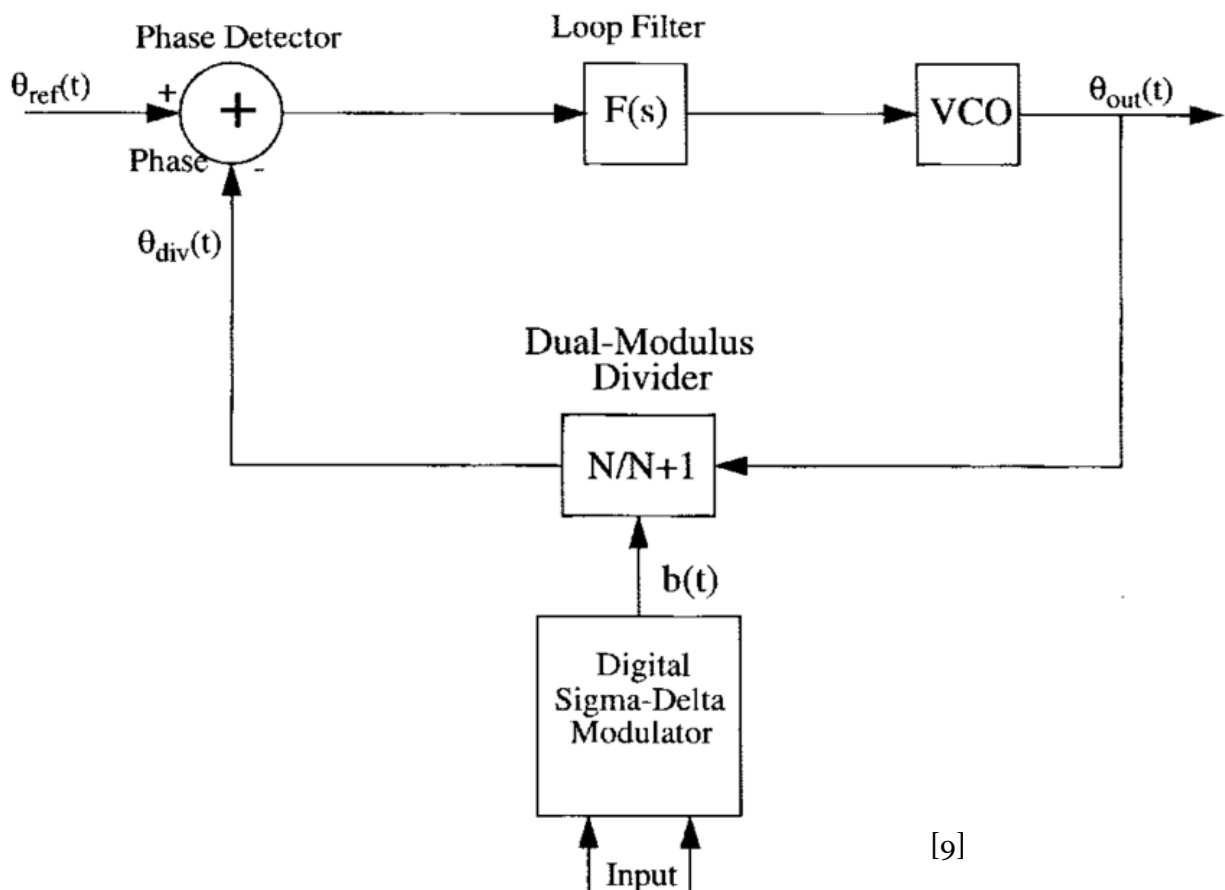
4.4.3. Analog Subsystem

The analog subsystem (RF module) contains a frequency synthesizer created using a PLL with a fixed low frequency oscillator and an N divider, a modem, power amplifiers, an antenna, and a DAC and ADC to convert between digital and analog signals. For our part of the project, we will design a frequency synthesizer to operate ten channels from 902 to 928MHz.

4.4.3.1. DAC

A digital to analog converter will be used to interface between the digital data stream and control signals to the analog components of the RF module

PLL



[9]

Figure 21: PLL Architecture

4.4.3.2. PLL: Low Frequency Oscillator

The oscillator will be attached via a breakout board and implemented as a crystal oscillator generating a frequency of a fixed, stable frequency. Based on the divider design, this will be a 10Mhz oscillator.

4.4.3.3. PLL: Phase Frequency Detector (PFD)

The charge pump is responsible of taking the signal from the PFD component and

4.4.3.4. PLL: Charge Pump and Filtering

The charge pump is responsible for taking the signal from the PFD component and incrementally charging/discharging the filtering component which will subsequently change the voltage input of the VCO thus tuning the output frequency.

4.4.3.5. PLL: Voltage Controlled Oscillator (VCO)

The VCO will generate a high frequency signal between 902 and 928MHz based on the input voltage.

4.4.3.6. PLL: Fractional N Divider

This divider will be implemented as a first order delta sigma divider. The fractional division allows for a reference frequency larger than the channel width. It will use a programmable dual modules divider, M, in the range of 90-93. Because we can use a 10Mhz reference frequency and the highest frequency channel is 928MHz, the largest division ratio is $928\text{MHz} / 10\text{MHz} = 92.8$. Two additional counters, N, in the range of 1-10, and A, in the range of 0-9, $A < N$. A and N count down simultaneously, when A = 0, the dual modules divider switches from dividing by M+1 to M for the remaining cycles N-A. This results in a fractional division of $(A(M + 1) + (N - A)M) / N$.

4.4.3.7. Modulator

The Zigbee protocol for the sub-GHz operating frequency requires a Binary Phase shift keying modulation (BPSK) which will take a sinusoidal input and change the phase by 180 degrees to represent a logical one and zero.

4.4.3.8. Power Amplifier

An amplifier circuit will be used to amplify the modulated signal to the off-chip antenna.

4.4.4. Digital Subsystem

The digital subsystem consists of two RISC-V cores, SRAM, a Wishbone crossbar, and peripherals to interface with other devices or accelerate basic functions. Due to constraints, the inter-processor FIFOs and one of the RISC-V cores will not be created during our part of the project but are included in the design since they will be part of the system in future iterations.

4.4.4.1. RISC-V Core

Several open-source options for generating a RISC-V core were investigated, since implementing this from scratch would be time consuming and be likely to result in errors that could result in non-working chips. The option that was chosen was VexRISCV, which had several configuration options for the core generation so that it can be tuned to increase performance or decrease die area use. It also natively supports the Wishbone bus, which is the interface that is already used by the Caravel harness processor. This enables easy interconnection between peripherals. The RISC-V core will be clocked using the 50 MHz clock that is also used by the management SoC. This will reduce additional clocking circuitry and make digital design easier by having all Wishbone devices on a common clock domain.

4.4.4.2. *SRAM*

SRAM is a critical part of the design, since it provides an area for data to be stored both for computation in a user program and for the data transmitted and received via Zigbee. The SRAM will be generated using OpenRAM, an open-source RAM compiler. This will enable more efficient area utilization than laying out the RAM by hand. Area utilization is a potential concern due to the 130 nm process being used, which cannot fit as much RAM as more modern processes. The minimum target is 4 KiB of SRAM for data and 4 KiB for instruction memory, but if this is not possible due to area constraints, a smaller amount may have to be used.

4.4.4.3. *Security Acceleration*

The security acceleration component will implement the AES-encryption specified by the Zigbee protocol, which will encrypt, and decrypt data transmitted and received on the microcontroller, allowing for safe and secure transmission of data. It uses 128-bit keys to encrypt/decrypt data using a symmetric block cipher.

4.4.4.4. *DMA Engine*

The DMA engine helps to reduce the amount of time the RISC-V processor spends transferring data to/from the RF subsystem to transmit/receive data. This enables more time to be spent on user program computations instead.

4.4.4.5. *Wishbone Crossbar*

The Wishbone crossbar will arbitrate access to the various peripherals that are memory-mapped over the Wishbone bus. There will be three Wishbone masters that can potentially access peripherals simultaneously. These are the management processor, the RISC-V core instantiated in the user area, and the DMA engine. The crossbar allows for much better bus throughput than using a common bus and some kind of arbitration mechanism, since all three masters can use the bus at the same time as long as they are not accessing the same resource.

4.5. FUNCTIONALITY

To use the radio MCU, users will need to connect it to power and connect any devices that they desire to be controlled by the radio MCU. Once everything is hooked up, a computer that contains the code desired to be run on the MCU and the programming software must be connected via a serial interface to the MCU. Then, the user can flash the program they wrote to the MCU, which will provide it with instructions to execute. Once the MCU is running the program, the user can then observe the signals output from the MCU, which are going to devices that the user has plugged into the various outputs of the MCU. Now that the program is running, the user can connect any wireless devices that support Zigbee to the radio MCU for wireless communication. Once the chosen devices are connected, the user will be able to observe the communication between the radio MCU and the devices by how they each respond to incoming signals. Overall, the radio MCU can be used to control and communicate with devices given that it has been powered, the devices have been connected (wired or wireless), and a program with the control code has been flashed to the MCU.

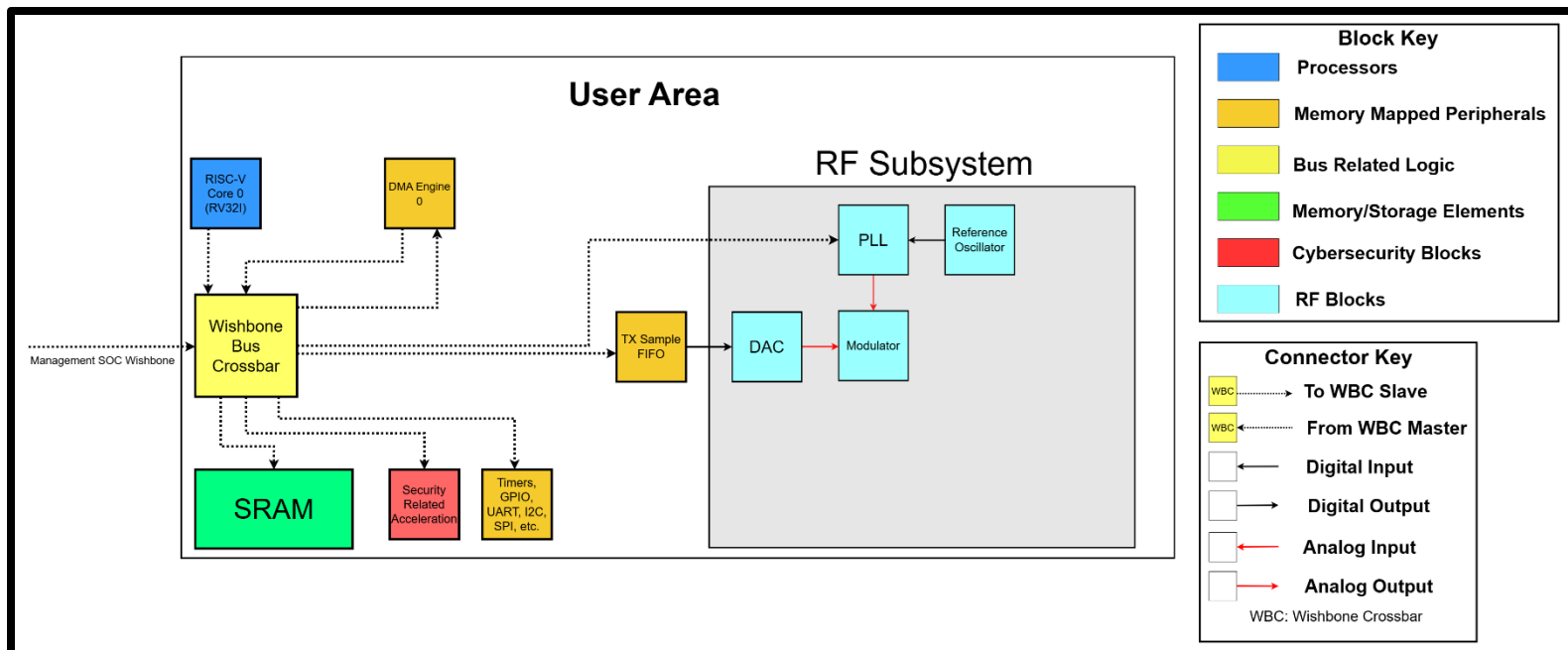


Figure 22: Minimum Design Implementation

4.5.1. Areas of Concern and Development

If we implement all we have described, our design will fully satisfy the user requirements. We would provide a platform that can be programmed by the user and control devices both wired and wireless. However, our biggest concern is going to be getting all the components implemented in our short timeframe. As seen in the “Personal Effort Requirements”, many of the tasks will take over 50 hours to complete. So realistically, we cannot finish all the tasks in two semesters, but we can finish a select few. The diagram below shows what elements of the design that constitutes our “minimum implementation”, which we believe that we can have finished by next semester.

As shown in Figure 8, our minimum implementation contains many of the digital components in the full design, but our analog section (RF Subsystem) contains much fewer components. With the digital components shown, we provide the user with a processor to execute their program, SRAM to store their program and act as memory, and peripherals that they can hook up devices to. This allows the user to control wired devices such as servos, motors, and LEDs. The RF subsystem is not complete, which means that radio communication will not be possible, which does not satisfy the RF communication user requirements. However, we still plan to implement a portion of the full RF subsystem, which then can be used by other teams who tackle this project to get a head start on finishing the RF subsystem. To address these timeline issues, we will be documenting our design and our implemented features well. Then, future senior design teams or individuals from ISU ChipForge can take our design and working components and build off them, approaching the full implementation which satisfies the user requirements.

Finally, another concern we have had while working on this project is not having all the information to create prototypes for our components. To resolve this, we have been finding online publications, talking with faculty members at ISU who have relevant experience, and attending the ChipForge weekly meetings to ask questions about the Caravel Harness and the tooling. With all these resources, we are confident that we can make substantial progress and implement our minimum design implementation.

4.6. TECHNOLOGY CONSIDERATIONS

The primary technology we are using for the design is the Caravel Harness from Efabless. This is a platform that utilizes the Skywater 130nm process and has a management system on chip (SOC) already designed for fabrication. The harness also contains a user space where digital and analog designs can be inserted and communicate with the management SOC as well as the pad frame (where the GPIO pins are). Using the Caravel Harness, our biggest limitations are the lack of documentation and the size limitations of the user area. So far, we have not found great documentation on the Caravel Harness, so all our learning has been done by looking at examples created by both Efabless and ChipForge as well as reading through the source code since the Caravel Harness is completely open source. For the size limitations, we are limited to a die area of 3mm x 3.6mm, which limits how many transistors we can fit in that space. Since using the Caravel Harness is a constraint for our design, there is not a way to remove these limitations.

4.7. DESIGN ANALYSIS

For our design implementation, we have primarily been learning how to put our own designs on the Caravel Harness. Currently, we have created a seven-segment display controller to demonstrate that we can put a design in the user space that interfaces with a real device and is then controlled from code that the management SOC's RISC-V core runs. In addition, we have started to prototype the Wishbone crossbar. We currently have a working small-scale crossbar that we will be scaling up to work with the rest of our project. Finally, we have investigated generating RISC-V cores from the open-source generator VexRISCV (which is the same generator that Efabless used to generate the RISC-V core in the management SOC). We have successfully generated a few cores from the example configurations provided by VexRISCV, but we still need to put the cores in the user area and provide instructions to them.

5. Testing

Testing is an integral part of the project, since the design cannot be altered after being submitted for fabrication. To ensure correct functionality, testing will need to be performed throughout the project and again after fabrication to make sure the implementation matches the design. Digital and analog testing will be performed using computer simulations during the implementation portion of the project. Digital testing will primarily use software during the bringup testing of the project, and analog testing will use a frequency counter and a spectrum analyzer to make sure the PLL is generating correct frequencies with acceptable noise figures.

5.1. UNIT TESTING

Unit testing will focus on individual blocks outlined in the system diagram. These tests will be relatively simple and focus on making sure the blocks function as intended.

5.1.1. Digital

Digital unit testing will be performed using the simulator provided by the Efabless tools. These tests can be automatically checked using a Verilog testbench to automate testing.

5.1.1.1. *Wishbone Crossbar*

- Verify that address mapping works correctly
- Verify that write/read data are sent to/from slaves correctly
- Verify that two masters can access different slave simultaneously
- Verify that two masters cannot access the same slave simultaneously
- Verify that crossbar functions with numerous transactions (stress test) to ensure non are skipped, returned out of order, or cause a bus to hang

5.1.1.2. *VexRISCV Core*

- Verify that a simple program can be loaded and executed correctly
- Verify that the processor generates correctly formatted Wishbone bus transactions
- Verify that interrupts function correctly and jump to intended vector

5.1.1.3. *DMA Engine*

- Verify that configuration registers can be read/written via Wishbone bus
- Verify that DMA engine generates correctly formatted Wishbone bus transactions
- Verify that DMA engine issues correct number and order of Wishbone transactions according to user register configuration
- Verify that DMA engine issues interrupts according to user register configuration
 - o On transaction completion
 - o On transaction error
- Verify that different modes of DMA transfer function as intended
 - o Buffer to buffer (memory copy)
 - o Buffer to single endpoint (for use with FIFOs)
 - o Buffer to single endpoint, transfer on interrupt (for use with UART or other peripheral that has to wait before next word can be written)

5.1.1.4. *TX Sample FIFO*

- Verify that data can be written to the FIFO via Wishbone bus
- Verify that FIFO status (full, empty, etc.) can be read through registers on the Wishbone bus
- Verify that data can be pulled out of the FIFO

5.1.1.5. *Security Acceleration*

- Verify that configuration and data registers can be read/written via Wishbone bus
- Verify that round keys are stored properly in registers
- Verify that substitute-byte transformation, shift row, and mix column operations compute the correct output
- Verify the key generated by expansion algorithm is properly copied and filled
- Verify the number of cycles encryption takes matches the expected target value

5.1.2. *Analog*

5.1.2.1. *Phase Frequency Detector (PFD)*

- Verify that it can set UP and DN signals as expected.
- Verify that the reset signal is delayed by 150 to 200 picoseconds to minimize jitter.

5.1.2.2. *Charge Pump and Filtering*

- Verify that the transistors are ON for the same delay that is designed in the PFD component. If the output fluctuates, increase the PFD Reset delay (150-200 picoseconds).
- Verify that the VCO control voltage does not change before PFD reset where both UP and DN and high causing current sourcing and sinking of the charge pump.
- Verify that the PFD-Charge Pump gain matches the post-layout simulation gain.

5.1.2.3. *Voltage Controlled Oscillator (VCO)*

- Simulate VCO and measure its gain to confirm voltage required from the PFD. Also measure maximum and minimum frequencies to ensure VCO functions in the necessary range.
- Simulate VCO and measure its phase noise. This measurement can then be used to characterize the expected phase noise for the entire system.

5.1.2.4. *Fractional N Divider*

- Simulate the divider and verify its maximum operating frequency is greater than 928MHz.
- Simulate the divider and measure its output frequency to confirm it functions correctly.
- Measure the simulated divider's spurring and determine if it is necessary to swap to the integer N design.

5.2. INTERFACE TESTING

5.2.1. Digital

The primary interface used for digital components of the design is the Wishbone interface, which is governed by an open standard from OpenCores. Since most of the individual digital components being tested have a Wishbone master or slave interface (or both), helper functions will be created to verify correct Wishbone functionality during simulation. This will be done with Verilog functions that can generate Wishbone transactions as well as check that the transactions performed are the transactions expected. Interface testing on the digital side will be primarily integrated with the unit testing, since the interface is integrated with the units.

5.2.2. Analog

The PLL will use the pad frame for main connection to the inputs and outputs of the PLL as well as two connections for testing its functionality and measuring its characteristics. The testing and the PLL divider controls will be fed from the designed RISC-V processor. Except for the divider and testing controls, the PLL is mostly isolated from the rest of the digital and security components. Therefore, after ensuring the functionality of the digital component, the main debugging interface will be through the pad connections.

5.3. INTEGRATION TESTING

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

5.4. SYSTEM TESTING

5.4.1. Digital

In addition to the unit tests to make sure each individual block functions as intended; additional simulations will be run on the entire digital system. These will be somewhat limited in scope, since simulating large programs would take large amounts of time which would slow down the development process.

- Verify that simple test programs can be run correctly
 - o These are programs that access several peripherals (DMA, FIFO, SRAM, etc.)
- Verify that system does not freeze or enter undesired states during program operation
- Verify that system comes out of reset in the correct manner

5.4.2. Analog

The system testing plan of the PLL mostly consists of the closed loop simulations of the system, ensuring optimal working conditions.

- Measure its output frequency to verify that it outputs the correct frequencies.
- Measure the simulated PLL's phase noise to ensure a useable signal.
- Measure the simulated PLL's lock time.
- Measure the simulated PLL's settling time and confirm that it conforms to the Zigbee standard. The [ATSAMR30M18A](#), a commercial Sub-GHz ZigBee compatible RF microcontroller, has a 170us settle time. We will design the PLL to have the same settle time.

5.5. REGRESSION TESTING

5.5.1. Digital

Regression testing will be performed by automating the unit and system tests and integrating them with a CI/CD flow on the Git repository. This will result in changes being automatically checked to make sure that no previous working functionality was broken by new code. Code will not be allowed to merge back into the main branch of the repository without first passing the CI/CD checks.

5.5.2. Analog

Due to the closed loop nature of the PLL, any change to the subcomponents will ultimately affect the overall functionality and characteristics. This is why the analog regression testing will consist of rigorous simulation testing of the closed loop system whenever a characteristic of a subcomponent changes. This will ensure we spot any design mistakes throughout the system integration and testing process before fabrication.

5.6. ACCEPTANCE TESTING

After the chips have been fabricated, the tests outline below in the bringup test plan will be performed either by our client, ChipForge, or future senior design teams, since manufacturing will

not be completed by the end of the spring semester. These tests ensure correct functionality of the various components and the system as a whole.

The following tests are intended to be performed in order since certain tests may assume that previously tested modules are working as intended. The one module assumed to function correctly from the beginning is the Caravel Harness management SoC. This is a reasonable assumption because the management SoC is silicon proven and is unchanged from previous revisions.

5.6.1. Digital

5.6.1.1. *Wishbone Crossbar pt. 1*

- Verify that management SoC can access dummy registers via Wishbone Crossbar
 - o Will have some safe data, dummy register to read/write, build information
- Verify that management SoC can access SRAM via Wishbone Crossbar
 - o Read/write some known sample data

5.6.1.2. *VexRISCV Core*

- Verify that a simple program can be loaded and executed via management SoC
 - o Program should not rely on peripherals, as they will be untested
 - o Hook lower bits of processor data bus to the logic analyzer to validate that processor is functioning
- Verify that core comes out of reset

5.6.1.3. *Security Acceleration*

- Verify that configuration and data registers are accessible via VexRISCV core Wishbone bus
- Verify that acceleration functions work correctly
 - o Perform some known computations on accelerator and confirm that results match expected

5.6.1.4. *DMA Engine*

- Verify that configuration registers are accessible via VexRISCV core Wishbone bus

5.6.1.5. *DMA Engine/Wishbone Crossbar pt. 2*

- Verify that VexRISCV core and DMA engine can both issue Wishbone transactions
 - o If accessing different slaves, both should run in parallel
 - o If accessing same slaves, one should block
 - o Use simple test programs to move some data around between SRAM and Security Acceleration blocks
- Verify that interrupts work correctly from DMA engine
- Verify that different DMA transfer modes work as intended

5.6.1.6. *System Testing*

- Verify that simple test programs can be run correctly
 - o These are programs that access several peripherals (DMA, FIFO, SRAM, etc.)
- Verify that system does not freeze or enter undesired states during program operation
- Verify that system comes out of reset in the correct manner
 - o This will already be partly tested by previous steps, but would be helpful to verify just in case

5.6.2. Analog

5.6.2.1. PLL Testing Architecture

The Fref input signal will be connected to an off-chip crystal oscillator through the pad frame. We will add a transmission gate after the charge pump and between the filter pad and the VCO pad. The loop filter will be connected off-chip which will save area for other necessary components. It can also slightly change the loop dynamics if the values are properly re-configured. When the transmission gates are off, the input of the VCO will only be connected to the frame pad which will allow open loop testing for the VCO.

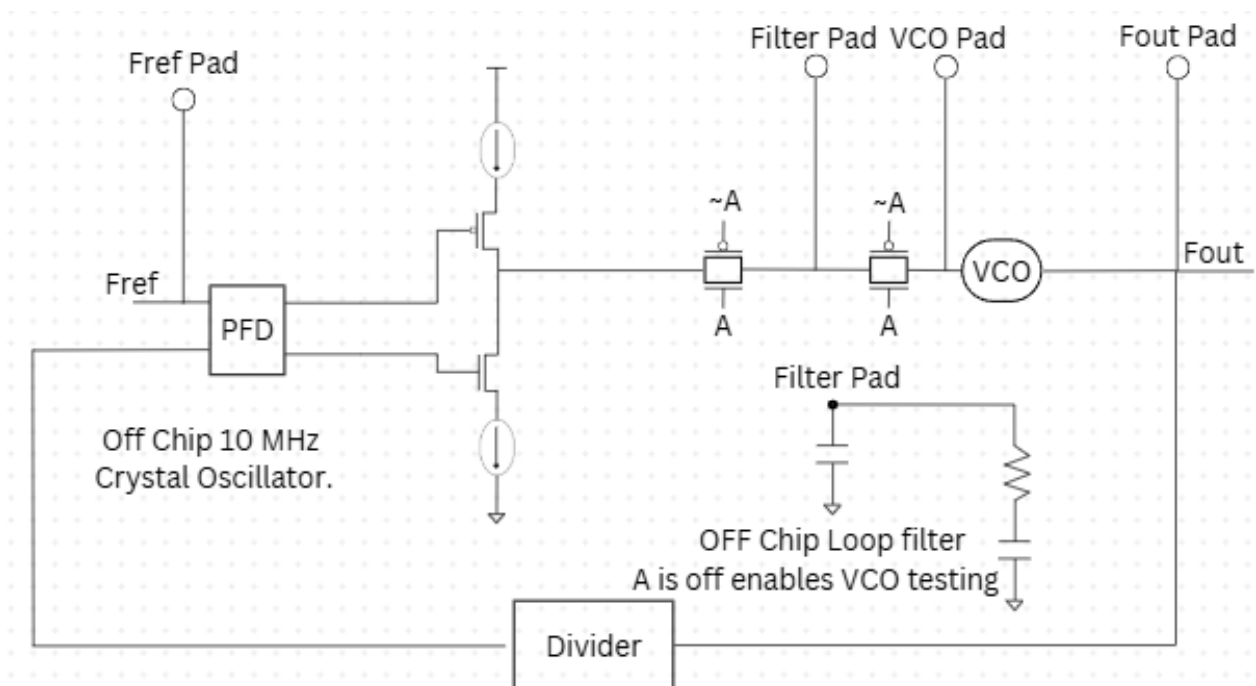


Figure 23: PLL Test Diagram

5.6.2.2. PLL Measurements and Characterizations

Low Frequency Oscillator

- Measure frequency output
- Should be a very stable 10MHz

VCO Open Loop Configuration

- Turn off transmission gates
- DC Sweep the input of the VCO
- Connect the output to a spectrum analyzer
- Verify the VCO gain
- Verify the VCO bandwidth of 0.8GHz -1.6GHz
- Measure the noise of the VCO and compare with Pre-fabrication

PLL Closed Loop System

- Verify output frequency matches each channel in the 915MHz Zigbee range
- The PLL can lock to each 1MHz channel from 902 to 928MHz
- Measure phase noise
- The phase noise of the PLL must be in the acceptable range for a standard Zigbee receiver.
- Verify lock time
- Confirm the lock time closely matches the simulated result.
- Verify settling time
- Confirm settling time closely matches simulated result.

5.7. SECURITY TESTING

Verify that the output of the plaintext is properly encrypted.

Ensure each round function and round key is properly stored in each register.

Each substitute-byte transformation, shift row, and mix column operation should be consistent in its output

Key generated by the expansion algorithm is properly copied and filled.

Verify the number of cycles each encryption takes matches the expected target.

5.8. RESULTS

Minimal testing has been performed so far, since we are in the early phases of prototyping the design. Work will begin on testing starting in the spring semester as the project transitions to implementation.

6. Implementation

6.1. WISHBONE CROSSBAR

The Wishbone crossbar is an important component of the design and presented concerns about area utilization. To help address these concerns, a simple crossbar was prototyped. The prototype crossbar is only 2x2, meaning that it has 2 Wishbone slave interfaces that connect to Wishbone masters, and 2 Wishbone master interfaces that connect to Wishbone slaves. Each external master can communicate with either slave, provided the other master is not accessing the slave at the same time. Master 0 will win arbitration over master 1 if both attempt to access at the same time. This crossbar has several advantages over trying to use a common bus, since as long as masters don't attempt to access the same slave, they can work in parallel, which provides extra throughput. In the full system design, there could be as many as 5 masters (2 RISC-V cores, Management SoC, and 2 DMA engines), so the crossbar would allow for much better system performance overall. The 2x2 seems to function correctly after some very minimal testbench simulations.

In the future, the Wishbone crossbar will be modified to be generic, so that it can take a configurable number of master and slave interfaces. This will require only minor modifications to

make some constants generic. Additional tests will need to be created since the crossbar must be as reliable as possible, since virtually all other digital components will rely on it to function correctly.

6.2. SEVEN-SEGMENT CONTROLLER

To make sure that we understand how to go from a concept to a design that can be fabricated, we implemented a simple design that controls two seven-segment displays. The FPGA testbench provided by the ChipForge co-curricular gives us the ability to emulate the entire Caravel harness and verify that the digital design is functional. In addition, a physical dual seven-segment display module is attached to the FPGA over a PMOD interface. The existence of this seven-segment module enables us to visually verify that the design is functional and spawned the idea for this simple project in the first place.

After much experimentation, we got a design to work on the FPGA that enabled us to write a single number to one of the seven-segment displays. The design was provided operational codes and the display information via the wishbone bus. A program written for the existing processor in the Caravel harness commands the processor to write to specific memory addresses, which are memory mapped to our design over the wishbone bus. This enables a user to create a C program that is ran by the existing processor on the Caravel harness that can control the seven-segment display. Once we had this working, we then modified our design to support writing to both the seven-segment displays in the dual seven-segment display module. After the design was created and verified visually on the FPGA, we ran the design through OpenLane to harden it. This produced all the artifacts required for fabrication. These artifacts were then taken and put into a fabrication tapeout, so next year (2025), we will get to see this project on a physical Caravel harness. Our team found this project vital in understanding how to take a concept to fabrication using the Caravel platform. We will be greatly relying on what was learned during this process when we go to implement our design in CPR E 4920.

7. Ethics and Professional Responsibility

7.1. AREAS OF PROFESSIONAL RESPONSIBILITY/CODES OF ETHICS

Table 4: Areas of Professional Responsibility and Codes of Ethics

Area of Responsibility	Definition	Relevant Item from Code of Ethics	Team Interaction
Work Competence	Completing assigned tasks and on time.	IEEE Ethics 5, 6	Team members have put in effort to learn about new tools and processes to ensure that they are competent during design and implementation.
Financial Responsibility	Creating a product that meets	IEEE Ethics 3	Designed a product that considered the

	expectations for its price.		financial amount provided, as well as made it accessible to the public through an open-sourced implementation.
Communication Honesty	Being truthful about work that has been completed.	IEEE Ethics 5	Team has done a good job with timely communication via Discord and reporting hours.
Health, Safety, Well-Being	Providing products and services that consider consumer safety, health, and well-being	IEEE Ethics 1	Team is using industry standards and best practices to make sure our final design is safe and works correctly to provide a positive experience to end users.
Property Ownership	Respecting the ideas and products of others.	IEEE Ethics 5	Team has checked all code used in the project to make sure it is open-source and we can use it without intellectual property infringement.
Sustainability	Considering impact on environment and natural resources	IEEE Ethics 1	Our team is using an older fabrication process that already exists, so fabricating our design does not require many additional resources that could hurt the environment.
Social Responsibility	The product or service is beneficial to its users and others.	IEEE Ethics 2, 10	Our team has designed product that enables users to learn about all the inter-workings of it. This product can be used in many educational contexts and promotes critical thinking and thinking like an engineer.

Our team is performing well in work competence, as each member has contributed significantly to the research and development of our project. Each member has been open-minded in creating a

positive environment where each member can freely express their creative ideas and obtain feedback from one another. This signifies strong performance because each member actively contributes to design decisions, research, and testing, and has completed their tasks on time. One area that our team needs to improve on is sustainability, as while our project does use existing fabrication processes that wouldn't require many resources, our design does not aim to benefit the sustainability of the environment in many ways future to improve, there could be more considerations on power consumption for environmental concerns.

7.2. FOUR PRINCIPLES

Table 5: Four Ethical Principles

	Beneficence	Nonmaleficence	Respect for Autonomy	Justice
Public Health, Safety, and Welfare	Our design will implement encryption to protect user data	Our design is designed with best practices to avoid potential safety concerns	Design allows complete user control	Our design is tailored towards students and general education about design process.
Global, Cultural, and Social	Allows people without many resources to learn about RF MCUs	Our design will be accessible to people with different experiences and backgrounds in engineering	The open nature of the design allows different users to use it for their unique purposes	The design's documentation and openness combine to allow users from marginalized communities to use it
Environmental	Efabless is utilizing previous processes to reduce waste	Our design is using best practices to reduce excess energy	Users will have some control over power consumption	The Efabless organization and fabrication facilities already exist, so the fairness of environmental impact is determined by Efabless and not us.
Economic	Allows for students and educational institutions to fabricate at a much lower cost than alternatives	The Efabless organization and fabrication facilities already exist, so implementing this project does not have a large economic impact.	Users can decide to fabricate this design if they want to or just view the artifacts associated with the design for free.	This project is entirely open-source and any existing artifacts we pull in are open-source. This means that no one has to pay to see our artifacts.

One point that is important to the project is the intersection of Global, Cultural, and Social and Respect for Autonomy. The open-source nature of our design will allow for users to use the design as they please, without the limitations of licensing that would come with closed-source designs. One of the largest constraints on the project is that it be open-source, so this will be implemented by default, but we will also help support this with good documentation, which makes it easier for users to do whatever it is they want with the product.

One point that we have not emphasized on the whole is environmental impact in any category. However, our design is functioning as a proof of concept and a prototype, and thus will not have a significant environmental impact, since only a small number will be produced. Additionally, we have limited ability to change the environmental impact, especially around the manufacturing process, since this is a constraint from our client. That being said, the process is based on older technology, and as such is reducing waste by reusing said technology. This results in a lesser environmental impact than trying to keep up with the most modern processes.

7.3. VIRTUES

7.3.1. Responsibility

The team defines responsibility as fulfilling the tasks assigned and expected of each other, in a timely manner. Being responsible means also being held accountable of what you oversee and making ethical choices. The team has met and will continue to meet the virtue of responsibility by holding each other accountable for the tasks that are assigned and provide updates on the progress of tasks. Team members will notify the team of any pending issues that may arise.

7.3.2. Respect

The team defines respect as treating each other equally and showing regard for each other's abilities and contributions to the team. Being respectful to one another also entails listening to each other's ideas and valuing each other's views. The team has and will continue to show respect to each other by listening to each other's ideas and showing consideration for one another even in disagreements.

7.3.3. Flexibility

The team defines flexibility as a willingness to compromise with one another and embracing new challenges and ideas. The team has already shown flexibility in establishing meeting times outside of class and allowed each other flexibility in work during high stress environments. The team will continue to show flexibility by being considerate of each other's time and commitments and be open to a workflow that best fits the schedules of each other.

7.3.4. Individual Virtue Assessment

7.3.4.1. *Ibram Shenouda*

The virtue that I demonstrated throughout the semester is responsibility. As an electrical engineer who is pursuing a career in VLSI design, this project hugely contributes to my overall goal of becoming a design engineer. I believe I was responsible in learning new engineering concepts about

ASIC design and Phase Locked Loops. Because no undergraduate class covers PLL concepts and design implementation, I took on the responsibility of educating myself these concepts to ensure proper working conditions of the PLL.

The virtue that I believe that I have not demonstrated so far is flexibility. Due to my tedious schedule and our large group contributors, it was time consuming to pick meeting times in the beginning of the semester. After a few weeks, we decided to meet on Saturdays which mitigated most of our time conflicts.

7.3.4.2. *Noah Thompson*

The virtue that I demonstrated the most throughout the first semester of this project is flexibility. Our project requires a significant amount of analog design and our team only has one electrical engineering student, Ibram. To implement our desired analog designs, I volunteered to learn analog and mixed signal design to assist in the development of the analog components. I recognized that, while it is not my expertise, we needed a larger analog team to implement the PLL. My goal is to contribute as best I can, where I can to ensure we have a successful project.

The virtue I have not demonstrated as well is responsibility. Due to my lack of knowledge in the mixed signal domain. I spent the start of the semester researching and learning to be able to assist to the analog team. I did contribute to team assignments as much as I should have initially. This improved as I became more comfortable with analog design and felt I could dedicate more time to team assignments.

7.3.4.3. *Nathan Stark*

The virtue I have demonstrated most throughout this semester has been responsibility. I have communicated clearly what my status on issues is, attended meetings to the best my schedule allows, and taken on extra tasks to make sure we got all assignments completed on time. Responsibility is important to me because I know how frustrating it can be to have to work with people who don't take responsibility, and I make it my goal to make sure that I am not contributing to a negative experience for my teammates.

The virtue I have not demonstrated as well is flexibility. I had many other commitments this semester and while I allocated a sufficient number of hours to the project, sometimes they didn't line up well with other team members. This sometimes resulted in miscommunications. In the next semester I will try and schedule team meetings ahead of time before my other commitments to help allocate more time to the project.

7.3.4.4. *Nolan Eastburn*

So far, I have demonstrated the virtue of responsibility the best in this first semester of senior design. I have clearly communicated what I will worked on and have kept to my commitments. At times, I have pushed some commitments a few days ahead, but I have clearly communicated this to make sure my teammates were okay with it. In addition, I committed to getting the seven-segment project into the fabrication tapeout, which took a lot of time out of my week. I made sure that the seven-segment project was implemented by the fabrication tapeout deadline to show that our team knew how to take a concept and fabricate it.

I have identified that one of my weaker virtues is flexibility. I enjoy getting immersed into things I enjoy such as embedded programming and digital design. Because of this, I at times did not communicate with the analog team to understand their design. I was so sucked into my own work that I did not take time to learn about what others were doing, which hurt my overall understanding of the project. In the future, I plan to be more flexible by taking time to learn about what my other team members are doing instead of getting sucked into my own work.

7.3.4.5. *Ethan Kono*

A virtue I have demonstrated throughout this project is respect for the rest of my team members by being an active listener and open to new ideas. Respect is important to me because it enables a positive environment where everyone feels they can contribute ideas freely and equally and provides everyone with the opportunity to express their thoughts in a constructive manner.

A virtue I have not demonstrated throughout this project as well has been flexibility. It is important to me because the team has been great at being flexible with meeting times and actively contributing even when they are busy, but I have not contributed nearly as much when I have been busy with other commitments. Going forward, I can be more flexible with my time by making this project more of a priority and commit more of my time towards the project.

7.3.4.6. *William Custis*

The virtue I have demonstrated during this project is respect for my team members by trusting in the quality of their work, especially when they were working on components that were out of my area of expertise. Respect is an important virtue to me as without it a team will fall apart. A lack of respect tears down trust and teamwork. By respecting my teammates, we are able to work together better and be more efficient, which has been critical to our project.

A virtue that I have been lacking in during the course of this project has been flexibility. While I came to as many of our out of class meetings as I could, I missed a few due to having conflicts at that time. Flexibility is an important virtue in a group project, especially a group project of this size. It will always be difficult to find a time for all six of us to meet as seniors in engineering. I could demonstrate this virtue better by trying to make more time to attend our meetings or offer more times that would work better for me.

8. Closing Material

8.1. CONCLUSION

The design work done so far was primarily focused on shaping overall system requirements and finding a subset of these requirements that could feasibly be implemented by our team in the course of the spring semester. Most of these requirements focused on digital components that are necessary for the system to function and the PLL for the analog portion. Moving into the second semester, our primary goals are to implement and simulate the digital and analog portions of the design according to the schedule outline in our project schedule, and submit the design for fabrication by April 11th. From our research and analysis, we believe that this is a realistic timeline.

Our work has also included small prototypes, including a basic digital component that was put in the tapeout of a project completed in the fall semester. These have helped to familiarize us with the tools, making it more likely that we can complete our goals.

8.2. REFERENCES

- [1] "CC1352P," *CC1352P data sheet, product information and support* | TI.com. [Online]. Available: <https://www.ti.com/product/CC1352P>. [Accessed: 07-Dec-2024]
- [2] *ESP32*. [Online]. Available: https://mm.digikey.com/Volumeo/opasdata/d220001/medias/images/425/MFG_ESP32-DEVKITC-VE.jpg
- [3] "At just \$6, raspberry pi pico W brings Wi-Fi to IOT designs - news," *All About Circuits*. [Online]. Available: <https://www.allaboutcircuits.com/news/at-just-six-dollars-raspberry-pi-pico-w-brings-wi-fi-to-iot-designs/>. [Accessed: 07-Dec-2024]
- [4] *STM32*. ST Microelectronics [Online]. Available: <https://estore.st.com/media/catalog/product/s/t/stm32wb09kev6tr.jpg?quality=80&bg-color=255,255,255&fit=bounds&height=&width=>. [Accessed: 2024]
- [5] "TI CC2540F256RHAR," *CC2540* | Buy TI Parts | TI.com, https://www.ti.com/product/CC2540/part-details/CC2540F256RHAR?utm_source=google&utm_medium=cpc&utm_campaign=ocbtistore-promo-epd_opn_en-cpc-storeic-google-ww&utm_content=Device&ds_k=CC2540F256RHAR&DCM=yes&gad_source=1&gclid=CjoKCQiAgdC6BhCgARIsAPWNWH04CRbtU3ZF1Rh5Fdk2EehsXaQrK8UR7MPKl9aB3ZX4Ntm79qtXZgoaAj5YEALw_wcB&gclsrc=aw.ds (accessed Dec. 7, 2024).
- [6] "Dual-modulus prescaler," *Wikipedia*, 03-Dec-2024. [Online]. Available: https://en.wikipedia.org/wiki/Dual-modulus_prescaler. [Accessed: 07-Dec-2024]
- [7] "Digital PLL, All digital PLL, Analog PLL," *Movellus*, 02-Apr-2023. [Online]. Available: <https://www.movellus.com/all-digital-pll-phase-locked-loop/>. [Accessed: 07-Dec-2024]
- [8] S. Palermo, 2024 [Online]. Available: <https://people.engr.tamu.edu/spalermo/ecen620.html>. [Accessed: 07-Dec-2024]
- [9] Frequency spectrum of sigma-delta modulator output bits. | download scientific diagram, https://www.researchgate.net/figure/Frequency-spectrum-of-sigma-delta-modulator-output-bits_fig10_2977787 (accessed Dec. 8, 2024).
- [10] "A study on hardware attacks against microcontrollers," Federal Office for Information Security, <https://www.bsi.bund.de/EN/Service-Navi/Publikationen/Studien/Hardware-Angriffe/Hardware-Angriffe.html> (accessed Dec. 7, 2024).
- [11] "Attacks against industrial machines via Vulnerable Radio Remote Controllers: Security Analysis and Recommendations," Trend Micro (US),

<https://www.trendmicro.com/vinfo/us/security/news/vulnerabilities-and-exploits/attacks-against-industrial-machines-via-vulnerable-radio-remote-controllers-security-analysis-and-recommendations> (accessed Dec. 7, 2024).

8.3. APPENDICES

8.3.1. Security Analysis

The security analysis appendix is an overview of the potential attack vectors and vulnerabilities on microcontrollers and radio frequency modules. This appendix splits the analysis into two major sections, 8.3.1.1 which covers the vulnerabilities in microcontrollers, and section 8.3.1.3, which covers the vulnerabilities in radio frequency modules. Each section covers how the different attack vectors work in concept, how they might affect the system, and proper countermeasures and solutions for each attack vector or vulnerability. The conclusion is meant to serve as a way to analyze which attack vectors are of the highest priority and concern within the context of this project.

8.3.1.1. Attacks & Vulnerabilities in Microcontrollers

Control Flow Manipulation Attacks:

Control flow determines the order in which instructions are executed. Attacks on control flow aim to modify the flow of execution on a program by executing malicious code or prevent code execution to bypass password checks or encryption. Some of the most common types of physical attacks on embedded devices and microcontrollers according to the Federal Office for Information Security [10] are on voltage and clock glitching, electromagnetic fault injection (EMFI) and laser fault injection (LFI).

Fault Attacks:

Fault attacks are a subunit of control flow manipulation attacks that specifically used for causing “erroneous behavior” such as bypassing execution of specific instruction(s). Corrupting data on memory, corrupting data during a bus transfer, modifying the instructions in the program, or changing the program counter to execute instructions in different orders [10].

Side-Channel Attacks:

- Physical properties of the microcontroller can be observed to obtain a cryptographic secret.
- Power consumption and electromagnetic emissions are the most exploitable in microcontrollers

Encryption & Key Distribution:

Encryption is the bare minimum of security in wireless systems. When communicating wirelessly with radio frequencies it is very easy for messages to be intercepted. While most information being transmitted using this microcontroller will not be of critical importance, having the choice to encrypt a message or not could provide a good learning opportunity for students or club members to learn about the importance of encryption.

AES is the encryption recommended by ZigBee documentation. Additionally, it is one of the most popular encryption methods for wireless communications and is incredibly secure.

AES is a symmetrical cipher so the receiving device will need to have the key used to encrypt the message to be able to decrypt it. Data is encrypted using the network key on a ZigBee network and every device on the network has the same password key. There are a couple options the ZigBee protocol supports for distributing the network key to new devices, but we will focus trust center networks for this project as we will always have our one device, we trust being our own microcontroller. When a new device wants to join the network, it will contact the trust center which can then send a copy of the network key to the new device, directly disallowing it from joining, or just ignore it preventing it from joining the network. For the purposes of our project, we will use a simplified version of this where we either allow or disallow the device from joining as we will likely only be ever communicating between our device and one other.

8.3.1.2. Countermeasures and Solutions for Vulnerabilities in Microcontrollers

Countermeasures to control flow manipulation attacks can be implemented in both hardware and software, and which one is implemented depends on the scenario. Listed below are the pros and cons of both hardware and software implementations.

- Hardware Pros:
 - o Less impact on system performance as it doesn't use execution time from the CPU
- Hardware Cons:
 - o Can be expensive to implement
 - o Cannot be retrofitted after development
 - o Require modification of the system hardware and potential additional peripherals.
- Software Pros:
 - o No special hardware is required
 - o Retrofitting of additional security measures can be done via software updates
- Software Cons:
 - o Compiler optimizations may have to be disabled
 - o Increases code base and complexity for developers

The actual solutions for control flow attacks include the following list below. Each of these potential solutions can be implemented in hardware or software, depending on where or how it is implemented, but are all potential options to consider when mitigating control flow manipulation attacks.

Classic Loop Hardening:

- Involves duplicating loop counters & exit conditions so that each condition is checked twice before exiting the loop.
- Mainly utilized to detect fault injection attempts.

Instruction Redundancy:

- Execute critical instructions at least two times, to ensure the same result is returned, meaning even if instructions are skipped, the critical instruction will still execute, even if it is skipped.

Function Duplication:

- Involves duplicating a function, and having both functions take the same input, but store the results in different variables and compare both after execution to detect fault attempts.

Countermeasures to side-channel attacks:

Masking:

- Involves applying random masks to secret values in order to create secret shares that are used for cryptographic computations [11].

Blinding:

- Like masking but utilizes an algorithm to mask sensitive values [11].

Shuffling:

- Hides the order that values are processed, and schedules operations at random, which thereby hiding the side-channel measurements with secret information [11].

8.3.1.3. Attacks & Vulnerabilities in Radio Frequency Modules

Replay Attacks:

“Replay attacks record the RF packets and replay them to obtain basic control of the machine” [11]. This involves the attacker intercepting a valid message and then either delays it or resends it or misdirects it. In extreme cases where sensitive information such as passwords are leaked, an attacker may retransmit the sensitive information after a delay, leaving little indication that the data was even stolen or compromised.

Command Injection:

Command injection attacks in a radio frequency context often involve the attacker recording commands via a data transmission of radio frequency module, capturing data and utilizing reverse engineering to derive other commands and then retransmitting known commands to attack a system. “Knowing the RF protocol, the attacker can arbitrarily and selectively modify RF packets to completely control the machine” [11].

E-Stop Abuse:

“Attack can replay e-stop (emergency stop) commands indefinitely to engage a persistent denial-of-service (DoS) condition” [11]. This type of attack looks to take advantage of specific instructions sets or features on radio frequency modules, ones that specifically send signals to the receiver unit in an emergency that cuts power off. Attackers who intercept these commands can create denial of service situations by preventing the radio frequency module from transmitting any data by consistently sending an e-stop command to prevent it from running.

Malicious Repairing:

“The attacker can clone a remote controller or its functionality to hijack a legitimate one” [11]. This attack mostly applies to radio frequency modules that allow for a “cloning” feature that allows creation of copies of transmitting units. A clear example would be having multiple transmitting units but only one receiver, which would allow multiple operators to control a single receiver.

Malicious Reprogramming and Remote Attack Vectors:

“The attacker ‘trojanizes’ the firmware running on a remote controllers to obtain persistent, full remote control” [11]. This attack utilizes scenarios where IT endpoints are not secured, allowing any firmware to be flashed to the microcontroller. If the microcontrollers are not code-protected and allow for flashing of the microcontroller’s memory to reassign and reconfigure a device before it is installed or sold, allowing for potential backdoors or harmful code to be installed on the device. In cases like this project that are open-sourced, it is important to consider.

8.3.1.4. Countermeasures and Solutions for Vulnerabilities in Microcontrollers

Countermeasures for replay attacks:

- Encryption
- Authentication from network protocol

Countermeasures for command injection attacks:

- Utilize open-standard RF protocols
- Encrypted communication protocols
- Authentication

Countermeasures for e-stop abuse attacks:

- Encrypted data transmission
- Limited access control to sensitive commands

Countermeasures for malicious re-pairing attacks:

- Code obfuscation makes it difficult to reverse engineer.
- Secure boot mechanisms, ensuring integrity of firmware

Countermeasures for malicious re-programming and remote attack vectors:

- Firmware rollback capabilities to restore to previous versions in case of compromise.

8.3.1.5. Conclusion

Due to the scope of this project, not every security concern can be implemented. The vulnerabilities are listed below based on the level of priority, being either high priority, medium priority, or low priority, with each choice being made regarding how likely the attack is to take place, its difficulty to pull off, and how reasonable it is given our projects environment.

High-Priority:

- Encryption

Medium-Priority:

- Replay Attacks
- Control flow manipulation attacks
- Fault attacks

Low-Priority:

- Side-channel attacks
- Malicious repairing attacks
- Malicious reprogramming and remote attack vectors

Encryption makes the most sense to implement, as it prevents attacks and vulnerabilities on both microcontrollers and radio frequency modules and is a minimum level implementation of security that meets the project environment. A form of encryption covers against a lot of different possible attack vectors, providing a strong basis that would deter most beginner level attackers. Due to this project likely being used in a class and project environment, attempting to provide too much security takes away other features this project aims to implement, especially if the countermeasures to some of the attack vectors were implemented in hardware with the limited die-space that the Efabless process provides. Even software implementations would somewhat hinder the user from using this project to its fullest extent. Since this project likely won't be sold commercially as well and serves as a learning platform, noting the security concerns while providing users options in securing the device itself allows the user to make educated decisions on what level of security best fits what the user wants to experiment with when using this device.

9. Team

Complete each section as completely and concisely as possible. We strongly recommend using tables or bulleted lists when applicable.

9.1. TEAM MEMBERS

1. Nolan Eastburn
2. Noah Thompson
3. Nathan Stark
4. Ibram Shenouda
5. Ethan Kono
6. Will Custis

9.2. REQUIRED SKILL SETS FOR YOUR PROJECT

- Designing digital components with HDL
 - o Verilog will be used, but VHDL experience useful as well
- Testing digital components with testbench simulations
- Writing programs using C for embedded systems
 - o Memory mapping will be used for most peripherals
- Using Git for version control
- Using NGSpice for analog simulation

9.3. SKILL SETS COVERED BY THE TEAM

- All team members have experience in some HDL (VHDL or Verilog) and C for embedded systems from previous ISU coursework
- Nathan and Nolan have experience in unit testing from previous work experience
- All team members have experience in Git from prior courses
- Ibram and Noah have experience in NGSpice

9.4. PROJECT MANAGEMENT STYLE ADOPTED BY THE TEAM

Our team is going with a Waterfall approach for project management. This was chosen because our project is strongly interconnected and we have a hard, inflexible deadline at the end of the project. Because of this, Waterfall seemed to be the best choice since it has a strong structure for design, implementation and testing.

9.5. INITIAL PROJECT MANAGEMENT ROLES

Team Organization - Noah

Project Management – Will

Analog Design Lead - Ibram

Digital Architecture Developer - Nathan

Digital Architecture Developer - Nolan

Security Architecture Developer – Ethan

9.6. TEAM CONTRACT

Team Members:

- | | |
|--------------------------|--------------------------|
| 1) <u>Nolan Eastburn</u> | 2) <u>Noah Thompson</u> |
| 3) <u>Nathan Stark</u> | 4) <u>Ibram Shenouda</u> |
| 5) <u>Ethan Kono</u> | 6) <u>Will Custis</u> |

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:

Weekly Advisor/Client Meeting:

Monday 2:00pm Durham 353

Weekly General Team Meeting:

Saturday 10:00 AM TLA

Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

Team Discord

Microsoft Teams (Advisor/Client)

2. Decision-making policy (e.g., consensus, majority vote):

Majority rules (4 members needed to approve)

At least 4 team members need to be present to make a decision

3. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Minutes will be kept in OneDrive folder, which is shared amongst the team

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

Members are expected to attend 80% or more of all team meetings.

If unable to attend, notify as far in advance as possible.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Complete all tasks by deadline.

Each member is responsible for their assignment tasks.

Communicate about roadblocks and give advanced notice if deadline is likely to slip.

3. Expected level of communication with other team members:

Communicate regularly at meetings and on Discord.

Bring up all issues and roadblocks encountered in a timely manner.

If you know the answer to a question that is posted, promptly answer it.

4. Expected level of commitment to team decisions and tasks:

All team members are expected to follow through on obligations, even if they did not vote in favor of it.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
 - a. Team Organization - Noah
 - b. Project Management - Will
 - c. Analog Design Lead - Ibram
 - d. Digital Peripheral Lead - Nathan
 - e. CPU/Memory Architecture Lead - Nolan
 - f. Software Lead - Ethan

2. Strategies for supporting and guiding the work of all team members:

Post solutions to common problems in Discord or OneDrive so everyone can see it.

Time at meetings will be devoted to discussing issues.

3. Strategies for recognizing the contributions of all team members:
 - a. Discussions of what was done the week prior
 - b. Presentations to our faculty advisor

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
 - a. Noah:
 - a. Embedded C code
 - b. VHDL and Verilog
 - c. Micro Processor integration
 - b. Nolan:
 - a. Knowledge of C programming for MCUs
 - b. Decent experience in VHDL, basic experience in Verilog
 - c. Intermediate Git knowledge
 - d. Have experience with system and unit testing
 - e. Can adapt to many different codebases quickly
 - c. Nathan:
 - a. HDL experience in VHDL
 - b. Some verification experience for VHDL
 - c. C/C++ programming for MCUs

- d. Ibram:
 - a. Analog Design
 - b. PCB design and testing
 - c. Embedded C
 - e. Ethan:
 - a. Security compliance and national standards (NIST)
 - b. Networking & Protocol Security
 - c. Wireless Security
 - f. Will:
 - a. Experience Programming in C
 - b. Some experience with encryption methods
 - c. Network and Wireless Security
2. Strategies for encouraging and supporting contributions and ideas from all team members:
 - Everyone is encouraged to contribute any ideas and will always be heard by the rest of the team.
 - Ideas will be taken into consideration by the team and will be discussed thoroughly to promote the inclusion of new concepts.
 3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
 - Individuals who feel that there are inclusion issues should bring up any problems with the team first.
 - When a problem is brought to the team, everyone is required to review any concerns together as a group.
 - If the problem consists of the individual should go directly to the faculty advisor for advice or potential guidance for themselves and the team as a whole.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
 - a. Finish the documentation report
 - b. Design the high-level hierarchy of the project
 - c. Implement some of the components that will go into the project
 - d. Show proof-of-concept of the project as a whole
 - e. If all goes well, develop a basic prototype of the project (this may be unrealistic depending on the complexity)
2. Strategies for planning and assigning individual and team work:
 - a. Task breakdown
 - b. Assign tasks with members with the most applicable skillsets
 - c. Flexible to reevaluate and reassign tasks based on needs.
3. Strategies for keeping on task:
 - a. Consistent weekly meetings with the team and professor
 - b. Acknowledging individual contributions

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

Issues will first be discussed during the team meetings or via Discord. All team members are expected to discuss issues in good faith.

2. What will your team do if the infractions continue?

If the issue cannot be resolved amongst the team, an instructor or faculty advisor will be informed to try and mediate the dispute.

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*

b) *I understand that I am obligated to abide by these terms and conditions.*

c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Nolan Eastburn DATE: 9/19/2024

2) Noah Thompson DATE: 9/19/2024

3) Nathan Stark DATE: 9/19/2024

4) Ibram Shenouda DATE: 9/19/2024

5) Ethan Kono DATE: 9/19/2024

6) Will Custis DATE: 9/19/2024