Open Source Bluetooth Low Energy MCU for Fabrication

Design Document

SDDec26-10

Client: Dr. Henry Duwe

Advisor: Dr. Henry Duwe

Zane F. Salti

Aiden Han-Lindemyer

Jackson B. Doyle

Daniel Pytel

Seth I. Klaassen

Parker Pederson

Tyler Bibus

Akash Gojuru

sddec26-10@iastate.edu

https://sddec26-10.sd.ece.iastate.edu/

Executive Summary

Open source radio hardware is not commonplace and students who would like to learn about it have few options. The goal of this project is to create an open source Bluetooth Low Energy system on a chip for use by ChipForge students. The design must manufacturable on the SkyWater 130nm open source process. It must be able to interact with other Bluetooth Low Energy devices. It must also adhere to the Bluetooth Low Energy 4.0 specification. The design utilizes a predesigned frame that contains a core that will run Zephyr, which provides the software driver for a Bluetooth device. Inside of the frame, the design consists of a digital controller which is responsible for timing and managing packets, as well as an analog radio driver that utilizes gaussian frequency shift keying to send information to other Bluetooth devices. Thus far, a model of the digital design has been created in Python as a standard to verify the design. On the radio side, a Simulink simulation has been run to verify that our architecture will work to drive the radio. A testing plan has been developed to ensure that the design meets the Bluetooth Low Energy requirements. This consists of a verifying the software model against known good software drivers. Once the design is implemented in hardware, an FPGA with transceiver will be used to ensure that the design can communicate with known good Bluetooth Low Energy devices. Moving forward the digital design will be implemented in SystemVerilog and the analog side will undergo schematic design and layout. Once the hardware has been fully tested in simulation digital and analog sides will be integrated together to form the final design which will be hardened into the pad frame and sent to chipIgnite for fabrication in December 2026. These chips should return to the university in May 2027 for testing.

Learning Summary

Development Standards & Practices

- BLE 4.0 protocol **[1]**
- IEEE 13213-1994 - Control and Status Registers (CSR) architecture for the host and controller.
- IEEE 802.15.4-2024 - Low Rate Wireless Networks, referenced for PHY layer and general low bitrate architecture
- Agile development pattern - weekly sprints with advisor check ins with targeted deliverables and updates
- GitLab for version control
- Open-source toolflow for ASIC design on Skywater 130nm PDK - xschem, Magic, NGspice, Klayout
- Standard ASIC development practices - gate-level simulation, RTL equivalence checking, DRC, parasitic extraction

Summary of Requirements

- The design shall be able to communicate with other Bluetooth Low Energy 4.0 or newer devices.
- The design shall target 1 MS/s rate.
- The design shall be manufactured on the SkyWater 130nm process.

- The design shall be open source.
- The design shall include a virtual Bluetooth Low Energy testing environment using python and external libraries (i.e. Bumble and BabbleSim).

Applicable Courses

- CPRE 3080: Operating Systems: Principles and Practice
- CPRE 3810: Computer Organization and Assembly Level Programming
- CPRE 4870: Hardware Design for Machine Learning
- CPRE 4880: Embedded Systems Design
- COMS 3090: Software Development Practices
- EE 3210: Communication Systems I
- EE 4220: Communication Systems II
- EE 3300: Integrated Electronics
- EE 4350: Analog VLSI Circuit Design
- EE 4650: Digital VLSI Circuit Design
- EE 4140: Microwave Engineering

New Skills/Knowledge

- Bluetooth Low Energy 4.0 protocol: state machine, PDU formatting, CRC, whitening, channel hopping.
- Fractional-N PLL design targeting 2.4-2.48 GHz found, but VCO will likely need to be off-chip due to Sky130 limitations.
- Cocotb for Python Simulation environment will enable the environment to be reused for the Verilog RTL section of testing.
- Bumble and BabbleSim integration with Zephyr serving as a control device provides best gold-standard model for our design.
- Open-source tools for ASIC design, specifically xschem, Magic, and NGspice.

Glossary

ADC: Analog-to-Digital Converter
ASIC: Application-Specific Integrated Circuit
BabbleSim: Open source simulation software designed to emulate the physical layer of radio protocols including BLE
BLE: Bluetooth Low Energy
Bumble: Open source simulation framework designed to emulate the host or controller of a Bluetooth stack
CRC: Cyclic Redundancy Check
FSM: Finite State Machine
GFSK: Gaussian Frequency Shift Keying modulation used in BLE communication
HCI: Host Controller Interface
HDL: Hardware Description Language such as Verilog or SystemVerilog
IC: Integrated Circuit
IP: Intellectual Property core (reusable hardware module)
IQ: In-phase and Quadrature signal components used for RF demodulation
LL: Link Layer
LO: Local Oscillator used in RF frequency mixing
MCU: Microcontroller Unit
PDU: Protocol Data Unit
PHY: Physical Layer
PLL: Phase-Locked Loop
RF: Radio Frequency
Ring Oscillator: Oscillator constructed from an odd number of inverters connected in a loop
RTL: Register Transfer Level
RX: Receive path of a communication system
Sky130: SkyWater 130 nanometer open-source semiconductor fabrication process
SPDT: Single-Pole Double-Throw switch used in RF routing
SRAM: Static Random Access Memory
TX: Transmit path of a communication system
VCO: Voltage-Controlled Oscillator
VM: Virtual Machine

Contents

1 Introduction

1.1 Problem Statement

Students and hobbyists would benefit from an open-source ASIC wireless microcontroller to both learn from, modify, and implement. Most radio microcontroller units (MCUs) are closed source, making it incredibly difficult for users to study their design or fabricate themselves. Notably, no open-source, silicon-proven Bluetooth Low Energy (BLE) ASIC microcontroller currently exists. At student led organizations, there is a need for additional ASIC resources to learn from, particularly for wireless components. Having an open-source, fabrication-ready BLE microcontroller would allow students, hobbyists, and academics to gain direct experience with the entire ASIC development process with a wireless component. Our team will address this gap in open-source ASIC by developing a BLE microcontroller for fabrication, with full documentation and a silicon-proven design.

1.2 Intended Users

1.2.1 Hobbyists

Hobbyists use platforms such as ESP32 or Raspberry Pi because they are affordable and easy to program. However, these platforms are closed at the silicon level, preventing users from studying, reusing, or modifying the underlying architecture. An open-source BLE microcontroller ASIC would provide a full reference design, from RTL through layout, that hobbyists can inspect, modify, and use as a starting point for their own ASIC projects. Unlike closed-source alternatives, an open-source design gives hobbyists visibility into the analog, digital, and RF subsystems that commercial vendors treat as proprietary. By lowering the barrier to ASIC design, the project addresses the broader problem of limited access to silicon development resources.

1.2.2 Students

Students studying VLSI often only work with digital design examples or use closed-source evaluation boards, creating a divide between their experience and the complete process. An open-source BLE microcontroller provides students with exposure to the full SkyWater 130nm PDK in a RF design context, which is a context rarely covered by current open-source tapeout examples. Clear documentation will enable students to learn specific skills such as floor planning, parasitic extraction, and verification in a real fabricatible context rather than purely digital current academic exercises.

1.2.3 Student Organizations (ChipForge)

Student organizations, such as ChipForge at Iowa State University, provide student engineers a place to experience hands-on ASIC design. Currently, there are no open-source ASIC projects available to these organizations that incorporate radio elements. Any existing RF designs are closed source or rely on external proprietary components, limiting the learning, modification, and modularity. This project would provide these organizations a documented, silicon-ready BLE ASIC that can be carried forward - enabling additional functionality or spin-off designs for future tape-outs. This project will fulfil student organization's needs for open-source wireless geared ASIC designs.

2 Requirements, Constraints, And Standards

2.1 Functional Requirements

2.1.1 Bluetooth Low Energy 4.0

- The device shall transmit and receive data using BLE 1M 4.0.
- The device shall transmit and receive at 1MS/s.
- The device shall implement the following states:
  - Standby: The device shall not send or recieve packets.
  - Scanning: The device shall listen for packets and provide advertsing reports to the host device
  - Advertising: The device shall send information about itself and its connection properties to devices within range
  - Init: The device shall establish a connection with a device designated by the host after recieving advertising packets from said device
  - Connection: The device shall send and receive data with a single device determined by the host
- The device shall implement BLE channel hopping algorithm 1
- The device shall support Gaussian Frequency Shift Keying modulation
- The device shall support an output range of 2400 MHz to 2483.5 MHz
- The device shall support 40 channels with a 2 MHz gap starting at 2402Mhz to 2480 MHz
- The device shall not have a center channel frequency drift greater than 50 KHz
- The device shall modulate further than 185kHz
- The device shall have a zero crossing error less than 1/8th the symbol period
- The device shall implement the following subset of BLE features:
  - c.1: LE controller supports transmitting packets.
  - c.2: LE controller supports receiving packets.
  - c.3: LE controller supports connection state.
  - c.15: LE controller supports transmitting scannable advertisements.
  - c.36: LE controller supports central role or supports both peripheral role and LE Feature (channel classification).
  - c.59: LE controller supports central role.
  - c.94: LE Create Connection or LE Extended Create Connection command is supported.
  - c.97: Mandatory if Advertising State is supported.
  - c.98: Mandatory if Scanning State is supported.

[1]

2.1.2 Virtual Environment Requirements

- The virtual environment shall implement a **Conductor** with the following:
  - The Conductor shall include global variables to control the routing of external binaries.
  - The Conductor shall include global variables to control both the length of the simulation as well as how long each time step lasts.
  - The Conductor shall include arguments that allow the previous global variable to be set.
  - The Conductor shall include arguments that define how many devices are made, along with their type and slot.
  - The Conductor shall start the global PHY Crossbar.

    - The Conductor shall instantiate both DUT and third-party devices in the same environment, defining their role as one of the following:
        - Advertiser: The device shall send advertising packets on channels 37, 38, and 39
        - Scanner: The device shall scan channels 37, 38, and 39 for advertising packets
        - Initiator: The device shall scan for a device with a specific ID and maintain a connection with the device using BLE 4.0 protocol in line with our functional feature set.
    - The Conductor shall time step each DUT device through the following process:
        - Updating the inputs for each module in the device, simulating the propagation delay and pipelining of our design
        - Run each module, emulating one clock cycle across the device
        - Step the PHY crossbar to the next time slot, allowing the devices to send and receive packets
    - The Conductor shall teardown the virtual environment at the end of the simulation to prevent zombie processes
- The virtual environment shall include a PHY Crossbar to implement a physics-accurate wireless environment simulation.
- The virtual environment shall include a Design-Under-Test (DUT) device with the following characteristics:
    - The DUT device shall include a emulated known-good Host to send and receive commands.
    - The DUT device shall include an interface to the PHY crossbar.
    - The DUT device shall have clear logging and exposed I/O for both the digital controller overall but also individual modules.
    - The DUT device shall implement modules aligning with the digital design.
    - The DUT device must be able to interact with a known-good controller and establish and maintain a connection.
- The virtual environment shall be well modifiable using encapsulation and low coupling for future development.
- The virtual environment shall have easily accessible I/O for Verilog testbench generation.

2.2 Constraints

2.2.1 Open Source Design Tools

In order to facilitate our users' needs to see the internal working of the ASIC design process. Each component of our design must creating using readily available open source design tools. Since prehardened designs are relatively limited for the SkyWater 130 process contributing entirely open source modules to the current library of modules will allow for easier development in the future.

2.2.2 Fabrication Constraints

- The device shall be fabricated on the SkyWater 130nm process
- The device shall pass all design rules check for the SkyWater 130nm process
- The device shall pass foundry precheck for the SykWater 130nm process
- The device shall use a padframe supplied from SkyWater

2.3 ENGINEERING STANDARDS

While working on the design process several different standards came up that could be applicable. The initial set of papers viewed were BLE 4.0, IEEE 13213-1994, IEEE 802.15.4-2024, and IEEE 1890-2018. BLE 4.0, IEEE 13213-1994, IEEE 802.15.4-2024 appeared to be useful applicable directly while IEEE 1890-2018 was redundant as parity checking is already defined in BLE 4.0.

2.3.1 BLE 4.0

The main engineering standard that will be followed will be BLE 4.0. This provides the standards that are required to be followed in order to enable our main requirement–wireless Bluetooth communication. The standards themselves are very detailed, providing feature lists that are mandatory or optional. Wireless communication has strict timing, security, and reliability needs that have to be met to be usable in real-world environments.

2.3.2 IEEE 13213-1994 - Control and Status Registers (CSR) Architecture for microcomputer buses [2]

This standard provides a method for how peripherals on memory-mapped buses should behave and decipher data. This standard is designed to prevent interference between third party designed peripherals that are positioned on the same bus. Since the BLE controller must be implemented on an integrated circuit, these requirements will help determine the interface design between the management core's memory bus and the BLE controller.

2.3.3 IEEE 802.15.4-2024 - Standard for Low-Rate Wireless Networks [3]

This paper provides standards for Low-Rate wireless networks, with a focus on low energy communication. The paper provides standards on both the physical layer (PHY) and medium access control for low-data-rate wireless devices that are power constrained. While this project is not power constrained to the same extent, an expectation of BLE is for it to be "Low Energy". This project will incorporate some design standards provided in the PHY as the PHY is not governed as strictly in BLE 4.0 standards.

3 Project Plan

3.1 Project Management/Tracking Procedures

3.1.1 Project Management Style

We will be using the agile methodology, were we focus on completing repeated small deliverables per team. this is because each part of our design is so large that we need to break it up into multiple parts to make it more manageable

3.1.2 Progress Tracking

The team uses Git to manage all source code and project documentation, with each member regularly pushing their work to a shared repository. Team communication, weekly status report, in-class activity updates, and sharing presentation slides are managed through Microsoft Teams. Each team member enters their total time spent per week into a shared Excel sheet.

3.2 Task Decomposition

There are two main ways this project was broken into, with each having several subcategories being broken down even further. This arrangement will allow members with different strengths in digital and analog design to contribute effectively to individual actionable tasks.

3.2.1 Digital Subsystem Tasks

This list is a near linear progression aligned with our sprint schedule.

Digital Tasks and Subtasks:

1. Top Level Planning
   - Host Controller Interface
   - Link Layer (PHY interface)
   - Feature list and Specification
2. Software Simulation
   - Host Simulation Integration
   - PHY Simulation Integration
   - Logic Validation and decomposition
3. HDL Digital Design Implementation
   - FSM implementation for Link Layer
   - Individual components in LL and HCI
   - Synthesis of components and top level design
4. RTL Verification using FPGA
   - Initial synthesis and deployment on FPGA
   - Test environment development
   - Final verification of RTL
5. Pre-Layout
   - Gate-level simulation
   - Equivalence checking (RTL vs synthesized netlist)
   - Static timing analysis
6. Post-Layout
   - Parasitic analysis
   - Post-layout static timing analysis
   - Post-layout gate-level simulation
   - Power analysis
7. Sky130 Implementation and Tapeout
   - Verify design is Sky130 compliant
   - Floor-planning
   - Place-and-Route
   - Clock tree synthesis
   - Final Tapeout

3.2.2 Analog Subsystem Tasks

1. Top Level Planning
   - Decide on Antenna
   - General architecture implementation
2. MATLAB Simulation
   - General Design Verification
3. PLL Design
   - VCO
   - Charge Pump
   - Loop Filter
4. LNA Design
   - Noise Analysis

5. Mixer
6. ADC
7. Filters
8. XSCHEM Simulation
   - Schematic building
   - Observe non-ideal behavior
9. MAGIC Layout
   - Build layout for each component and test independently
   - Parasitic Testing
   - Combine into Caravel/Caravan template

3.3 Project Milestones, Metrics, and Evaluation Criteria

Since the project is split into two sub-teams that can proceed relatively independently of each other, it makes sense to split the milestones by sub-team as well.

3.3.1 Digital Milestones

Many of these milestones require the creation of virtual test sets and test benches. These milestones will be slightly vague because each hardware module will have different acceptance criteria. In order to achieve an implemented IC with the digital design the following milestones need to be completed in order.

3.3.1.1 Integrated Software Simulation Verification

The first milestone is to have a complete simulation of the system utilizing bumble sim **[4]** to create a fake software host and babble sim **[5]** to create fake physical layer in order to simulate over the air traffic. Since each of these simulators have been externally developed to test Bluetooth devices they will serve as external verification of a software simulation of the entire system. The device must be capable of supporting a Bluetooth LE connection from both bumble and babble.

3.3.1.2 Module Digital RTL Simulation Verification

Once every module has been translated into hardware each module will have a test bench which will match it's functional description. Each module will pass its individual test bench.

3.3.1.3 Integrated Digital RTL Simulation Verification

Once each module passes its test bench the entire system will be integrated and it will be run through a sequence of inputs generated by capturing the previous simulation data. The results will be compared to the results from the integrated software simulation to ensure correctness.

3.3.1.4 Post Layout Parasitic Extraction and Timing

Once the design has been hardened into a layout the design will be run through post layout simulation to ensure functional correctness as well as generate a timing analysis. This milestone presents a risk of failure to close timing.

3.3.2 Analog Milestones

These milestones are concerned with the design, testing, and verification of various analog components. Once completed the resulting components will be used to assemble a working phy layer for the BLE radio microcontroller.

3.3.2.1 High Level System Model

This step will be completed once a system level model of the phy layer, including allocated gains, is up to desired specifications.

3.3.2.2 Component Parameterization and Schematic Design

This step will be completed once all scoped phy components have been designed and parameters determined. This will be realized in the form of completed low level models and schematics testable within the open source tool-flow.

3.3.2.3 Components Ready For Layout

The components of the phy layer can be considered ready for layout when they have gone through extensive performance tests and have achieved desired performance.

3.3.2.4 Layout Completion

The layouts are considered complete once they pass the required tests and meet sizing requirements along with any process requirements for tape out. Required tests would include passing of DRC (design rule check) and LVS (layout vs schematic). DRC passes if no process rules are broken in the design. LVS tests if the layout matches the expected form from the schematic. It also tests if the design can be realized.

3.3.2.5 Post Layout Parasitic Extraction and Timing

Once layout is complete additional tests must be run to ensure functionality. This step may face difficulties and failures regarding timing and possible parasitic issues

3.3.3 Final Integrated Simulation

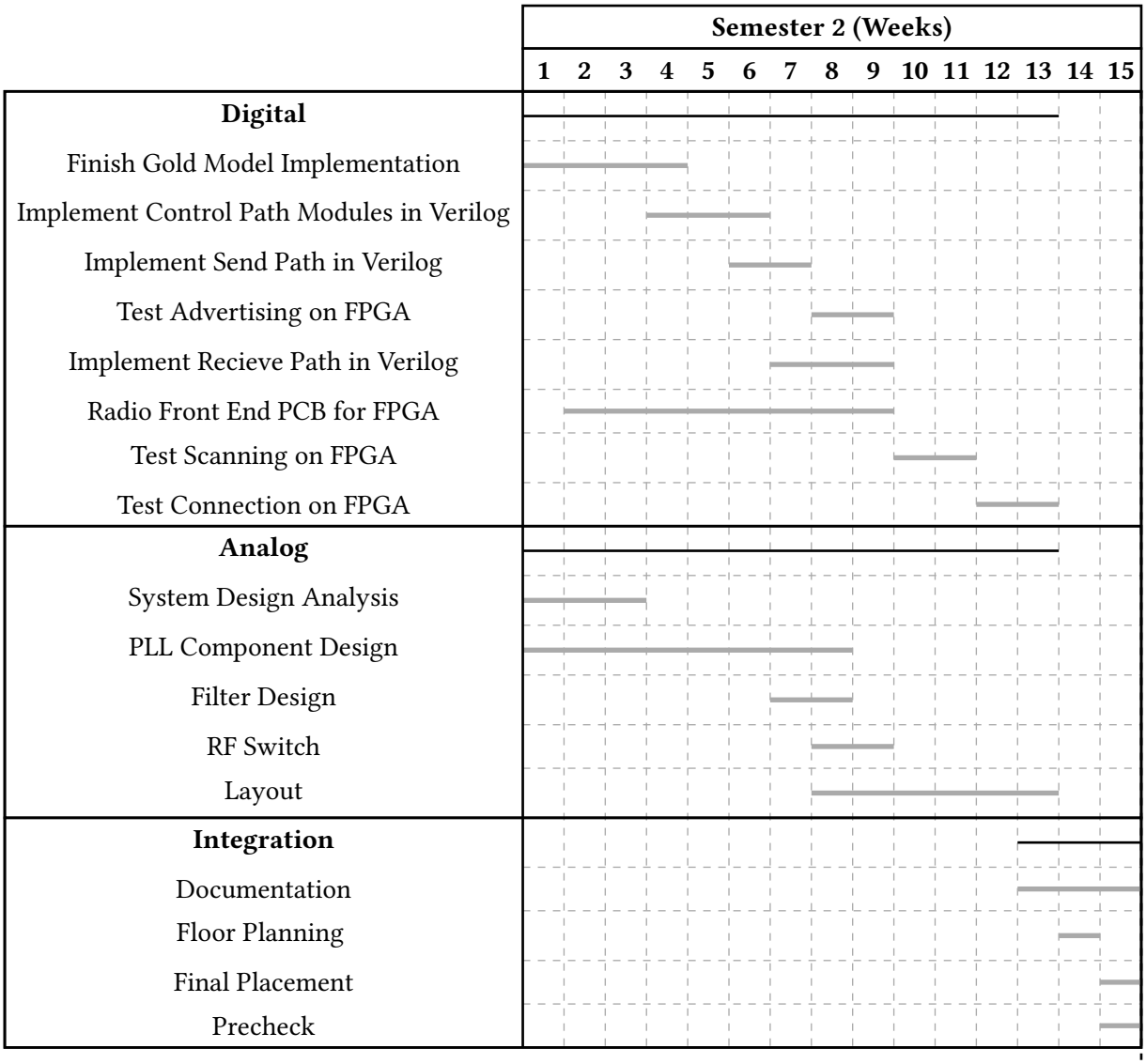
Once all analog and digital milestones have been completed the systems will be integrated together and the final system will be tested by running a post layout parasitics simulation. This will be used to verify timing as well as for function.

3.3.4 Tapeout Submission

Once the final simulation is complete with both the digital and analog ends integrated it will be submitted to the ChipFoundry chipIgnite CI2612 shuttle.

3.4 Project Timeline/Schedule

Gantt Chart (Task vs. Deliverable)



Schedule Description

The project follows a hybrid agile development model. While high-level phases are planned sequentially, development within each phase is iterative and parallelized across subsystems.

Three major phases emerge:

Phase 1 (Weeks 1–4) Foundation & Scoping: Focused on BLE specification analysis, toolfow onboarding, and defining system scope. This phase experienced delays due to toolflow issues but established critical project direction.

Phase 2 (Weeks 4–10) Architecture & Simulation: Emphasis on building a simulation-first workflow using Bumble and BabbleSim. This phase is central to the project, enabling a "golden model" for validating digital logic before RTL implementation.

Phase 3 (Weeks 10–16) Implementation & Integration: Transition from simulation to hardware-oriented design, including RTL development, PHY modeling, and system integration. Verification runs continuously during this phase.

Key Milestones and Deliverables
Week 4: BLE requirements and reduced feature set finalized
Week 6: Top-level architecture and interfaces defined
Week 8: Functional simulation environment operational
Week 11–12: Core Link Layer and HCI functionality simulated
Week 14: Integrated system (digital + PHY models)
Week 16: Final validated design and presentation

Schedule Justification

The schedule reflects lessons learned from early progress:

Tooling and environment setup required more time than expected, so early weeks include buffer. Simulation is treated as a first-class deliverable, not just a development tool, since it enables validation against BLE standards. Parallel development (digital + analog) reduces bottlenecks and aligns with team specialization.

Critical path items include:

Link Layer state machine implementation PHY feasibility and PLL design Integration between simulation and RTL

By prioritizing early validation and overlapping subsystem work, the schedule reduces the risk of late-stage integration failures and ensures steady progress toward a tapeout-ready design.

3.5 Risks and Risk Management/Mitigation

The root of most of the risk comes down to time. Since the timeline of the project is only six months and some of the verification simulations can take upwards of five days to run. This provides a hard limit to the number of attempts that can be made and revised.

3.5.1 Digital Risks

Risk	Severity	Probability	Description	Mitigation
Protocol Non-compliance	major	40%	It is possible that the final ASIC does not behave correctly under edge case conditions. This would be caused by an insufficient simulation that could fail to catch certain conditions.	In order to mitigate this we will test the design in implementation on an FPGA with an external physical layer. This will ensure that the system is compliant and can connect to real Bluetooth devices.
Timing Closure Failure	minor	30%	Once the design has been hardened it may have a minimum slack width that is long enough that the digital side cannot	Device pipelines can be split at a late stage to reduce the longest stages which should fix any timing issues

			drive the RF switch fast enough.	
Post Layout Simulation Failure	minor	10%	Once the initial layout is completed and combined with the analog side the timing analysis will be rerun. This may have tool issues because the interface to the analog side has some paths without defined registers	

Table 1: Digital Risk Assessment

3.5.2 Analog Risks

Risk	Severity	Probability	Description	Mitigation
Parasitics Lead to Altered System Behavior and Failure	major	10%	Unlikely to occur, especially if proper worst case testing is applied to the system, but effects can be severe if parasitics play a major part.	Ensure worst case parasitic testing is done thoroughly and that any possible sources of parasitic problems is addressed.
VCO Frequency Shift/ Phase Noise Degradation	minor	very low	The VCO is incredibly sensitive to changes in values. Prevalent parasitics could induce a shift in frequency outputs. This could result in a large enough change that the VCO falls out of line with desired BLE functionality.	Similarly to the prior example, taking care in the verification and extraction steps can reduce the risk of this occurring.
Post Layout Simulation Failure	minor	20%	The completed analog component layouts will be combined with digital components and the timing analysis will be rerun. This may have performance issues or timing problems	ensure proper post-layout extractions and protocols.

Table 2: Analog Risk Assessment

3.6 Personnel Effort Requirements

Task	Assigned	Est. Hours	Justification
Define Project Scope	ALL	80	The scope of the entire system is quite large, so deducing what is feasible in the given timeframe is quite a task.
Research RF Switch	Jackson, Seth, Dan	15	Relatively simple component with set architectures, allows PHY members to get used to the toolflow.
Research PLL	Jackson, Seth, Dan	40	The PLL is a critical component of the system, likely being the most intensive part within the PHY realm.
Research VCO	Jackson, Seth, Dan	25	Untested VCO designs already exist in the process, so it must be determined whether to use and modify those or to use an external VCO.
Research Charge Pump	Jackson, Seth, Dan	25	Another key component of the PLL, with which the team has limited experience.
System Simulation In MATLAB	Jackson, Seth, Dan	25	Simulate an ideal form of the PHY protocol within given parameters.
Design all PLL Components	Jackson, Seth, Dan	80	Create each component and extract all parasitics.
Simulate PLL Components	Jackson, Seth, Dan	40	Simulate the behavior of each component.
Finalize PLL Components	Jackson, Seth, Dan	25	Observe if components work well in tandem.
Link Layer Architecture Design	Zane, Tyler, Parker, Aiden, Akash	35	State machine, data flow, interface definitions
HCI Design & Simulation	Zane, Tyler, Aiden, Parker, Akash	32	Command handling, register mapping, simulation infrastructure
Simulation Environment (Bumble/BabbleSim)	Tyler, Parker, Aiden, Akash	28	VM setup, integration, golden model creation
RTL Development (Digital LL Components)	Zane, Tyler	38	CRC, whitening, serializer, FSM implementation

PHY Architecture & Analog Design	Dan, Jackson, Seth, Zane	42	RF switch, PLL, VCO, system modeling, radio architecture
Toolflow Setup & Verification	All	24	ChipForge onboarding, debugging statfs/tool issues
System Integration	Aiden, Zane, Parker, Dan	26	LL ↔ PHY ↔ HCI interfacing and debugging
Testing & Validation	All	24	Simulation validation, BLE compliance checks, debugging
Documentation & Reporting	All	20	Weekly reports, design document, diagrams

Table 3: Personnel Effort Requirements

Summary

The total estimated effort for the project is **1024** person-hours, based on observed team time tracking and projected remaining work. Early project phases focused heavily on onboarding and toolchain setup, which introduced nontrivial overhead due to environment issues (e.g., SSH, statfs errors, VM configuration).

Effort is naturally divided into three major domains:

Digital (Link Layer + HCI): Significant effort due to complexity of BLE protocol implementation, state machines, and RTL development.

Analog/PHY: High effort due to system-level modeling, PLL/VCO design, and RF constraints.
Simulation & Integration: Critical cross-team effort enabling verification through Bumble and BabbleSim.

Compared to initial expectations, a larger portion of time has been allocated to simulation infrastructure and architecture definition, reflecting the need to validate correctness before hardware implementation. This aligns with project risk mitigation, as early verification reduces costly redesign later in the ASIC flow.

3.7 Other Resource Requirements

For this project we will need access to the FPGA's for testing, Virtual Machines for simulations, ASIC design software(chipforge), Babblesim, Bumblesim, Cadence, and the SkyWater Sky130 PDK process.

4 Design

4.1 Design Context

4.1.1 Broader Context

This project creates an open-source BLE microcontroller ASIC that includes the digital link layer and analog RF, made to be fabricated on the SkyWater SKY130 process. The communities that are most affected by this are students, hobbyists, and groups like ChipForge at Iowa State who want to work with wireless chips but have nothing open source to look at or start from. People who use BLE devices every day are also affected because almost every BLE chip out there is closed source, meaning nobody outside the company can look at how it actually works.

The big problem this project is trying to solve is that BLE is everywhere phones, fitness trackers, medical devices, and IoT devices, but all the chips that make it work are proprietary and locked down. This makes it really hard for students to learn from them, for security researchers to check if they are safe, and for smaller teams to build anything on top of them. This project tries to fix that by making a real, working BLE chip design that is fully open and documented so that anyone can use it, learn from it, or build on it in the future.

Area	Description	Project-Specific Considerations
Public health, safety, and welfare	BLE is used in a lot of medical devices like heart rate monitors and glucose sensors. Since most of these chips are closed source, it is hard for anyone outside the company to check them for security issues.	If the RTL is open, researchers and developers can actually look at the link layer and make sure it is working correctly, which is important when people are using these devices for their health.
Global, cultural, and social	Everyone uses BLE in their daily life but the chips that make it work are owned bu just a few big companies. students and smaller groups basically have no way to get involved with this technology at the hardware level.	This project lets students and hobbyists get their hands on a real BLE design they can actually learn from and change around without having to pay for licenses.
Environmental	Every time a new team wants to build a wireless chip, they basically have to start over from scratch, which wastes a lot of time and resources.	Because this design separates the HCI, link layer, and analog sections into their own modules, future teams at ChipForge can reuse the parts that already work instead of rebuilding everything again.
Economic	The market for BLE chips is dominated by big companies like Nordic Semiconductor, making it very difficult and expensive for small teams or students to get involved in wireless ASIC development.	By making a fully documented and verified BLE design freely available, this project lowers the barrier for anyone who wants to build on top of it without starting from zero.

Table 4: Broader Context

4.1.2 Prior Work/Solutions

4.1.2.1 Digital Prior Work

For digital implementation there are some software implementations of the HCI and LL. These range from simulation environments to full on software stacks that run on ESP32 development boards.

1. Zephyr BLE Controller

Zephyr is a popular open-source real-time operating system (RTOS) that includes a BLE stack. The BLE controller has software-implemented layers, which interface with existing proprietary hardware controllers. This implementation handles everything down through the link layer with features necessary for BLE like state machines and packet formatting. **[6]**

2. Nordic Semiconductor BLE nRF522832

The nRF52 Series by Nordic Semiconductor is one of the lower-end BLE SoC sets they offer. Nordic Semiconductor is a very popular company in the BLE space, providing chips like the nRF52 series while boasting high performance and a complete feature set. Nordic Semiconductor is a closed-source company however, which means users cannot meaningfully inspect, modify, or reuse hardware components in their own design.**[7]**

3. Apache NimBLE

NimBLE is Apache Mynewt OS' open-source BLE stack and has been ported to popular development boards such as the ESP32. This stack is strictly software implementation for the host and controller, similar to Zephyr. The lack of a analog PHY implementation means that NimBLE also relies on closed-source silicon for the final part. **[8]**

Digital: Pros and Cons

Feature	Prior Work (Zephyr, NimBLE, Nordic)	SDDEC26-10 Implementation
Pros	• **Zephyr:** Flexible, hardware agnostic. • **NimBLE:** Optimized for low memory footprint. • **Nordic:** High performance, mature timing closure.	• Full open-source availability enabling customizability. • Hardware RTL provides performance edge over software implementations. • Fabricatible on Sky130 Process.
Cons	• **Zephyr & NimBLE:** Software overhead; rely on black-box silicon. • **Nordic:** Complete closed-source architecture.	• Highly limited feature set due to the short tapeout timeline. • Design will be fresh and not matured. • More expensive to make changes and re-fabricate compared to software implementations.

Table 5: Digital Pros and Cons

4.1.2.2 Analog Prior Work

Several implementations of Bluetooth microcontrollers exist. However, none of these have fully open-source hardware, relying on either proprietary architectures or an FPGA.

1. Previous Design

This project is a continuation of a previous design from May of 2025, which was using Zigbee as the communication protocol. While that required more relaxed requirements, it was a still a highly intensive project which laid the groundwork for this one, albeit with no tapeout. **[9]**

2. PLL Design

There have been multiple attempts to design a Bluetooth-compliant PLL in the SKY130 PDK, although there is minimal evidence suggesting that any of them have been taped out. Given the open-source nature of the PDK, these will likely be used as reference for our own design.**[10]**

4.1.3 Technical Complexity

Since the project contains both digital and analog portions there will be components and subsystems pertaining to each. These components will be rigorously designed, tested, and implemented through their respective pipelines before being combined as a final product.

4.1.3.1 Digital Complexity

The digital side of this project will contain several different layers, with each layer housing multiple different sub-modules, with the end result being a controller that meets BLE specification. The strict BLE specification will result in a need for strong timing, state handling, and robust design. These requirements set by the BLE protocol provides a significant challenge while BLE is a ubiquitous technology.

HCI Layer Complexity: As the HCI bridges the gap between the host and the rest of the controller, there is a lot of difficult data management and command decoding/encoding that takes place.

Link Layer Complexity: The Link Layer connects the HCI and the analog interface, handling states, PDU generation, serialization, RF switch control, and de-serialization. All of these aspects provide their own challenges and need to meet the strict BLE specification to function properly.

4.1.3.2 Analog Complexity

This project will be a combination of both digital and analog components, the complexity of the analog components in particular is high. To both take in signals and send them out through the antenna requires careful balancing of phase, frequency, and noise through usage of a PLL, modulation, and filtering. To both transmit (send out signal at antenna) and receive (take in signal from antenna) requires switching logic in the form of an RF switch. In order for these signals to be translated into the digital domain requires some conversion aswell (ADC/DAC).

4.2 Design Exploration

4.2.1 Design Decisions

Since this project employs both a substantial analog and digital portion there will be shared overall decision types. These include determining methodology, architecture, and protocol. Firstly, the method used to achieve the desired output will be considered for both layers. This decision pertains to the system level/high level approach. The architecture must be chosen in order to reach desired specifications while considering optimization and efficient operation. Lastly is protocol, how these components will interact and how the controller will interact with both the digital and analog layers.

4.2.1.1 Digital Design Decisions

- Layer Interfaces

The BLE standard only describes how the host and controller interact. It is generally assumed by the standard that the HCI, the link layer, and the physical layer are colocated on the same chip. It therefore does not prescribe any communication method between the HCI, link layer and physical layer. Therefore, the design team will need to define what the interface between those three layers is. If the interfaces are designed poorly, it will result in slow development since teams will not know

where responsibility for processing ends for one layer and starts for another. The design of these interfaces also affects testing; if there are many logical paths that cross over the boundary it increases the complexity required in the tests and increases the chances of failure.

- Frame type

There are two options for frames that have management cores that are required to serve as a host. Caravel, which is used more frequently for projects and has better support due to the frequency of use. However, this frame lacks the unbuffered IO that may be required to pass RF off-chip. Caravan, by contrast, is used less frequently but is designed for analog IO.

- Module Breakdown

The BLE HCI and link layer specifications do not include information on how the module should process it's inputs and outputs. Therefore the team must decide how to break down the functional description into modules that can be described and then implemented.

4.2.1.2 Analog Design Decisions

- Transceiver Architecture

The transceiver serves to downconvert an RF signal into a baseband frequency that tends to be much lower than the frequency of the original signal. In this pursuit there are two primary approaches that one can take

  - Low-IF: Low-IF is a technique where the LO frequency is set to the carrier frequency which allows direct conversion to baseband, removing the need for a IF stage and reducing complexity. This approach is suitable in low power implementations
  - Direct Conversion: This technique employs a local oscillator that matches the carrier frequency. This approach simplifies the circuitry but can result in some issues such as DC offsets.

- PLL Topology

The PLL topology is highly dependent on operation range and design constraints. For our purpose it must operate BLE compliant. To do so we need operation range 2.4 GHz - 2.48 GHz. Our PLL will be a standard analog PLL with a phase detection block, a loop filter, a charge pump, and n-fractional divider as required by the needed reference frequency. Our PLL being n-fractional is of importance as we require fine control over the frequency for transmitting and receiving. In transmitting, symbols are modulated as frequency shifts in the hundreds of kilohertz range which is quite a fine adjustment. In receiving, we need our PLL set to the relevant channel's center so to be used as a local oscillator (LO) for down mixing. The most troublesome component is the VCO. To achieve the functionality we are aiming for a quadrature oscillator (LC VCO) is ideal. This would require Inductor implementation. The benefits of a quadrature oscillator is that it provides us with a quadrature signal to use for our IQ down conversion mixers.

- IQ Down-Conversion

A major design decision for the radio architecture was to go with IQ down conversion. Using in-phase and quadrature mixing (LO being supplied by the PLL) allows us to track the frequency as being positive or negative from the channel's center.

- Absence of Power Amplifier

While most radio architectures have a power amplifier, we decided without it to prioritize design and implementation effort on more essential components.

4.2.2 Ideation

4.2.2.1 Digital Ideation

Sim Ideation: The Sim Environment as a deliverable experienced a large amount of ideation, initially being considered as just C modules that could be individually tested. After some exploration of existing libraries and projects it was decided to use Bumble as a Python host and BabbleSim to simulate a wireless connection.

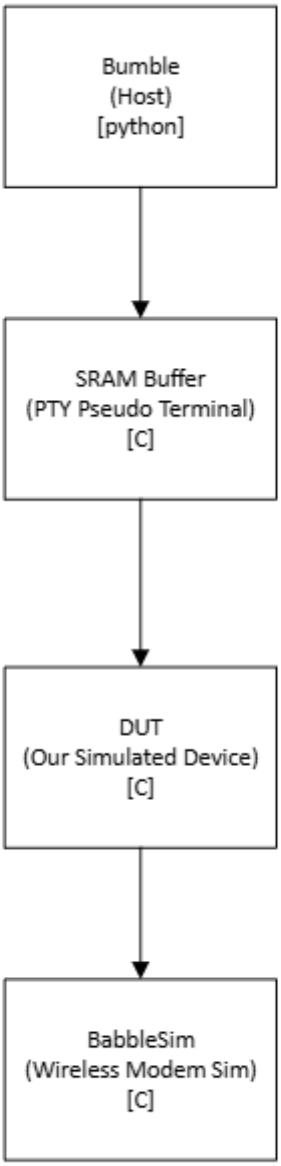


Figure 1: Original Sim Environment Design

It was discovered that C to simulate hardware–while often done–gets too cumbersome and results in delayed progress and reduced flexibility. With the experience learned from the previous failure we redesigned our simulation environment. The new design is similar, but uses a “conductor” to start and time step “devices” in a simulated environment. Python is now the dominant language, which will improve development speed, flexibility, third party library support to generate test benches, and

debugging. We still use Bumble as the host and BabbleSim as the crossbar, but the DUT is now python and has been encapsulated into a “device”. Maybe the largest improvement of this design is it allows us to use third party controllers–like Zephyr–to compare our design against.

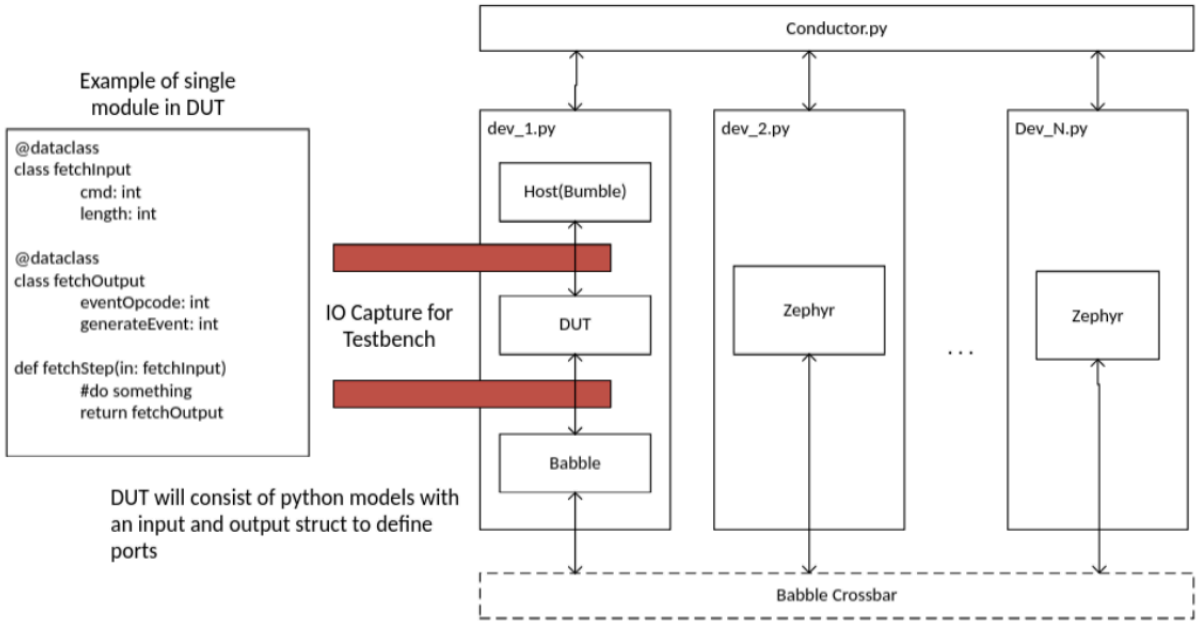


Figure 2: Updated Simulation Environment Design

Digital Design Ideation: The current design for our digital aspect can be seen in the top level design for the SoC. There was not much ideation behind that, as most of the modules are specified in behavior in BLE specification. This gives the design good credibility, but there are still many design choices that can be made when we develop the HDL modules for the controller.

4.2.2.2 Analog Ideation

While narrowing down the specifics of the controller has been an issue, the same general principles that apply to all wireless communication systems remain intact. It requires an antenna, some filters, an RF switch, some amplifiers, a synchronization circuit, a modulator, an ADC, and a demodulator. At the heart of our synchronization circuit is the PLL, which is perhaps the most complex individual component. The operating range is a given, but there are several PLL options which could work in the system. We elected to utilize a fractional-N PLL as it has the ability for finer tuning, faster switching speed, and less noise. This comes, however, at the cost of complexity and higher spurious emissions. This decision should be well worth it in the long run, considering that an imperfect fractional-N PLL likely still settles faster than any integer-N PLLs.

4.2.3 Decision-Making and Trade-Off

The design process followed a structured approach to evaluating different architectural options, focusing on balancing performance, simplicity, scalability, and feasibility within the project timeline. Decisions were made by comparing potential approaches across key criteria including implementation complexity, compatibility with the Sky130 process, ease of verification, and alignment with open-source goals. Each design option was evaluated based on factors such as development time, modularity, testability, and risk of failure during integration or tapeout. Options that introduced excessive complexity or relied on poorly understood components were de-prioritized in favor of designs that could be reliably implemented and verified within the available timeframe.

For the digital subsystem, this resulted in selecting a modular architecture with clearly defined interfaces between the HCI, link layer, and physical layer. The shared memory interface was chosen due to its compatibility with the management core and its flexibility in handling different types of data. Similarly, a reduced BLE feature set was selected to ensure that core functionality could be fully implemented and tested rather than partially implementing a larger specification.

For the analog subsystem, decisions were driven by feasibility within the Sky130 process and the availability of open-source design tools. Simpler and more well-understood architectures were favored to reduce the risk of failure in high-frequency operation.

The design reflects a balance between ambition and practicality. By prioritizing a working, verifiable system with clear documentation, the project achieves its goal of providing an open-source BLE microcontroller that can serve as a foundation for future work, even if it does not implement the full BLE specification.

4.3 Proposed Design

4.3.1 Overview

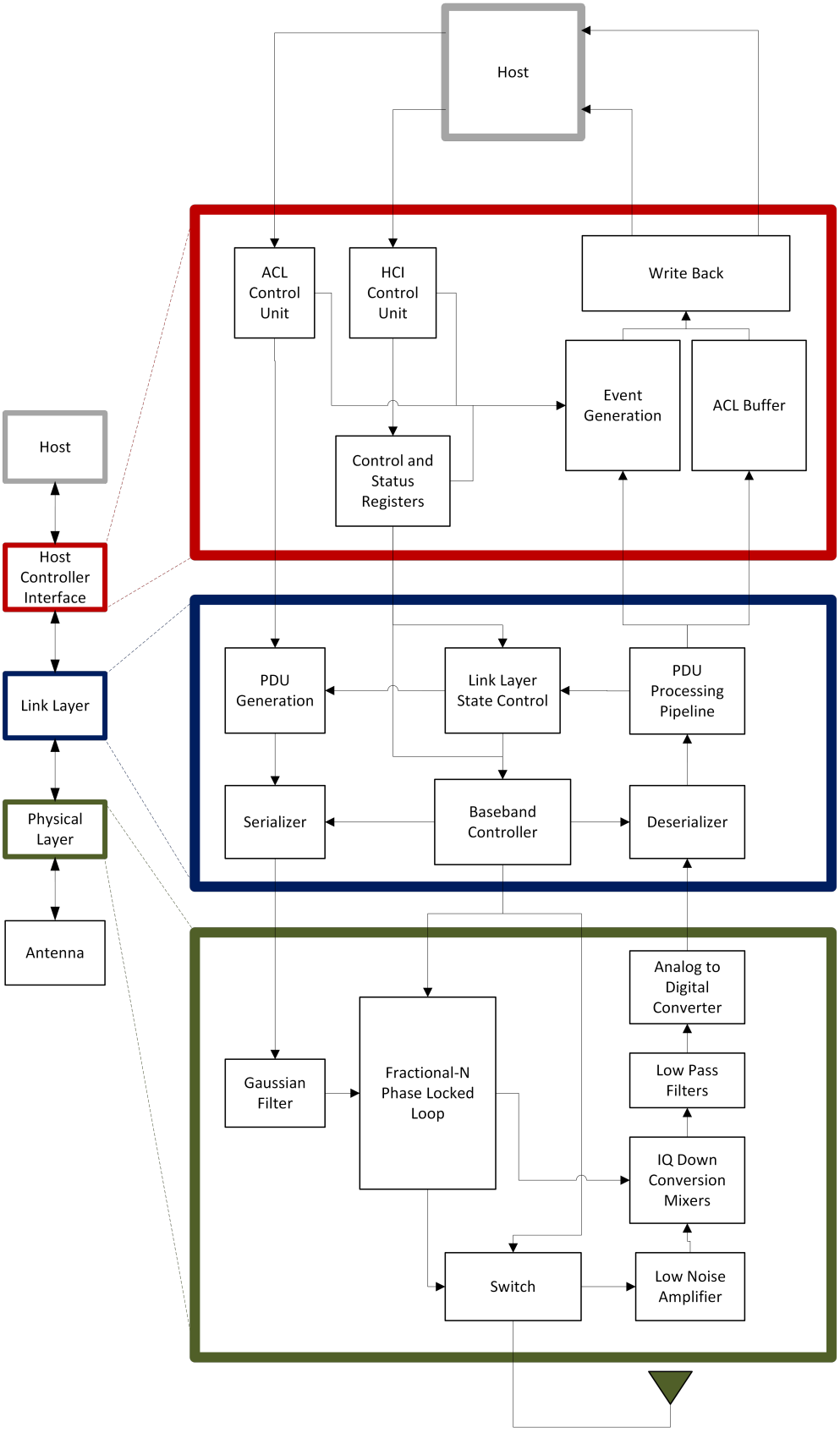


Figure 3: Top Level Design of entire system

The digital side of the design is responsible for communicating between the host, controller, and physical layer. It is also responsible for providing control signals that control the analog radio elements. The digital design is split into two layers: the host controller interface, which is responsible for communication to the host device, and the link layer, which forms and receives packets to be sent over the air.

The physical layer, also known as the PHY, is what contains all analog components. These are responsible for either taking the signal received at the antenna and converting it into a form usable by the digital layer, or transmitting out a signal to the antenna if designated to do so.

4.3.2 Host Controller Interface Module Design

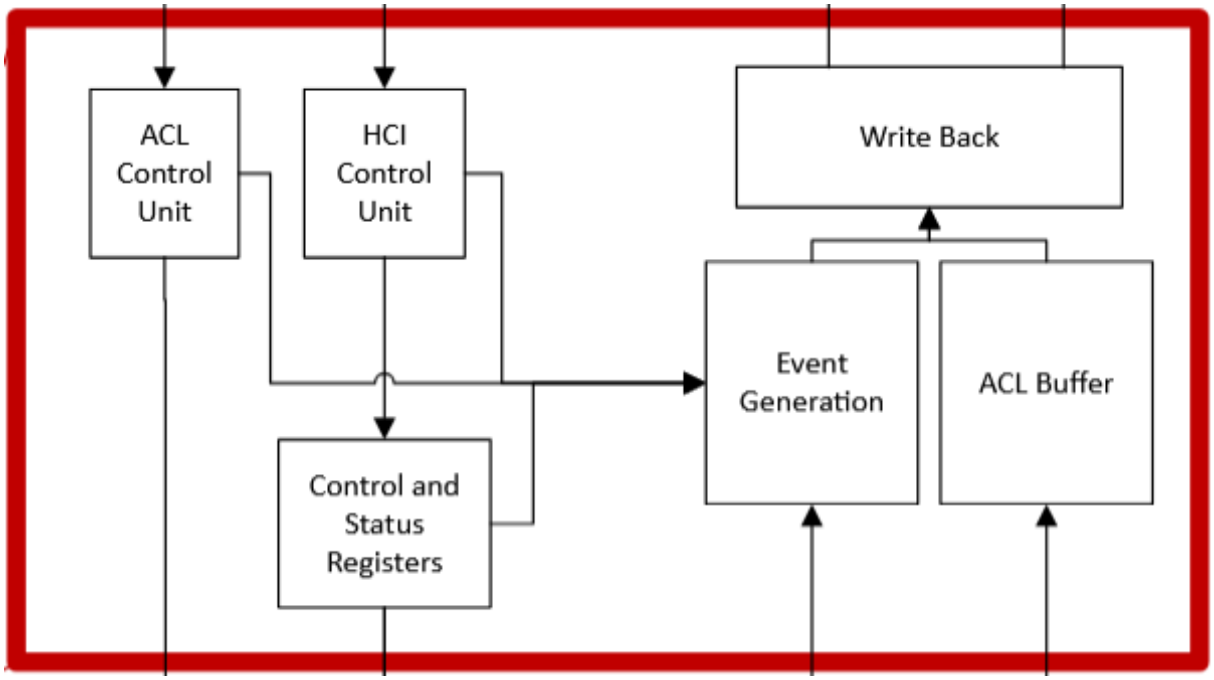


Figure 4: Breakdown of HCI design

- ACL Control Unit

Responsible for passing ACL data into a queue to be formatted for transmission over the air. Will update the control and status registers to reflect if there is data queued for transmission. This module will also parse the header information from the incoming ACL packet and determine if it is valid, if it is invalid it will tell the event generator to raise an event

- HCI Control Unit

Responsible for processing HCI commands and updating status registers accordingly. Will pass relevant information to the event generator in order to generate responses to HCI commands

- Control and Status Registers

Contains state variables that contain both information about the system and operational information that will be used to drive the operation of the device.

- Event Generation

Formats response events that are signaled from other modules in the system. These typically occur due to HCI commands but can arise from events in the link layer as well. These will be passed back into write back and then into the host.

- ACL Buffer

Buffers ACL data directed at the host coming from the link layer.

- Write Back

Has read ports on the wishbone bus as well as a bit to indicate to the host that the data inside of the read out register is currently valid. Will take events from the event generator with priority over ACL data transmission. Previous modules are buffered to allow stalls due to collisions between events and data.

4.3.3 Link Layer Module Design

The link layer generally serves as the layer that formats packets for transmission over the air, processes information received over the air, and determines its routing. It implements several operating modes that describe different protocols and timings for operating the radio. The five implemented states are shown in the diagram below. Transitions between the states are determined by both information in the status and control registers as well as packets received over the air.

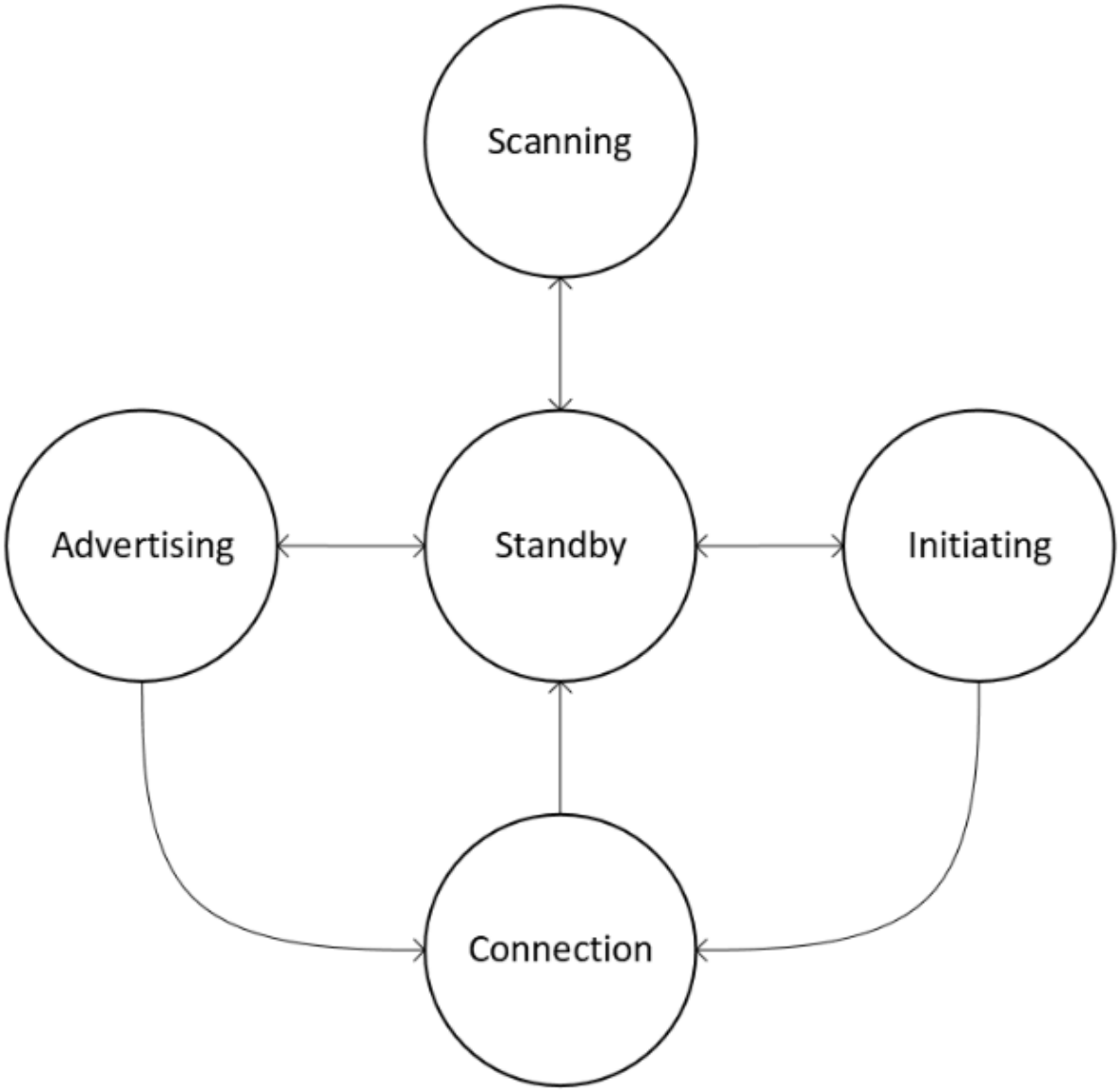


Figure 5: Link Layer State Diagram

- Standby State

In this state, the radio is not functioning, the link layer will neither send or recieve data. This serves as the default state, and any state can transition to this state.

- Advertising State

In this state, the controller will send at periodic advertising packets to inform other devices of its address and specifications. It will also listen for incoming connection requests to allow for a transition to the connection state

- Scanning State

In the scanning state, the radio will be set to receive only, and will listen for packets on the designated advertising channels. This is done to provide a report of nearby devices to the host to allow a user to initiate connections

- Initiating State

In this state, the controller will listen for advertising packets from a specific device and respond to initiate a connection with another device.

- Connection State

The connection state is when two BLE devices are actively exchanging information with each other. In this state the device that entered from the initiating state takes the central role and the device that entered from the advertising state takes the peripheral role. The device in the central role is responsible for dictating timing in the messages.

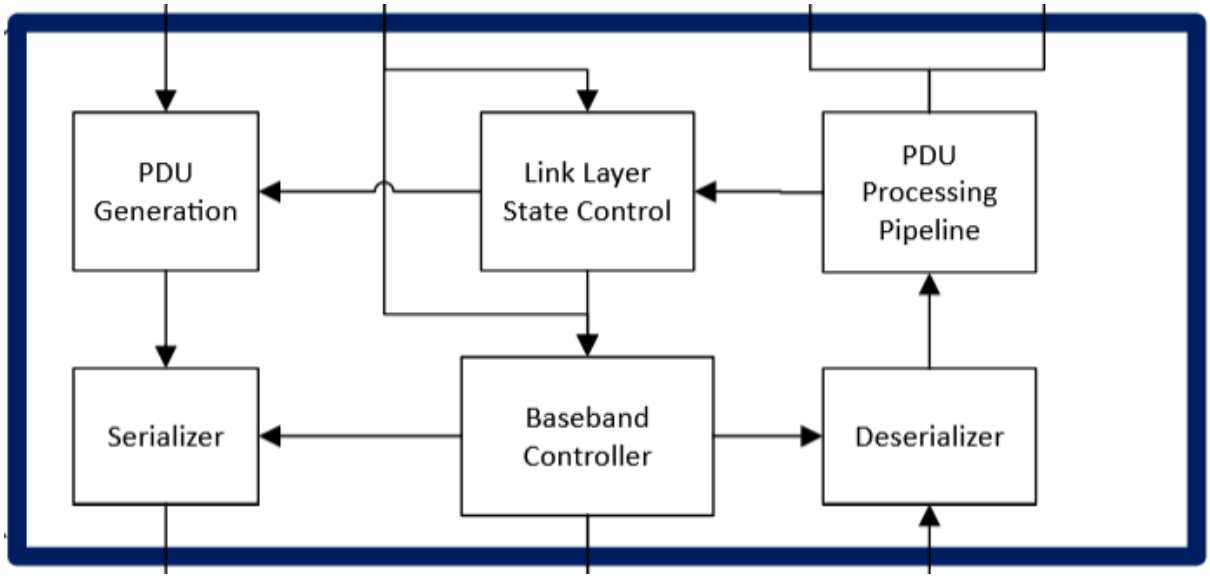


Figure 6: Breakdown of Link Layer Design

- Link Layer State Control

The link layer state control contains an FSM that takes inputs from the status and control register block inside of the hci and from the pdu processing pipeline in order to create outputs that control the baseband controller and PDU generation.

- PDU Generation

This module forms packets that will be sent over the air. When in the advertising mode, it will create standard advertising packets. When it is in the initiating mode it will send standard connection

requests. When in connection mode it will exchange ACL data coming from the ACL control unit, when in this state it only prepends the connection handle of the device.

- Serializer

The serializer takes in full packets into a holding register and outputs them one bit at a time according to the data rate when the device is transmitting. This module will also calculate a CRC and whiten the data. Data whitening is the process of programmatically inserting extra bits to prevent large series of 0's or 1's.

- Baseband Controller

The baseband controller drives all of the control signals into the analog radio controller according to the state provided by the link layer state controller. This includes if the radio is transmitting or receiving and what channel the device is transmitting or receiving on.

- Deserializer

This module is responsible for receiving a bitstream from the radio and formatting it into a packet that can be moved around at the register transfer level. It will first dewhiten the data, then check the CRC to ensure that the data is received correctly.

- PDU Processing Pipeline

The PDU processing pipeline will process data received from the radio and determine it's intended destination. Sometimes this will dictate changes in state like cancelling a connection. Other PDU's may cause changes to status registers in the HCI. If the received packet is ACL data then it will be routed up towards the ACL buffer in the HCI.

4.3.4 Radio Design

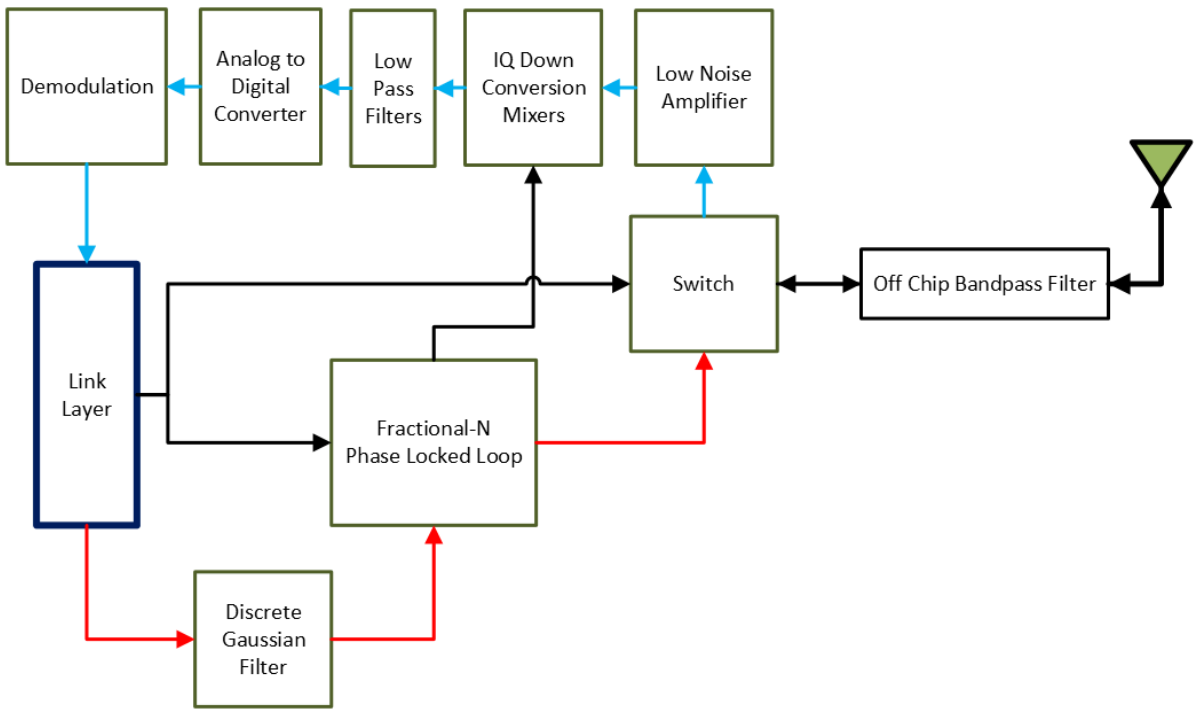


Figure 7: High Level Analog Layer Design

The BLE protocol is half-duplex, meaning that we can either be transmitting or receiving, but not both at the same time. Our radio architecture reflects this with an RF Switch which lets us either

take in a signal form the antenna and guide it through the receive path, or transmit a signal coming from our frequency synthesizer, the PLL. The red arrows traverse the path of transmission while the light blue outlines the path for receiving.

4.3.4.1 Transmission of Packets

BLE has 40 channels over the BLE frequency spectrum where the center of each channel is spaced 2 MHz apart. Symbols, zero and one, are modulated as frequency deviations from the center of the channel. A zero bit is modulated as negative frequency deviations from the center while a one bit is modulated as positive frequency deviations from the center. More specifically, BLE uses Gaussian Frequency Shift Keying (GFSK) where the deviations are ran through a gaussian pulse shaping filter as opposed to sharp instant frequency changes.

The Link Layer provides the radio with a stream of symbols for us to modulate. The deviations are ran through the gaussian filter and the PLL is given a value corresponding to the desired frequency. The PLL should output to the switch and then through the antenna. This architecture lacks a power amplifier in an attempt to prioritize design and implementation of other more essential components.

4.3.4.2 Receiving of Packets

When a packet over the air catches onto the antenna, and we are in a receiving type mode, the signal goes through the receive side of the radio through the switch. We use a low noise amplifier (LNA) to amplify the signal as they tend to become weak as they propagate over the air. It is important that our amplifier has a low noise figure as the noise introduced in the earlier stage are more impactful on noise. We use two mixers for IQ down conversion. One mixer takes in an in-phase signal from the PLL (which should be at the relevant channel's center) while the other mixer takes in the quadrature (90 degrees shifted) signal from the PLL. They both mix with the post low noise amplified signal that is being received. IQ down conversion allows us to track wether the incoming signal is positive or negative (in terms of frequency) in respect to the locally generated signal of our PLL. Once down conversion has taken place, we use low pass filters to try to filter out any out of channel communications. Following this, the signal is to then be digitized, demodulated into symbols, and then passed to the Link Layer.

4.3.5 Functionality

The system operates as a Bluetooth Low Energy controller that allows a host device to communicate wirelessly through a fully integrated digital and analog architecture. From a user perspective, the device behaves as a standard BLE controller capable of advertising, scanning, initiating, and maintaining connections with other BLE-enabled devices.

A typical interaction begins with the host writing commands into the shared memory buffer, such as initiating a scan or starting advertising. These commands are processed by the HCI module, which updates internal control registers and triggers the appropriate link layer state transitions. For example, a scan command will move the system into the scanning state, where it listens for nearby BLE devices.

Once in an active state, the link layer manages all timing-critical operations. In advertising mode, it periodically generates and transmits packets containing device information. In scanning mode, it listens for incoming packets and reports discovered devices back to the host. If a connection is initiated, the system transitions into the connection state, where it exchanges data packets with another device following BLE timing constraints.

Data transmission follows a structured pipeline. Outgoing data from the host is formatted into protocol data units, serialized, and passed to the analog front-end for modulation and transmission.

Incoming signals from the antenna are demodulated, converted into digital data, and processed by the link layer before being forwarded to the host.

The system continuously cycles through these operations based on host commands and wireless events, maintaining synchronization with other BLE devices while ensuring reliable communication. This interaction between host control, digital processing, and analog transmission enables the device to function as a complete BLE controller in a real-world environment.

4.3.5.1 Digital Functionality

The digital subsystem is responsible for coordinating communication between the host, link layer, and physical layer while managing control flow and data movement throughout the system. At a high level, the digital design operates as a bridge between software-visible host interactions and time-sensitive radio operations.

Communication with the host occurs through a shared SRAM buffer, which acts as the primary interface for command and data exchange. The host writes HCI commands and ACL data into predefined memory regions and signals the controller when new data is available. The fetch unit monitors these regions, reads incoming data, and determines whether it should be routed through the HCI command datapath or the ACL datapath.

Once received, HCI commands are processed and reflected into the control and status registers, which act as the central state repository for the system. These registers drive behavior in both the HCI and link layer, allowing high-level host commands to influence low-level radio operation. Events generated as a result of command execution or link layer activity are formatted and written back to shared memory through the write-back unit, ensuring the host receives timely updates.

The link layer operates as a state-driven system that controls packet generation, transmission, and reception. Based on the current state and register values, it determines when to transmit, receive, or remain idle. Data intended for transmission is passed through the PDU generation module, serialized, and sent to the physical layer. Incoming data follows the reverse path, where it is deserialized, validated, and routed either back to the host or used to update internal state.

Interaction between modules is performed through buffered connections and defined datapaths, minimizing timing hazards and allowing independent operation of subsystems. Control signals from the link layer also drive the baseband controller, which configures the analog front-end for transmit or receive operation.

Overall, the digital functionality is centered around structured data flow, state-driven control, and tight coordination between host-visible behavior and real-time radio communication.

4.3.5.2 Analog Functionality

The analog subsystem handles all of the RF signals, whether that be transmit or receive. This relies heavily on high-frequency devices and high operating speeds.

The device should contain 40 channels, each spanning 2 MHz, within the realm of 2.402 - 2.480 GHz. There should be no more than ±150kHz deviation from the center frequency for a given signal, and should be able to switch channels and settle at a frequency within 150µs. It should be able to transmit at between −20dBm to 20dBm, ideally functioning effectively within around a 3 meter radius.

A receive signal should be able to be received, amplified, synchronized through the PLL, filtered, digitized with an ADC, and digitally demodulated. A transmit signal should be able to go from the Link Layer, get shifted via GFSK, modulated through a sigma-delta modulator, synchronized with the PLL, amplified, and transmitted.

4.4 Areas of Concern and Development

4.4.0.1 Digital Concerns

The digital aspect of the project is very feasible and well grounded, but there are many concerns that will need to be mitigated. Given the scope defined with the specific BLE requirements that will need to be met, the current design would fulfil these requirements. The primary concern is the SoC is able to meet the features, which requires it to meet BLE specification. The immediate development plans is to create and verify the logic for each sub component before continuing further into development.

The digital side is not constrained to as limited open-source knowledge as the analog side, which gives the digital side much more flexibility. Even with concerns with the analog side, the digital design should hold strong regardless of the outcome from analog given options to move some analog aspects off-chip if absolutely necessary. Even with an external RF interface, the digital side would still meet specification while delivering a open-source SoC that is BLE compliant with an external chip.

4.4.0.2 Analog Concerns

Many elements of the chip (inductors, high frequency devices) have very limited open testing knowledge for RF design on the given process. While the chip itself may pass all simulation and post-layout checks, it remains to be seen whether this reflects real-world performance, particularly when it comes to devices in the GHz range.

Our chip is also designed around a single antenna. As such, performance with other antennas may vary, potentially resulting in the device not being truly “Bluetooth compliant”, even while meeting all other criteria. BLE also has support for multiple antennas, and it remains to be seen whether that will be feasible to implement.

The open-source software tools available are not perfectly optimized, and as such, certain revisions may take longer than expected. There may end up being many cases where revisions cannot be completed until parasitic calculations are complete, to the point where it may push the timeline beyond what is allotted.

4.5 Technology Considerations

4.5.1 Digital Technologies

The primary technologies for the digital side can be broken up into the simulation environment phase and the digital module phase.

Simulation environment technologies:

- Python:

Python was chosen as the language for the simulation environment because development is fast, many members already know it, and there is good library support for BLE and simulation. Initially C was used for the first pass at simulation however the advantages of C do not really apply to our project. Doing memory management is not helpful for this project and python is fast enough to run large simulations. C added complexity slowing development down without much gain.

- Bumble:

Bumble was chosen as the host software for the simulation because it is in python which makes interaction easy. It is also reconfigurable at run time allowing for different host patterns to be tested on the design without needing to recompile a host like Zephyr would require.

- Babblesim:

Babblesim is used to simulate the physical radio layer between devices. This provides the ability to inject noise in the simulation allowing our model to be tested in nonideal conditions. It also tracks transmission timings which allows for the device to test timeout conditions and other time based functions.

- Zephyr:

Zephyr serves two roles for this project. The first is as a simulated device for our design to talk to two. It is useful for this task because it has full driver stack and can connect to Babblesim. The alternative plan was to write a small bridge to connect a Bumble host and controller to Babblesim. Since both methods are compliant to the BLE specification Zephyr was chosen for easier operability with tools that we are already using. The other purpose of Zephyr is to be the operating system for our host. This was chosen because the Zephyr is already known to work with the RISC-V core. This means there will need to be only a slight driver modification to allow Zephyr to write to the correct memory address.

Digital module technologies:

- SystemVerilog:

SystemVerilog was chosen as the HDL because the open source hardening tool flow can use Verilog and Systemverilog and we may use some of the features inside of Systemverilog.

- Cocotb:

Cocotb is a python library that allows for python scripts to drive and read signals from hardware simulators. This will enable the capability to plug the hardware design directly inside of the virtual testing environment. The alternative is to create test scripts in TCL from the io of the testbench. This will make verification easier since making a change to the virtual simulation environment will cause be immediately reflected in inputs to the gold model and hardware. This library is recommended by chipIgnite for verification.

4.5.2 Analog Technologies

For the analog implementations the use of a few key technologies is crucial. As a base, this project will use the open source SkyWater 130nm process node (PDK) to implement the components. These implementations will begin as schematics in xschem. Testing will occur with NGspice, and layout will be completed using Magic. This will allow for fully open source analog implementation. However, the PDK is not explicitly designed for the purpose of RF design and as such will struggle to implement components that operate well at the required frequencies.

4.6 Design Analysis

4.6.1 Digital Analysis

The simulation environment was built initially with the old design with support for advertising before deciding to go with the alternative design. Currently the simulation environment is in active development and appears to be going much smoother based on unit tests on individual modules on the DUT as well as the conductor and device.

For digital modules, CRC and Whitening have been built and tested using Verilog. The test bench has been hardened and results in a serializer that functions properly. While most modules will need to wait until the simulation environment is completed CRC and whitening is well defined in the BLE

specification with very little design flexibility. This gives proof that the toolflow laid out for us is working properly and future modules should be quick to implement with the proof-of-concept from the simulation and test benches.

4.6.2 Analog Analysis

Implementation of components in xschem and tested using transient analysis, frequency analysis, voltage sweeps, etc such that the designed components are able to operate at necessary spec, in this case being BLE compliant. Design rule checks and layout versus schematic must be passed in order to confirm component functionality.

5 Testing

The goal of testing is to provide a development environment that ensures that each component works as it is built. To accomplish this a test plan has been developed that incorporates simulations and physical tests during the development process to ensure that our design meets the requirements.

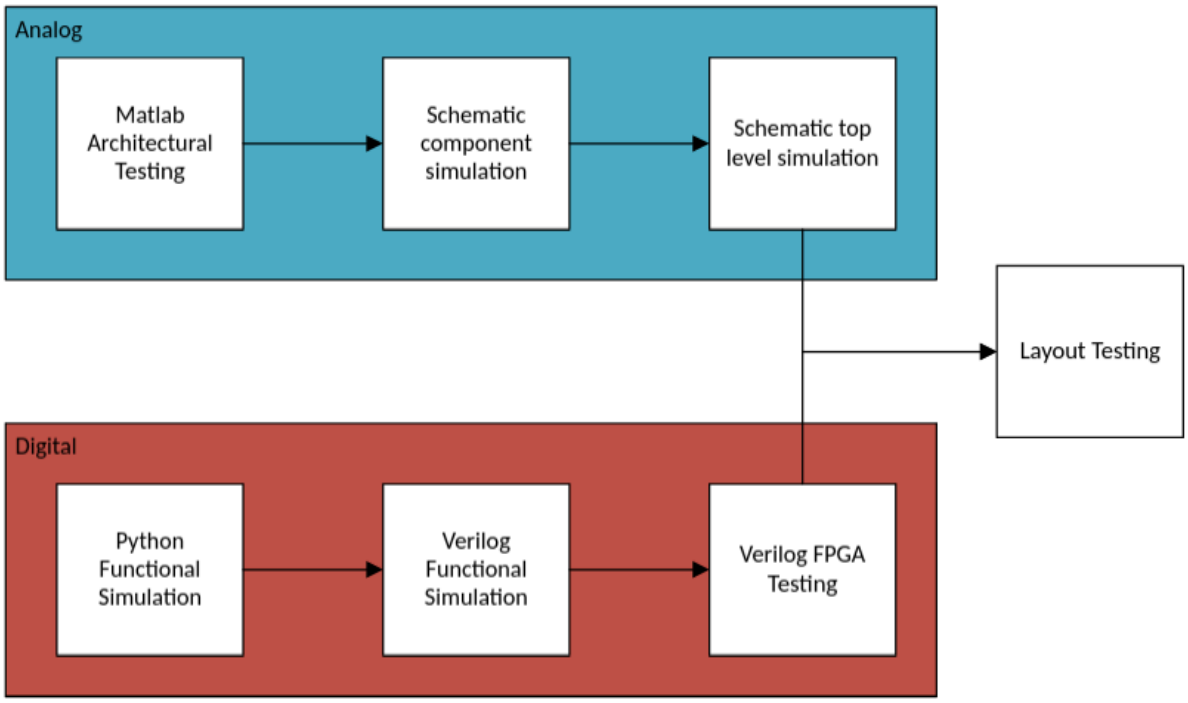


Figure 8: Test Plan Diagram

5.1 Digital Plan

The digital testing plan is to first develop a golden model of the BLE controller in python. This will be used to verify the function of later steps. Once this is complete the device will be implemented in verilog and plugged into the virtual sim environment. Once this is complete the digital end will be verified on the FPGA. Once FPGA testing is complete the digital side will be integrated with the analog side and undergo further testing.

5.1.1 Software model in simulation environment

The first step in the digital simulation is to create the simulation environment. This consists of three tools:

- Bumble: an open source BLE host emulator maintained by google. It will be used to emulate a host in the simulation environment. This tool is verified by its maintainers by using it's outputs to drive bluetooth devices.
- Babblesim: an open source physical layer simulation that is designed to simulate radios.
- Zephyr Project: an open source RTOS project. It contains an open source bluetooth driver that can be wired into babble sim.

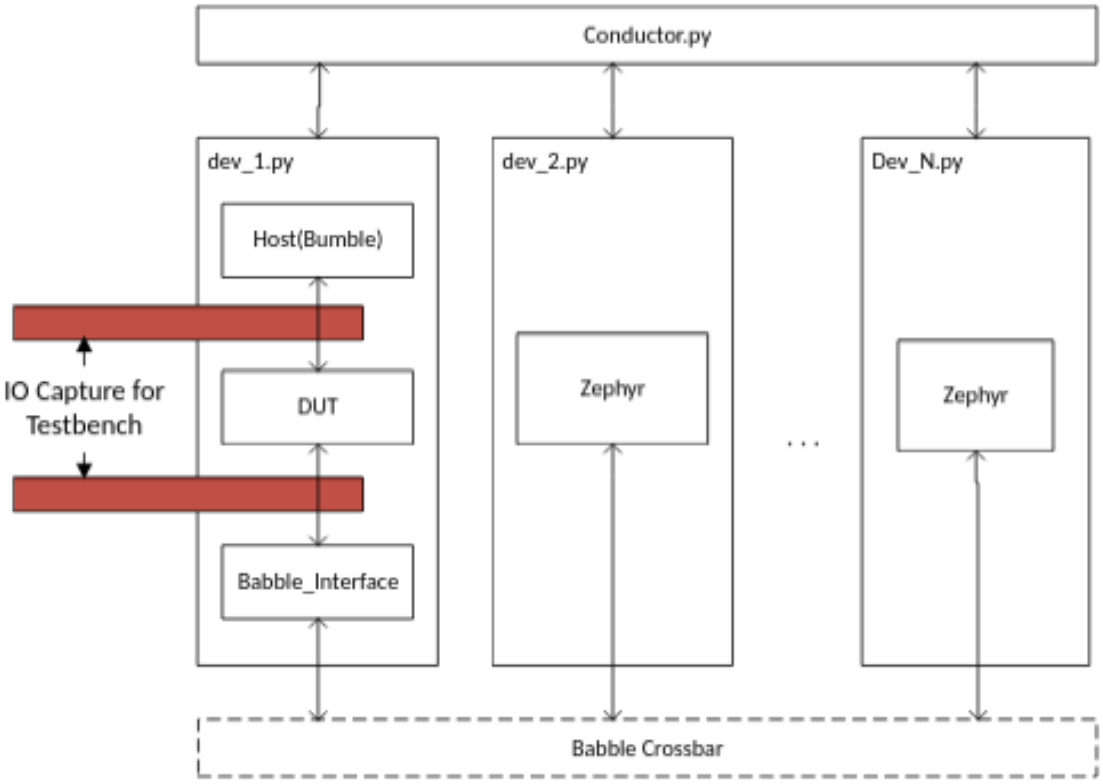


Figure 9: Simulation Environment Architecture

This environment will have a host attached to the DUT in the form of Bumble, it will also be able to communicate with Zephyr over Babblesim. This provides proven emulators on each side of our device to ensure our device is complaint with BLE specifications. This testing will show that our golden model meets the requirements of BLE 4.0 adverting, scanning, and connection. A top level python file called the conductor will start all of the simulated devices as well as run time steps for each device.

5.1.2 Verilog design in simulation environment

After the golden model is developed and proven inside of the simulation environment, the Verilog implementation of the controller will be plugged into the simulation environment to verify it's function using a python library called cocotb. This provides python controlled connections into devices being simulated.

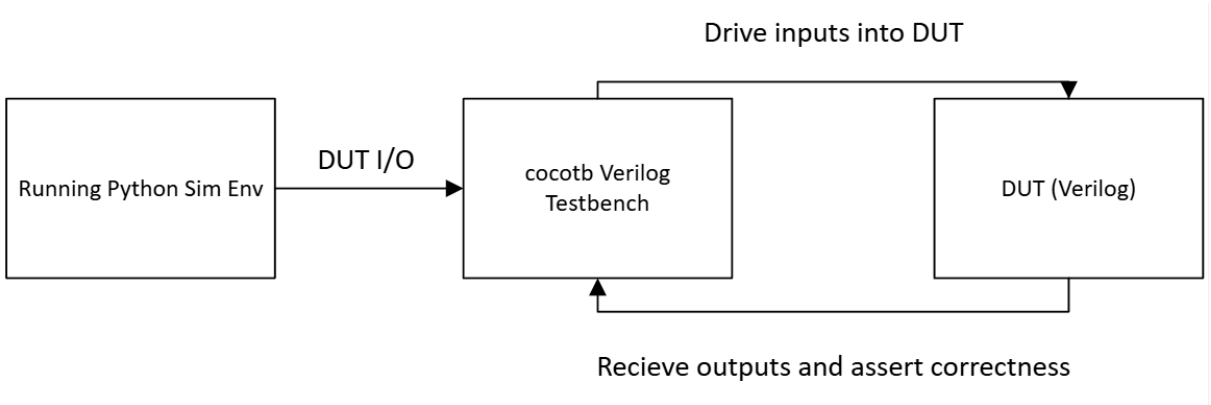


Figure 10: Simulation Environment Architecture

The inputs will be cloned from the those provided to the golden model and provided to the Verilog implementation. The outputs of the Verilog implementation can then be checked against the outputs from the python golden model to determine correctness.

5.1.3 FPGA Testing

FPGAs allow us to test digital hardware before fabrication. The host side and Link Layer can be placed on the FPGA. While we can't place our analog radio onto the FPGA (given that it's analog), we can utilize the high speed serial transceivers that are commonly fitted on FPGAs to implement a radio to serve in testing our Link Layer. More specifically, we have access to Zynq Ultrascale+ FPGAs which have GTH transceivers which contain fractional-n PLLs which we can use to transmit packets. It being a fractional-n PLL is particularly convenient considering it provides us a similar interface of what we expect out of our analog side.

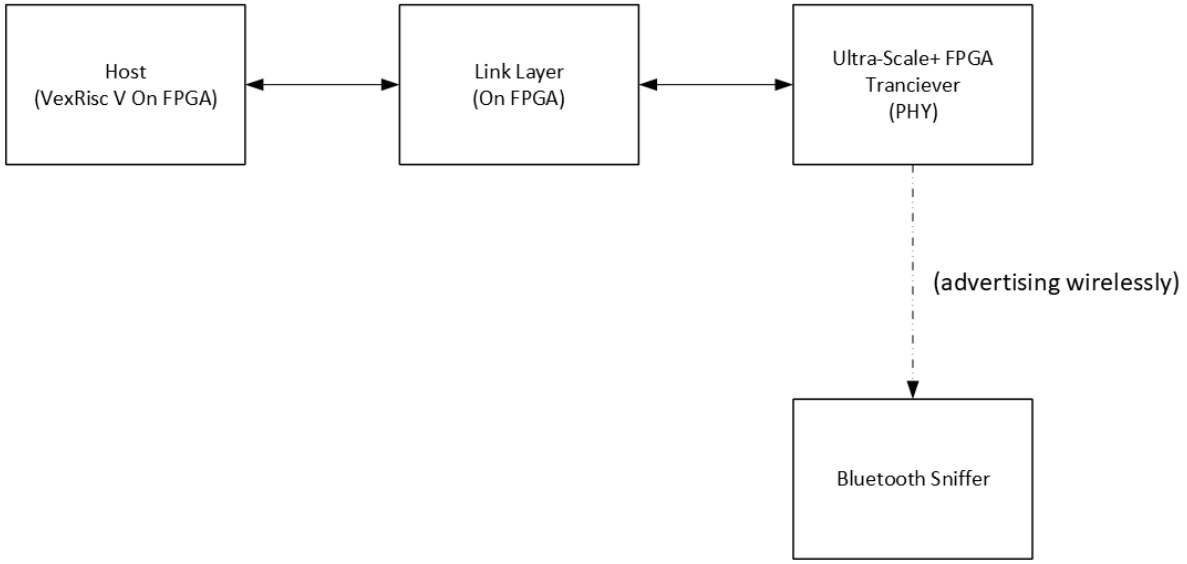


Figure 11: FPGA testing diagram

In terms of being able to receive BLE packets on the FPGA, a PCB can be designed to receive, amplify, IQ down-mix, and digitize as an extension the FPGA for packet receiving. VGA refers to variable gain amplifiers with the AGC signal being the automatic gain control to the VGAs:

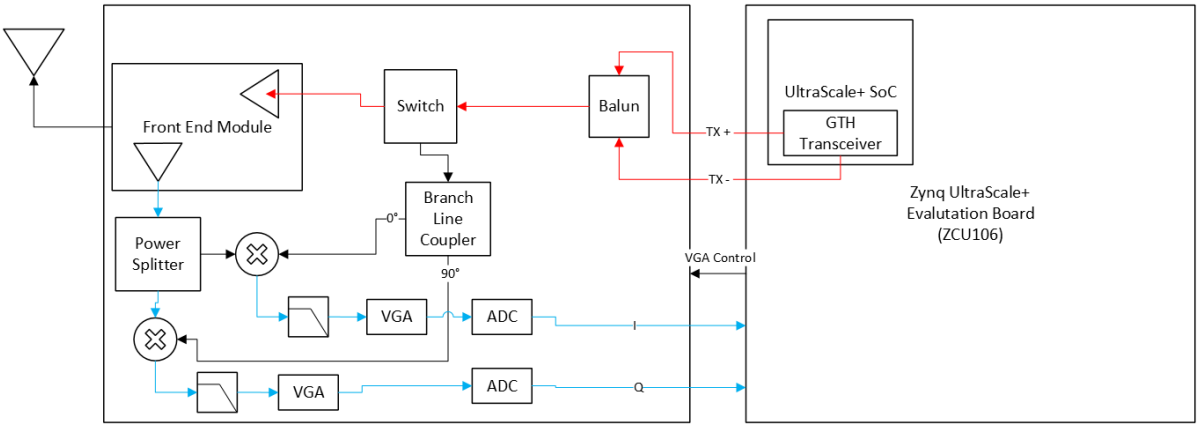


Figure 12: PCB extension to FPGA evaluation board for enabling receiving

5.2 Analog Plan

The analog testing plan is to develop a model in Matlab/Simulink and prove funcitonality. From there the components will be designed in xschem and put to layout in magic before beginning integration with digital components and final tests.

5.2.1 Architectural Simulation

Using Simulink's various toolboxes including the RF/Mixed signal toolbox, system level testing of the physical layer can be achieved using ideal blocks and functionality. With ideal functionality as a proof of concept, noise analysis and testing of non-idealities can be done in Simulink aswell. This will provide confidence in the next phase of design when implementing components as schematics.

5.2.2 Component Schematic Simulation

Using the analog VLSI toolflow, specifically in regards to Xschem, schematics can be developed for the components in the analog portion. These components can then be tested to gauge performance. Thorough analysis of all components is essential to ensure functionality at the high level and not just low level.

5.2.3 Schematic Top Level Simulation

With all components working as expected the next step is to test multiple components together. Individual components may meet spec on their own, but as a high level system it may not work. Once functionality can be proven in a high level system, the components can be considered ready for layout.

5.3 Layout Tests

After the digital and analog sides have been integrated into a single design there are final verification tests that will be run. The Design Rule Checker will be run to ensure that the layed out design meets all rules specified for the Sky130A process. The ensures that the design is able to fabricated by ChipFoundry. Next, an LVS check is run to determine that the layout matches the integrated schematic design. Once both of those pass, parasitic simulations can be run to ensure that the added capacitance and resistance of an actual layout does not seriously affect the operation of the device. If not, the layout will likely need to be rearranged in order to minimize said parasitics.

5.4 Results

5.4.1 Digital Results

Simulation Environment: The conductor-based simulation environment is partially functional. The Python DUT performs advertising and scanning, exchanging packets with a sister Zephyr controller through BabbleSim. This demonstrates a functional RX and TX pipeline through our DUT. Standby, Initializing, and Connection states remain to be implemented into our DUT. BabbleSim does not require to specify RX or TX state which is handled by our baseband switch, so this will need to be manually specified and tested outside our Bumble and BabbleSim environment.

Serializer Verification: In parallel to the simulation environment the serializer, which is well-defined in BLE specification, has been designed, tested, and verified. The serializer includes CRC and data whitening, which are defined in BLE specification. The device has been created in System Verilog and has been tested with System Verilog testbenches based on real BLE packet exchanges.

FPGA Based Radio: While the Ultrascale+ GTH transceiver based radio implementation is still a work in progress, FSK modulation has been achieved. A discrete gaussian pulse shaping filter has also been designed and implemented as a lookup table, though has not been used yet for testing GFSK modulation.

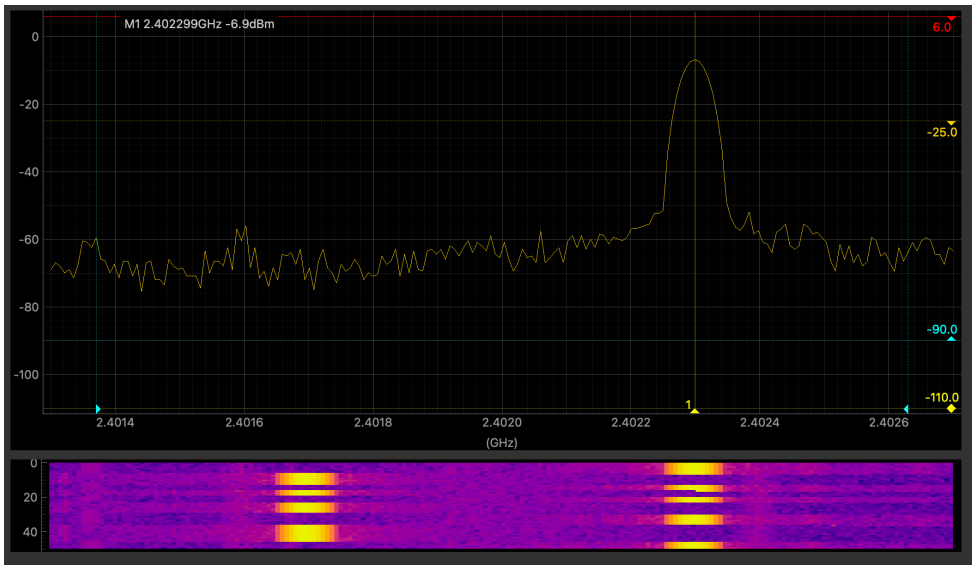


Figure 13: FPGA FSK modulation on channel 37 with slowed down symbol times (so to show on spectrum analyzer)

Known Issues: There are currently two known inconsistencies between our planned design and actual DUT implementation in the simulation environment.

- The HCI datapath currently handles ACL traffic. The planned controller has a seperate module for ACL traffic, so the DUT HCI datapath will need to be broken up to reflect our planned hardware design.
- The device.py, which controls the individual DUT device, currently handles the device state, which needs to be determined directly in the link layer state controller.

5.4.2 Analog Results

Simulink Environment: The Simulink environment is operational for the main components of the PLL system and successfully demonstrates functionality within the desired operating parameters. This includes operation after injection of various thermal noise sources. Several other components, including some mixers and filters, have also been simulated.

RF Switch: A switch architecture has been selected and had a schematic successfully simulated in xschem. Based on preliminary transient simulations, the switch may have to be moved off chip in order to minimize latency.

6 Implementation

6.0.1 Digital Implementation

The current digital implementation follows a path from simulation environment to hardware module verification, but there has been some parallel work done. Currently this implementation can be broken up into two workstreams: Simulation Environment, and RTL Modules.

Simulation Environment: The Python conductor and device implementation is in active development. Bumble provides the host, BabbleSim provides the wireless channel, and a Zephyr controller serves as the sister device to verify our implementation. The DUT currently supports advertising and scanning, with packets exchanged successfully through BabbleSim to a sister Zephyr device. Remaining work includes the standby, initiating, and connection states, along with restructuring the DUT using System Verilog to align with the planned RTL architecture.

RTL Modules: The serializer, which contains CRC and whitening, has been written in System Verilog and tested with a testbench based on real BLE communication packets. Additional modules will be written as their behavior is fully fleshed out and tested in our simulation environment. Our simulation environment combined with cocotb will give us accurate System Verilog testbenches for future RTL modules.

6.0.2 Analog Implementation

Analog implementation follows a path from system level design into component architecture determination. From there the components can be designed to meet certain constraints placed on them in order to give the results that are needed for the project to work as intended. These constraints are determined by the standards to which BLE 4.0 requires for functionality

System Level Model: A system level model is being developed using Matlab/Simulink. It can replicate RX/TX switching logic and circuit behavior. In receive mode, a correct signal output is seen for the time it was in receive mode at the end of the RX path. While in transmit mode, a correct signal output is seen at the antenna for the given frequency controls. ADC and DAC logic is still to be implemented.

Schematic Simulation: In Xschem, a RF switch has been modeled and simulated, the performance is still a work in progress. Other components such as the PLL or Gaussian Filter will be started in 4920.

7 Professional Responsibility

7.1 Areas of Professional Responsibility

Area of Responsibility	Definition	Code of Ethics Item[11]	Team Interaction
Honesty & Integrity	Being upfront and honest about where the project is at, including things that went wrong or are still progress	IEEE 3: “To be honest and realistic in stating claims or estimates based on available data”	Every week, we prepare a status update and presentation slides for our advisor meeting. If a task is incomplete or a test fails, we clearly report it in the slides. Being honest about our progress has helped us receive more useful and accurate feedback, rather than trying to present the work as better than it actually is.
Health, Safety, Wellbeing	Keeping the safety of users and anyone affected by our design as a top priority throughout the project.	IEEE 1: “To hold paramount the safety, health, and welfare of the public”	Our BLE strictly follows the Bluetooth specification, including correct channel usage and timing. If our implementation is incorrect, it could interfere with other wireless devices. So we treat following the specification as a safety requirement, not just a technical one.
Property Ownership	Giving proper credit to external sources and being clear about what work is ours and what came from somewhere else.	IEEE 9: “To avoid injuring others, their property, reputation, or employment by false or malicious actions”	We use resources like the Zephyr BLE code and Bluetooth specification for learning. However, all the code we submit is written by our team. If any idea comes from an external source, we mention it in our documentation.
Equitable Collaboration	Making sure the work is actually split fairly and nobody ends up doing way more than everyone else.	IEEE 7: “To treat all persons fairly and with respect”	We use a shared Excel sheet to track each team member’s weekly work hours. Tasks are assigned based on strengths, like digital or analog, so everyone works in areas they are comfortable with. While the system is not perfect, it helps us track contributions and keep the workload more balanced.

Verified Completion	Ensuring a component is fully validated through testing before marking it as complete.	IEEE 6: “To maintain and improve our own technical competence”	We define clear test criteria that must be met before any task is marked complete. This includes passing unit tests for CRC and whitening, maintaining a stable connection for at least 10 events, and verifying correct packets using Wireshark. A task is only considered done if all tests pass, regardless of code quality.

Table 6: Professional Responsibility & Code of Ethics

7.1.1 Area of Strength

An area where our team has performed well is honesty and integrity. From the beginning, we agreed that our weekly updates and meeting slides would accurately reflect the actual project status, even when tasks are incomplete or tests are failing. This transparency has made our discussions with the professor more productive, as we receive feedback based on real issues rather than an overly polished view. It has also helped us avoid overestimating progress and prevented us from building on unverified components.

7.1.2 Area for Improvement

One area our team can improve is thinking more about the wider social and environmental impact of our design. Right now, we focus mainly on making the system work correctly, but we do not fully address how it might be misused or its long-term impact. Moving forward, we should include clearer usage guidelines and explore simple safety measures in the design to make the system more responsible and aligned with ethical engineering practices.

7.2 Project-Specific Professional Responsibility Areas

	Beneficence	**Nonmaleficence**	**Respect for Autonomy**	**Justice**
Public health, safety, welfare	Open-source RTL allows security researchers to audit the BLE implementation for vulnerabilities, which is especially important for health-adjacent BLE devices such as glucose	An incorrect BLE implementation could cause unintended RF interference in the 2.4 GHz band, potentially disrupting other medical or safety-critical wireless devices operating nearby.	Because the design is fully open, medical device developers and security researchers can inspect and modify the implementation to meet their own safety requirements rather than	Provides equal access to BLE hardware-level security knowledge for researchers and institutions that lack the resources to audit closed-source alternatives.

	monitors and heart rate sensors.		relying on opaque, proprietary hardware.	
Global, cultural, social	Democratizes access to BLE ASIC design by providing a fully documented, fabrication-ready reference design to students, hobbyists, and academics worldwide who would otherwise have no access to silicon-level BLE resources.	Open RF designs, if improperly replicated without adequate testing, could result in non-compliant emissions. Clear documentation and a defined test plan mitigate this risk for downstream users.	Any individual or group is free to use, study, or extend the design without licensing restrictions, preserving their freedom to make independent design decisions at every level.	Lowers the barrier for under-resourced institutions, particularly those outside major research universities, to participate meaningfully in ASIC and wireless design education.
Environmental	The modular architecture allows future ChipForge teams and others to reuse verified components rather than redesigning and re-fabricating from scratch, reducing unnecessary tape-out runs and associated material consumption.	Semiconductor fabrication involves chemical processes and energy consumption with real environmental impact. A failed or insufficiently verified design represents environmental waste that cannot be recovered.	Future teams are free to choose which modules to reuse, redesign, or discard based on their own design goals and environmental priorities, rather than being locked into any particular architecture.	By sharing the design openly, the cumulative environmental cost of BLE ASIC development is spread across many future users who would otherwise each incur the full cost of designing a similar system independently.
Economic	Provides a free, fully documented BLE ASIC design that reduces the cost of entry for students, small teams, and academic institutions into BLE hardware development,	There is a minor potential for negative economic impact on commercial BLE chip vendors; however, the scale of this project is academic and the	Users are free to use the design for academic, personal, or commercial purposes without licensing fees, preserving their economic independence	Redistributes access to BLE silicon resources away from a small number of large companies and toward a broader community of students, researchers, and

	which is otherwise dominated by expensive proprietary solutions.	effect on industry is negligible.	from proprietary vendors.	hobbyists regardless of institutional funding.

Table 7: Project-Specific Professional Responsibility Areas

Important context–principle pair: Global/Cultural/Social + Beneficence. The core benefit this project works toward is making BLE ASIC design accessible to students and hobbyists who currently have no open-source silicon-level reference to learn from. We will ensure this by maintaining thorough documentation throughout the design process, releasing all RTL, schematics, and layout files under an open-source license, and structuring the design modularly with clear comments so future ChipForge teams can build on it directly.

Lacking context–principle pair: Environmental + Nonmaleficence. Every tape-out consumes real fabrication resources, and a non-zero risk exists that the design requires re-spins due to issues discovered post-fabrication. This cannot be entirely avoided. However, it is partially overcome by the fact that a successful open-source tape-out prevents many future teams from independently incurring the same costs. To further reduce this risk, we are investing heavily in simulation and post-layout verification before committing to tape-out, minimizing the likelihood of a wasted fabrication run.

7.3 Most Applicable Professional Responsibility Area

7.3.1 Team Virtues

Honesty: The team has demonstrated honesty by openly reporting the state of the simulation environment, including its full redesign mid-semester, in weekly advisor updates. Rather than presenting incomplete work as finished, we clearly documented what was working, what was not, and what the known issues were. The pivot from a C-based simulation approach to the Python conductor model is a direct example: we acknowledged the original approach was not working and changed course rather than continuing down an unproductive path.

Responsibility: Each team member has been responsible for their assigned deliverables throughout the semester. Weekly reports, design document check-ins, and presentation slides have been completed on time across the team. In cases where tasks took longer than expected, members communicated delays early rather than missing deadlines silently. The shared time-tracking spreadsheet reinforces individual accountability and makes contribution visible to the whole team.

Diligence: The analog team has demonstrated diligence by systematically working through the challenges of RF design on the Sky130 process, which offers limited open-source precedent for 2.4 GHz operation. Rather than abandoning difficult components, the team has persisted through simulation and schematic work to identify workable architectures. On the digital side, diligence has shown through parallel development of the simulation environment and RTL modules, ensuring steady progress on multiple fronts simultaneously.

Humility: The project operates at the edge of what is feasible on the Sky130 process at 2.4 GHz, and the team has been willing to acknowledge that certain analog components may not perform post-fabrication as they do in simulation. Rather than overstating confidence in the design, analog

concerns and open risks have been transparently documented in this report. This humility prevents overcommitment to an approach that may require revision once tape-out results are returned.

Transparency: The project is fully open-source by both requirement and design philosophy. All RTL, schematics, documentation, and toolflow configurations are maintained in a shared repository accessible to future teams and ChipForge members. This transparency ensures that the work done this semester remains useful beyond the immediate project, and that future teams can clearly understand what was built, what was tested, and what work remains.

7.3.2 Individual Responses

Parker:

- **Demonstrated - Diligence:** Diligence is important because there will be a lot routine work required to hit deadlines. Staying ahead of the work so that there is time to revise it before the final deadlines. This semester I was consistently done with work with days to revise it. This also leaves time to help other members with work before things get out of hand. If we are consistently working to the deadline there will be no chance for revision which will cause errors.
- **Needs improvement - Humility:** Humility is important to me because it was will prevent me from making choices thinking that I am informed on the situation when I actually don't know. It also helps to be humble because you can incorporate feedback on your design without having ego get in the way of it. I found myself giving suggestions with far too little information for what I was suggesting. I need to be more clear about how much I know about the topic. Largely this occurred when discussing analog issues. I will let the analog team be more independent in their choices in the second semester.

Tyler:

- **Demonstrated - Responsibility:** Responsibility matters to me because senior design, but our project specifically, only works if each person can be relied on to deliver what they have been assigned. This project is too large and technical for anyone to fill someone else's role entirely if they fail to deliver. I have been on top of my deliverables, ensuring that reports, design document check ins, and slides are completed in a timely manner.
- **Needs improvement - Diligence:** Diligence matters to me because the digital path to tapeout is long and the work compounds. A week of poor progress causes technical debt to compound quickly resulting in a failed tapeout and failed project. Early on in the semester I was burnt out and my progress was below what I would normally be capable and expect, which led us to redesign the simulation environment later than if I was making ample progress. Going forward, I have recovered from the slump and will pick up more work from other members to ensure a timely tapeout.

Aiden:

- **Demonstrated - Thoroughness** Thoroughness is important to me because in a project of this complexity, gaps in understanding at one stage create compounding problems later. A module that is not fully understood before it is implemented will cost far more time to debug than it would have taken to reason through correctly from the start. In senior design, I have demonstrated this by working to understand the BLE specification and system architecture before contributing to interface definitions and HCI design, ensuring that my contributions are based in requirements rather than assumptions.
- **Needs Improvement - Assertiveness:** Assertiveness matters because good ideas only contribute to a project if they are communicated clearly and at the right time. Staying quiet to avoid conflict or uncertainty allows poor design choices to persist longer than necessary. In senior design, I have

at times held back from raising concerns early enough, particularly in cross-team discussions where I am less confident in my knowledge of the other subsystem. Going forward, I will make a point to voice concerns or suggestions earlier, even when I am not fully certain, and frame them as questions when appropriate so the team can address them before they become larger issues.

Akash:

- **Demonstrated - Thoroughness:** Thoroughness is important to me because in a BLE Link Layer implementation, small details like the CRC initialization value or whitening seed can break the entire system if they are incorrect. I have shown this by carefully studying the Bluetooth Core Specification before implementing any feature instead of making assumptions. I also ensure that each module passes its unit tests before moving on to the next stage of development.
- **Needs Improvement - Communication:** Communication is an area I need to improve. At times, I have encountered blockers but did not communicate them to the team early enough. In a tightly connected system like this project, even small delays can affect other parts of the design. Moving forward, I plan to report issues earlier through Teams so that the team can address problems quickly and avoid delays.

Jackson:

- **Demonstrated - Dedication:** I feel that dedication to your work, whatever that may be is incredibly important. There is no point in doing something if you don't give it your best effort. Despite my own lack of experience with Analog VLSI design, I gave my all to getting up to speed with my teammates so that I can contribute my fair share to completing this project.
- **Needs Improvement - Communication:** While I often find myself working for hours on end towards a goal once I set my mind to it, I tend to not make it known that I am working towards that goal. In a team project like this it is essential to know what you and your team mates are working on at all times to ensure efficiency of work and prevent the possibility of overlapping work. I will aim to better document my work for my team and move towards encouraging more open communication and dedicated analog work meetings in the future.

Dan:

- **Demonstrated - Integrity:** Integrity has always been a critical emphasis for me, since without it, there is very little holding myself, my team, or anyone accountable. I have been honest this whole semester on my abilities, my limitations, and what I can get done in a given timeframe. This is not an easy project, and credit also needs to be given where it is due, whether that be advice or a reference. With the rise of AI especially, it has become increasingly common not to do so.
- **Needs Improvement - Empathy:** Empathy is a core part of existing in society, and something I could always improve on. Oftentimes, it is quite easy to get frustrated with a teammate for any number of reasons, getting wrapped up in negativity rather than trying to clear the air or seek explanations. I will try to keep my head level and an open mind from here on out, placing myself in their shoes before casting any judgement.

Seth:

- **Demonstrated - Resilience:** Resilience is important it me because in every project there will be hardships. Where it is important that we persevere in our efforts to try and further our goals. When a program doesn't work, or data is lost resilience is what is needed to continue on and get the job done. I feel that I have demonstrated resilience when working with new open sourced programs for chip design. I spent many hours fighting with file paths, libraries, and un-intuitive UI

without the resilience to get these programs working we would have not been able to get to where we are in the physical team.

- **Needs Improvement - Humility:** Humility is important to me because it is the virtue that saves time. Humility is knowing when something is out of your reach and to ask for help. In the future I hope to show humility by reaching out to my team when I run into hard issue sooner rather than spending a lot of time on something I cannot achieve on my own. because we are a team and we help each other make this work.

Zane:

- **Demonstrated - Commitment:** Commitment is an important thing especially when it comes to large team projects. I have committed myself consistently throughout the semester in putting in effort to research, design, test, and implement parts of the project.
- **Needs Improvement - Communication:** Communication, especially with a project of this scale is essential. I sometimes found myself late to responding to team messages relating to in person group coordination. I also found that I didn't always talk through certain design decisions. Moving forward, I would like to be more attentive to group communication.

8 Closing Material

8.1 Discussion

8.1.1 Digital Discussion

The first semester has surfaced two findings worth carrying forward on the digital side.

The simulation environment redesign was the most significant pivot. The original C-based sim plan grew unwieldy quickly and forced premature commitment to hardware-style data flow before the DUT behavior was fully understood. Switching to a Python conductor with multiple devices restored flexibility, allowed third-party libraries to generate test benches, and made it practical to compare our DUT against a known-good controller (Zephyr) over the same simulated PHY **[5]**. The pivot cost calendar time but is paying off now that advertising and scanning are functional.

The serializer implementation, while modest in scope, lays the groundwork for future module development and testing. Future modules will use cocoTB to generate the test benches as well, providing another layer of ensured correctness.

Open questions for next semester include how cleanly the Python DUT will translate to RTL once the simulation environment is feature-complete, and how will the DUT interface with our analog RF interface properly?

8.2 Analog Discussion

Quite a few points have emerged on the analog side this semester. As proven in Simulink, the general architecture should be achievable, but there are quite a few bottlenecks and hurdles before a fully functional design can emerge. The open source toolflow itself is far less intuitive than its industry counterpart. The most critical there, though, is the speed of parasitic calculations. While we

may manage to get some or most of the device layout complete, some calculations supposedly take many hours, perhaps even days. With the few short months given towards this project, this severely hinders the ability to revise our designs effectively.

Another pivot point is regarding on-chip inductors. While theoretically possible, they are not actively characterized in the PDK, and consistent, post-tapeout results are seemingly impossible to find. As a result, devices like the VCO may have to pivot to either off-chip inductors, or other architectures, such as a ring oscillator.

8.3 Conclusion

In this semester we have made significant progress towards working simulation models. On the digital side, the gold model is currently capable of the advertising and scanning state. This, combined with the FPGA advertiser, builds a solid base of testing for next semester. This provides a clear pathway towards creating a working hardware design. Similar progress has been made on the analog side towards modeling the entire system inside of Simulink.

Largely, we were constrained by an inability to create interfaces. The BLE standard only provides two defined interfaces. One between the Host and Controller, and one between each device. This means the design needs to decide on interfaces between the HCI, LL, and PHY as well as all of the modules. This leads to a situation where the requirements of a module can be described in a paragraph, but the ports are not entirely defined because the total scope is hard to capture without creating the module. This makes it difficult to split up work because the ports are not fully defined until it is created. This led us to try to split up work by feature instead of by module. This helped get the design to its final state.

Looking forward, the development of the hardware will probably go smoother than the design of the simulated environment. This is because the modules and function will be well defined before the start of hardware development. This will allow for the work to be split up easier than it was in this semester.

On the analog side, it was a slow start in finding the direction in which we wanted to design our system. With the guidance of faculty and trial and error, we came to our current proposed system level design. With near completion of the Simulink model, next semester will allow for the start of component design with a strong confidence in functionality.

8.4 References

[1] Bluetooth Sig, “Bluetooth Low Energy 4.0 Core Specifcation.” [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-4-0/

[2] “ISO/IEC/IEEE International Standard for Information technology--Microprocessor systems--Control and Status Registers (CSR) Architecture for microcomputer buses,” Oct. 05, 1994, *IEEE*.

[3] “IEEE Standard for Low Low-Rate Wireless Networks,” Dec. 12, 2024, *IEEE*.

[4] Google, “Python Bluetooth Host Stack.” [Online]. Available: https://google.github.io/bumble

[5] BabbleSim, “Physical Layer Simulator for BLE and 802.15.4.” [Online]. Available: https://babblesim.github.io/

[6] Zephyr Project, “Zephyr RTOS Documentation.” [Online]. Available: https://docs.zephyrproject.org/

[7] “nRF52832 Product Specification.” [Online]. Available: https://docs.nordicsemi.com/bundle/ps_nrf52832/page/nrf52832_ps.html

[8] “Apache Mynewt NimBLE.” [Online]. Available: https://mynewt.apache.org/download/

[9] Ibram Shenouda, Noah Thompson, Nathan Stark, Nolan Eastburn, Ethan Kono, and WIll Custis, “Open-sourced Radio Microcontroller for Fabrication.” [Online]. Available: https://sdmay25-27.sd.ece.iastate.edu/CompleteDesignDocument.pdf

[10] “Bluetooth PLL Design on Skywater's 130nm technology.” [Online]. Available: https://github.com/mabrains/PLL_design

[11] IEEE, “IEEE Code of Ethics.” [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

9 Team

9.1 Team Members

- Zane Salti
- Parker Pederson
- Tyler Bibus
- Seth Klaassen
- Daniel Pytel
- Jackson Doyle
- Akash Gojuru
- Aiden Han-Lindermyer

9.2 Required Skill Sets for Your Project

- **Digital Design and Digital VLSI:** Required to implement the BLE link layer and HCI in SystemVerilog, synthesize the design using the Sky130 open-source toolflow, and complete place-and-route for tapeout. Directly tied to the fabrication requirement and all BLE controller functional requirements (BLE c.1–c.3, c.97, c.98, c.107).
- **Analog Design and VLSI:** Required for every analog component, including any amplifers, ADCs, DACs, and switches. Gain analysis, propagation delay, and parasitic evaluation all critical steps in ensuring a functional device.
- **Embedded Systems:** Required to integrate the RISC-V management core with the BLE controller, configure Zephyr as the host software driver, and modify the HCI software interface to write to the correct memory-mapped addresses. Tied to the requirement for host-controller communication.
- **Version Control:** Required to coordinate parallel development across digital and analog subsystems, manage documentation revisions, and maintain a reproducible open-source project history accessible to future teams. Tied to the open-source design constraint.
- **Signals and Signal Processing:** Required to understand and implement GFSK modulation, data whitening, CRC generation and checking, and the baseband interface between the digital

controller and the analog radio. Tied to BLE 4.0 PHY layer requirements and the GFSK modulation requirement.
- **RF Design:** Required to design the 2.4 GHz transceiver front-end including the mixers and RF switch. Tied to the BLE transmission and reception requirements and the PHY functional requirements.
- **Microwave Engineering:** Required for PLL design, high-frequency layout considerations, impedance matching at 2.4–2.48 GHz, and ensuring acceptable phase noise and channel settling time. Tied to BLE channel frequency accuracy requirements (±150 kHz maximum deviation, 150 µs settling).

9.3 Skill Sets Covered by the Team

Skill Set	Team Member(s)
Digital Design and Digital VLSI	Zane Salti, Tyler Bibus, Parker Pederson, Aiden Han-Lindemyer, Akash Gojuru
Analog Design and VLSI	Jackson Doyle, Daniel Pytel, Seth Klaassen, Zane Salti
Embedded Systems	Zane Salti, Tyler Bibus, Parker Pederson, Aiden Han-Lindemyer, Akash Gojuru
Version Control	All team members
Signals and Signal Processing	Zane Salti, Jackson Doyle, Daniel Pytel, Seth Klaassen
RF Design	Jackson Doyle, Daniel Pytel, Seth Klaassen, Zane Salti
Microwave Engineering	Zane Salti, Daniel Pytel, Seth Klaassen, Jackson Doyle

Table 8: Skill Sets Covered by the Team

9.4 Project Management Style Adopted by the Team

For our senior design project, we are using an Agile methodology. We work in weekly sprints where each team member is responsible for their assigned tasks across the digital, analog, and project management sides of the project. We chose Agile over Waterfall because the BLE Link Layer implementation involves a lot of iteration and discovery along the way things like the CRC implementation, whitening, and connection state machine all required multiple rounds of testing and fixing before they worked correctly, and a rigid upfront plan would not have accounted for that. We track our progress through weekly meetings with our professor, weekly status reports and presentation slides, and a shared Excel time-tracking sheet where everyone logs their hours each sprint.

9.5 Initial Project Management Roles

- **Zane Salti** - Project Lead
- **Parker Pederson** - Digital Team Lead
- **Tyler Bibus** - Digital Simulation Lead
- **Jackson Doyle** - Analog Team Lead
- **Akash Gojuru** - Digital Design Lead
- **Aiden Han-Lindemyer** - Digital Testing Lead
- **Seth Klaassen** - Analog Layout Coordinator
- **Daniel Pytel** - Analog Team Coordinator

9.6 Team Contract

Team Name Radio Microcontroller

Team Members:

1) Zane Salti
2) Jackson Doyle
3) Daniel Pytel
4) Tyler Bibus
5) Parker Pederson
6) Akash Gojuru
7) Seth Klaassenn
8) Aiden Han-Lindemyer

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    a. In person: Wednesday 10:00-12:00 (Coover Hall) and Friday 3:30-4:30 (Advisor Meeting; time may change)
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
    Teams/Discord will be the primary mode of contact, with SMS as backup.
3. Decision-making policy (e.g., consensus, majority vote):
    Talk it out. Try to get everyone on the same page (consensus).
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
    a. Record meetings using Teams.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    a. Each member should communicate whether they can attend meetings
    b. Ideally attend at least one meeting per week
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    a. Collaboration on team assignments
    b. Deadlines respected, communication when there are concerns regarding deadlines
3. Expected level of communication with other team members:
    a. Communicate with other team members when it's relevant for other members work
    b. Communicate with other team members if stuck on an issue
4. Expected level of commitment to team decisions and tasks:
    a. Everyone is expected to contribute to deliverable write-ups.
    b. Each member should have input on decisions that directly affect or are adjacent to their technical work.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.): *flexible
    a. Zane: Project lead, meet with and communicate with client, high level architecture designer (top level design of system and radio front end), link layer (digital side of controller) implementation, maybe some analog component implementation.

    b. Tyler: Meet with and communicate with client, implementation of link layer, integrating zephyrOS with ble library on management core.
    c. Jackson: Individual component design, meet and communicate with client, primarily analog implementation
    d. Dan: Manage team deliverables, PLL and other analog design, “bridge” between teams
    e. Akash: Verification and testing of digital ASIC blocks.
    f. Aiden: Link layer implementation and digital subsystem development; RTL design, verification, and integration.
    g. Parker: ZephyrOS driver support; Link layer design and integration.
    h. Seth: Analog design and implementation. With design testing and verification.
2. Strategies for supporting and guiding the work of all team members:
    a. Share results and have several members review and comment on designs
        i. Create Teams channel for feedback
    b. Task board in teams regarding aspects/issues that need to be addressed
3. Strategies for recognizing the contributions of all team members:
    a. Our progress can be discussed during meetings
    b. Weekly report to advisor/client should include individual progress from each member

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
    a. Zane: Hardware design, microwave engineering, firmware development, experience with open source ASIC development (developed a mixed design project for tapeout), FPGA accelerator design, interest in radio architecture and design, experience with working with radio communications.
    b. Tyler: Digital Logic Design, Software for wireless communication, programming implementation using existing libraries/documentation.
    c. Dan: VLSI experience, some communications systems knowledge, VHDL, technical writing experience
    d. Parker: Accelerator and peripheral design for micro controllers, Basic driver implementation for application-specific accelerators, application-specific pipeline design, and some Cadence experience.
    e. Akash: Digital Logic Design, Accelerator design and implementation on FPGA, Firmware development, and RTL and FSM using VHDL and Verilog
    f. Aiden: VHDL, embedded systems experience, firmware development, python based data analysis, RTL implementation, pipeline design.
    g. Seth: experience with hardware and digital design using VHDL and Verilog, as well as cadence. Is good with embedded systems
    h. Jackson: Experience with cadence, communications, verilog. Some VLSI experience.
2. Strategies for encouraging and supporting contributions and ideas from all team members:

    a. Each team member should actively seek out ideas from those working around their area of expertise.
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?)
    a. Discussion with "team leaders" first, and if needed, can be brought to the attention of the whole team to discuss resolutions.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    a. Develop a concrete scope for the project
    b. Have a complete and settled on radio architecture to aim for implementation
    c. Divide into teams and develop a process pipeline
    d. Major implementation of digital side of controller (Link Layer)
        i. Verification via simulation and on an FPGA
        ii. Testing on an FPGA interfacing with a potential retail transceiver
    e. Make progress on components for the May tapeout deadline.
2. Strategies for planning and assigning individual and team work:
    a. Discuss needed tasks and have tasks allocated.
    b. Keep track of tasks and responsibility on Teams' integrated taskboard.
3. Strategies for keeping on task:
    a. Use Teams' integrated taskboard for keeping track of progress and tasks. Try to have defined goals and subgoals to be achieved within a set time.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    a. Have a meeting and discuss issues instead of ignoring them.
    b. Document the infractions and intervention through written communication.
2. What will your team do if the infractions continue?
    a. Address the infractions with the advisor, imposing potential penalties including grade infractions and, worst-case, removal from the team.

****

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) *Daniel Pytel* DATE **02/10/2026**
2) **Tyler Bibus** DATE **02/10/2026**
3) **Aiden Han-Lindemyer** DATE **02/10/2026**
4) **Akash Gojuru** DATE **02/10/2026**
5) **Seth Klaassen** DATE **02/10/2026**
6) **Parker Pederson** DATE **02/10/2026**
7) **Jackson Doyle** DATE **02/10/2026**
8) **Zane Salti** DATE **02/10/2026**