Open-Source Radio Microcontroller Design Document

CPR E 491
Team: sdmay25-27
Team Members: Ibram Shenouda, Nolan Eastburn,
Ethan Kono, Nathan Stark, Noah Thompson, Will Custis

Introduction

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.

Intended Users

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.

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.

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.

Requirements, Constraints, And Standards

Requirements & Constraints

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.
  - 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 will be able to transmit over the 915MHz zigbee broadcast band.
    - The carrier signal will be generated by an analog PLL to 915MHz using a reference oscillator.
    - The input signal from the processor will go through a DAC before being modulated with the carrier signal using quadrature phase shift keying
    - The modulated signal will be sent to a power amplifier and transmitted from an antenna.

Testing Requirements

- The MCU shall have a testing interface that enables a user to fully test the unit.
    - 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.
    - The test plan shall provide steps to test the electrical characteristics of the system.
        - Supply voltage, power consumption, output voltages, slew rates
    - 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. The internal signals will be outputted to the pads via transmission gates (PMOS and NMOS in parallel) which are basic on/off switches.
    - The wave forms can then be used to confirm the components behave as expected.

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).

Resource Requirements

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

Engineering Standards

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.

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.

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.

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.

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.

Project Plan

Project Management/Tracking Standards

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.

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.

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.

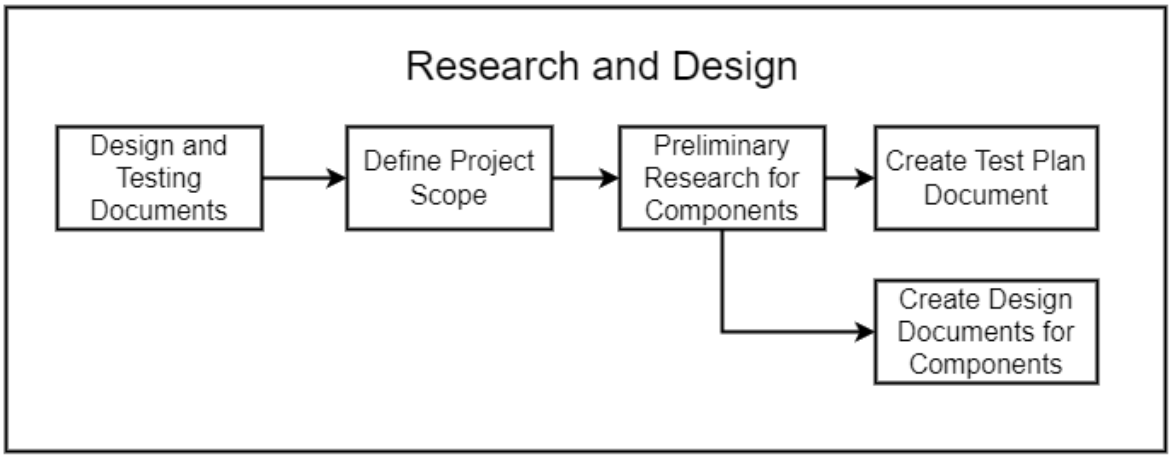

Figure 1: Research and Design Task Decomposition

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.

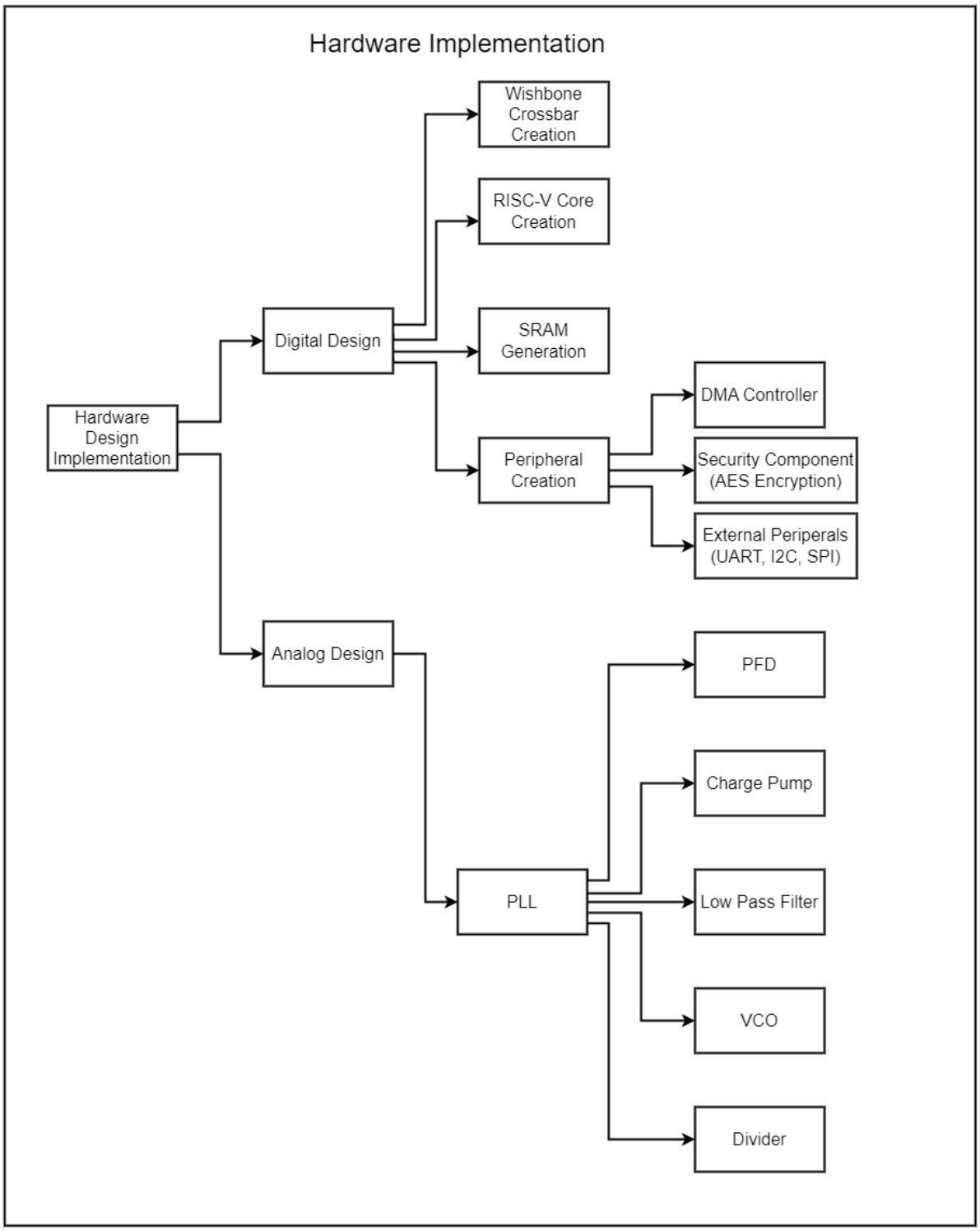

Figure 2: Hardware Implementation Task Decomposition

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.

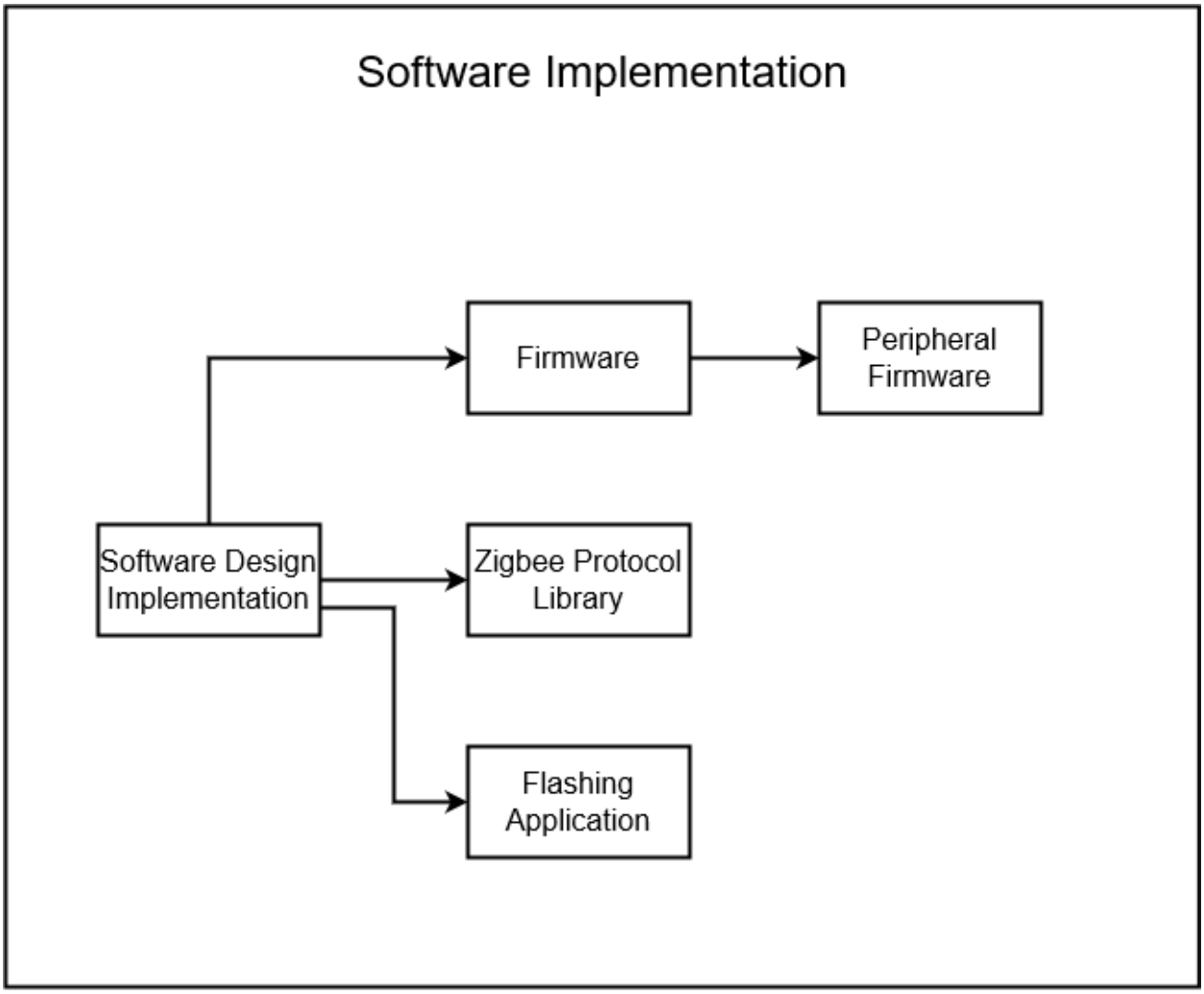

Figure 3: Software Implementation Task Decomposition

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.

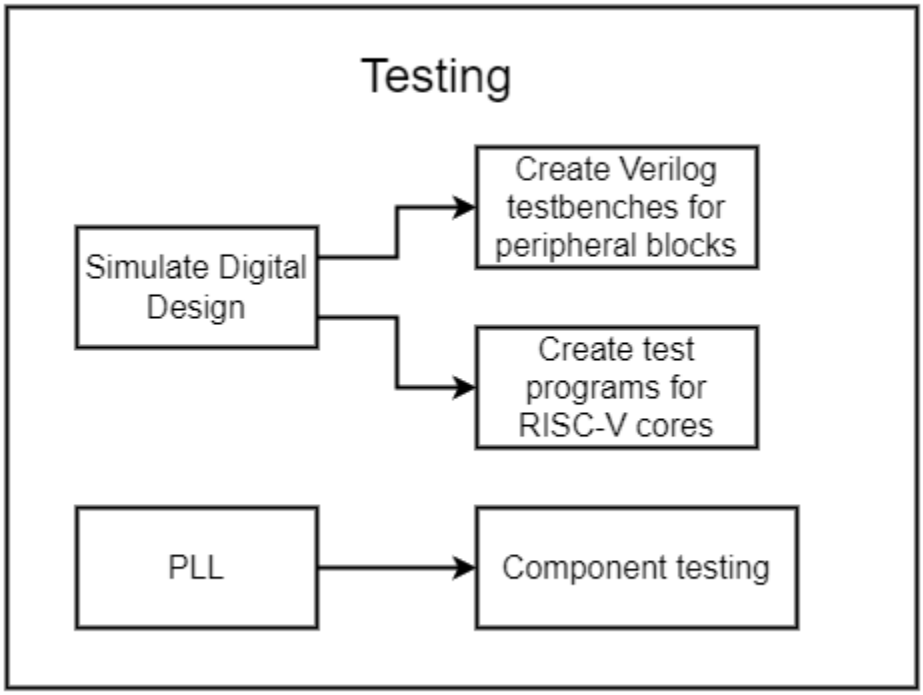

Figure 4: Testing Task Decomposition

Project Proposed Milestones, Metrics, and Evaluation Criteria

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

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

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.

Gantt Chart for Fall 24

Weeks/Task		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Project Scope																	
Research available tools	OpenRAM																
	RISC-V Core Generation																
Wishbone Crossbar Creation																	
RISC-V Core Creation																	
SRAM Generation																	
DMA Controller																	
Security Component	AES 128-bit Encryption																
External Peripherals																	
Fabrication																	
Radio Frequency Components	Bluetooth																
	ZigBee																
Phase Locked Loop (PLL)	Research																
	Phase Frequency Detector																
	Charge Pump																
	low Pass Filter																
	Voltage Controlled Oscilator																
	Divider																
	Testing																
Create Design Documents																	
Create Testing Documents																	
Testbench Creation	Peripherals																
	RISC-V Core																

Key:	
	Initial Design
	Research
	Implementation
	Fabrication

Figure 5: Fall 24 Gantt Chart

Gantt Chart for Spring 25

Risks and Risk Management/Mitigation

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.

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.

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.
Implement Bluetooth Protocol	100	Implementing the Bluetooth protocol to work with our hardware implementation (or segments of our hardware implementation) is extremely difficult due to the many layers in the

		Bluetooth protocol. This will require a substantial amount of effort.
Implement ZigBee Protocol	100	Implementing the ZigBee protocol to work with our hardware implementation (or segments of our hardware implementation) is extremely difficult due to the many layers in the ZigBee protocol. This will require a substantial 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"**
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.

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.

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.