**University of Central Florida**

**Department of Electrical & Computer Engineering**

**EEL 4914**

**Senior Design 1: Divide and Conquer**

****

***Authors: Instructor:***

Davi Dantas (ECE) Dr. Chung Yong Chan

Evan Kasky (CPE) ***Sponsor:***

Nicolas Sayed (ECE) Northrop Grumman Corp.

Pablo Rodriguez (CPE) ***Review Board:***

Timothy Ogg (CPE) Aaron Lingerfelt (NGC)

Youssef Samwel (ECE) Alberto Reategui (NGC)

Marlon Fuentes (NGC)

Dr. Mike Borowczak

**September 5th, 2024**

**Chapter 2 - Project Description**

**2.1 - Project Background and Motivation**

Our senior design group is challenged with designing an Application-Specific Integrated Circuit (ASIC), specifically focused on the design and synthesis of a RISC32I Central Processing Unit (CPU) core. A complete CPU comprises several computational units, along with various interconnects to external hardware such as RAM and PCIe devices. Our project zeroes in on the core computational unit of the CPU, which is responsible for executing the user’s instructions—ranging from logic and arithmetic operations to memory read/write functions. These fundamental tasks lie at the heart of modern digital computing.

Our CPU core will implement the RISC32I Instruction Set Architecture (ISA), which defines the interface between software and hardware. The RISC32I ISA is an open-source standard that already has established compiler support. This is a key advantage, as it allows us to leverage existing tools such as operating systems, compilers, linkers, APIs, and software. By utilizing a well-supported ISA, we sidestep one of the primary challenges to adopting new CPU architectures: the availability of toolchain support. A powerful CPU architecture can be held back if the necessary development tools—compilers, debuggers, and other software—are not readily available. Our choice of the RISC32I ISA ensures that we can tap into a robust ecosystem of existing technologies, facilitating a smoother development process and increasing the potential for adoption.

Semiconductor fabrication and design are integral to the United States' economic stability, national security, and supply chain resilience. A disruption in the steady supply of chips can have significant consequences, as demonstrated by the global semiconductor shortage triggered by the COVID-19 pandemic in 2020. Industries ranging from defense to automotive manufacturing experienced substantial setbacks due to port closures in key production regions. This disruption highlighted the dependency of modern industries on a reliable supply of chips. For example, the automotive sector faced production halts as critical components became unavailable, and the defense industry experienced delays in both maintenance and production of essential equipment. This project aims to address such vulnerabilities by advancing the design of Application-Specific Integrated Circuits (ASICs), contributing to a more secure and self-sustaining semiconductor ecosystem.

This project aims to contribute to the growth of a more resilient semiconductor industry in the United States by providing UCF students with hands-on, industry-level experience in ASIC design. By equipping students with the skills necessary for ASIC development, this initiative helps cultivate a larger pool of specialized talent within the U.S. semiconductor workforce. Additionally, this project establishes a pipeline of UCF graduates who are well-prepared to pursue careers in ASIC design, strengthening the domestic talent base and supporting the nation's technological leadership.

**2.2 - Goals and Objectives**

The goal of this project is for students to successfully design an Application Specific Integrated Circuit (ASIC) that implements the RISC-V Instruction Set Architecture, utilizing industry-standard electronic design automation (EDA) tools and relevant Cadence software. The project is divided into two phases: the Fall phase and the Spring phase, with specific objectives outlined for each stage.

In Phase 1 (Fall 2024), the project involves designing an ASIC that implements the RISC-V RV32I base instruction set, excluding memory ordering and environment/system calls. The design must undergo functional and/or formal verification to confirm its functionality, utilizing the Universal Verification Methodology (UVM) as a standardized approach, with a target of achieving a minimum of 85% code coverage. The goal is to successfully tape out an ASIC that implements the RISC-V instruction set, using existing electronic design automation (EDA) tools and relevant design technology libraries, ideally at a process node of approximately 45nm or smaller. The team is expected to gain knowledge about the ASIC design flow, become proficient with Cadence tools, enhance troubleshooting skills, and develop thorough project documentation. To validate the performance and system-level functionality of the design, the team will create and execute test cases in RV32I assembly.

In Phase 2 (Spring 2025), the project will involve conducting static timing analysis (STA), IR drop analysis, and electromigration analysis to ensure that the physical performance requirements are met. The targets for this phase include a maximum IR drop of 5%, no timing failures, and maintaining the current density within acceptable limits. Additionally, the finalization of functional and/or formal verification of the design will be completed, ensuring that the target code coverage is achieved. Physical design techniques such as design rule checking (DRC) will be used to confirm the manufacturability of the design, and layout versus schematic (LVS) checks will be performed to ensure the correctness of the physical design.

The stretch goals for the project include inserting design for test (DFT) structures to validate the functionality of the manufactured IC. Additionally, gate-level simulations will be performed to verify the design with real clock delays. Logic equivalence checking will also be used to ensure functional equivalence between different stages of the development process.

Regarding deliverables for phase 1 and phase 2 are as follows: Mock tapeout a chip using a maximum 45 nm technology with Cadence digital design, verification, and signoff tools. In Phase 1, the deliverables should include the RTL design, initial verification metrics, and the synthesized netlist. In Phase 2, deliverables should include finalized verification metrics, a GDSII file, a clean timing report, and IR drop and electromigration within specifications. Leverage Cadence resources, including training materials and Rapid Adoption Kits (RAKs), to familiarize students with the ASIC design flow and accelerate project completion.

**2.3 - Description of Features and Functionalities**

Processing 100 million instructions per second (100 MIPS) within the operating range of 0.95 -1.25V and 0-125 C and consuming a maximum of 1W. The design should incorporate an L1 Cache of at least 16 MB.

**2.4 - Existing Products Used to Identify Project Features**

To better understand our goals and objectives in this project, we need to dive into the history to have context around what is trying to be accomplished. In today’s world, most computers and phones are based on the Instruction Set Architecture (ISA) of x86. The x86 is a closed-source ISA that requires users to pay to be able to use the software. An x86 can be categorized as a Complex Instruction Set Computer (CISC), so it has a very large set of instructions that are capable of performing complex operations. There are several complex operations that are able to perform such as Load String, Move with Sign-Extension, and Repeat String Move. All these operations would not be as simply written in a Reduced Instruction Set Computer (RISC) architecture.

In x86 architecture, the variable length instructions are between 1 and 15 bytes, which allows for diverse functions in exchange for added complexity to instruction decoding. X86 typically has a smaller number of general-purpose registers which could require more frequent memory access which would make it slower.

An alternative today to the x86 is the ARM processor which is also a closed source ISA. ARM is a RISC architecture and it has a fixed-length instruction set. This architecture aims to perform one instruction every cycle for better efficiency. ARM has more general-purpose registers than x86 so it can access memory faster. ARM architecture performs pipelining more efficiently because of its simpler nature.

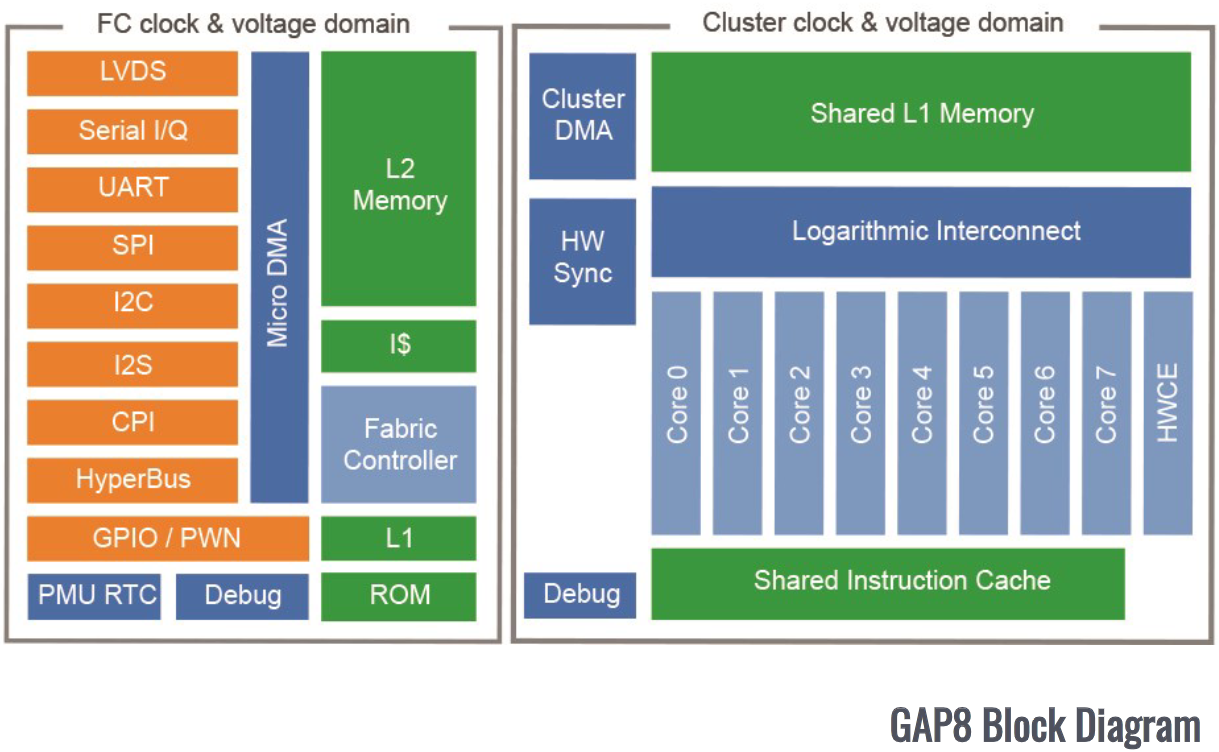
RISC-V is an open-source ISA, making this software free to design and sell processors with RISC-V ISA without constraints. Regarding architecture, RISC-V is more closely related to ARM. Originally there was a RISC-I created at the University of California Berkeley. There are 32-bit options as well as 64-bit with the option of extending it to 128-bit. The most basic ISA version within every category is the base integer (e.g. RV32I and RV64I) There are extensions to expand the features such as M- Integer Multiplication and Division, A - Atomic Instruction, F - Single-Precision Floating Point, D - Double-Precision Floating Point, Q - Quad-Precision Floating-Point and C - Standard Extension for compressed Instructions.

Several commercially available products make use of RISC-V due to it being an open instruction set architecture (ISA) suitable for use in academia, research, and capable of being directly implemented in hardware. The ISA is separated into the RV32I Base Integer Instruction and supports several extensions for additional functionality. RISC-V implementations that only meet the RV32I set are adequate for simple ASICs or educational purposes. These products give an insight into the common characteristics of RISC-V implementations.

**GreenWaves GAP8**

The GreenWaves GAP8 IoT Application Processor is a RISC-V multi-core, low-power, battery operated data analysis processor capable of collecting data from image sources, radar, sound and other signal sources. It is also capable of performing AI-driven workloads and is viable to perform tasks such as image recognition, speech recognition, and sensor monitoring.

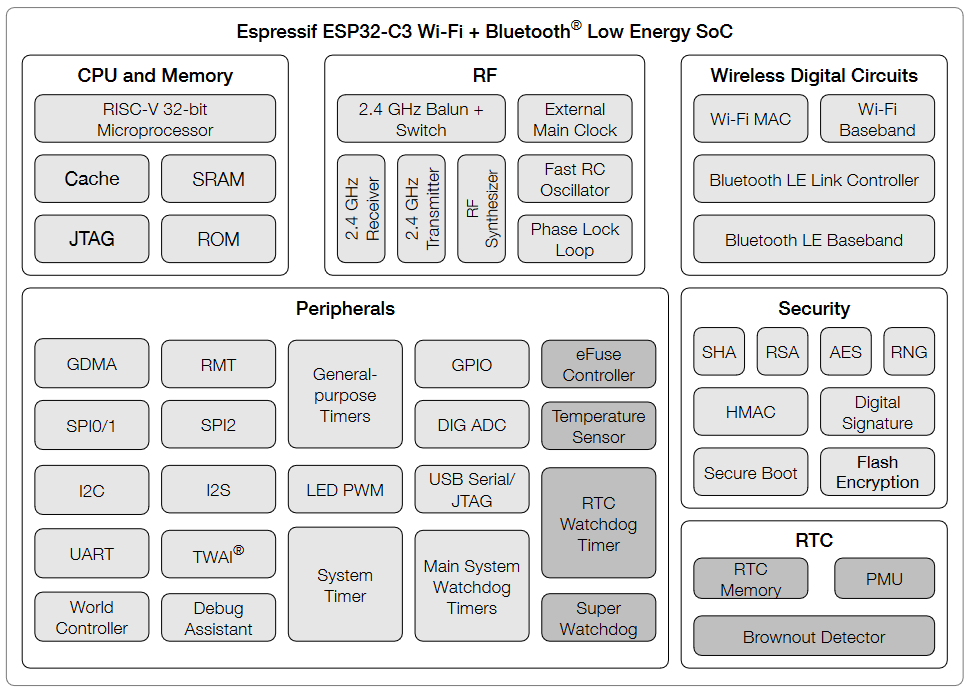
This product can operate at up to 175 MHz, within an operating voltage range of 1.0 V to 1.2 V with an operating power of about 70 milliwatts. It is capable of delivering up to 200 MOPS within these limits. It implements a 512 kB L2 cache shared across its cluster of eight cores. It is fabricated with a 55nm process. The GAP8 has found uses in the application areas of drones, surveillance cameras, face detection, robotics, among others.



**Espressif ESP32-C3**

The ESP32-C3 is a low-power MCU with a single 32-bit RISC-V processor that incorporates Wi-Fi and Bluetooth wireless communication interfaces. The product has seen success in industrial automatic, consumer electronics, audio processing devices, and sensor data loggers, among others use cases.

The MCU’s recommended input voltage lies between 3.0 V and 3.6 V which drives the core at up to 160 MHz. It incorporates a 16kB cache and has an average power consumption when active of about 80 milliwatts. This MCU is fabricated with a 40 nm process.



**Other products**

The following table outlines some of the characteristic of various popular commercially available RISC-V processors:

***Table 2.1: Different RISC-V processors***

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Fabrication Process** | **Clock speed** | **Miscellaneous notes** |
| SiFive FE310 | 180 nm | 320 MHz | 5-stage pipeline |
| Renesas RH850/U2B | 28 nm | 400 MHz | Single issue, 2-stage pipeline |
| Bouffalo Lab BL602/BL604 | 22 nm | 192 MHz | 3-stage pipeline |
| GigaDevice GD32VF103 | - | 108 MHz | Single issue, 2-stage pipeline |

**Summary**

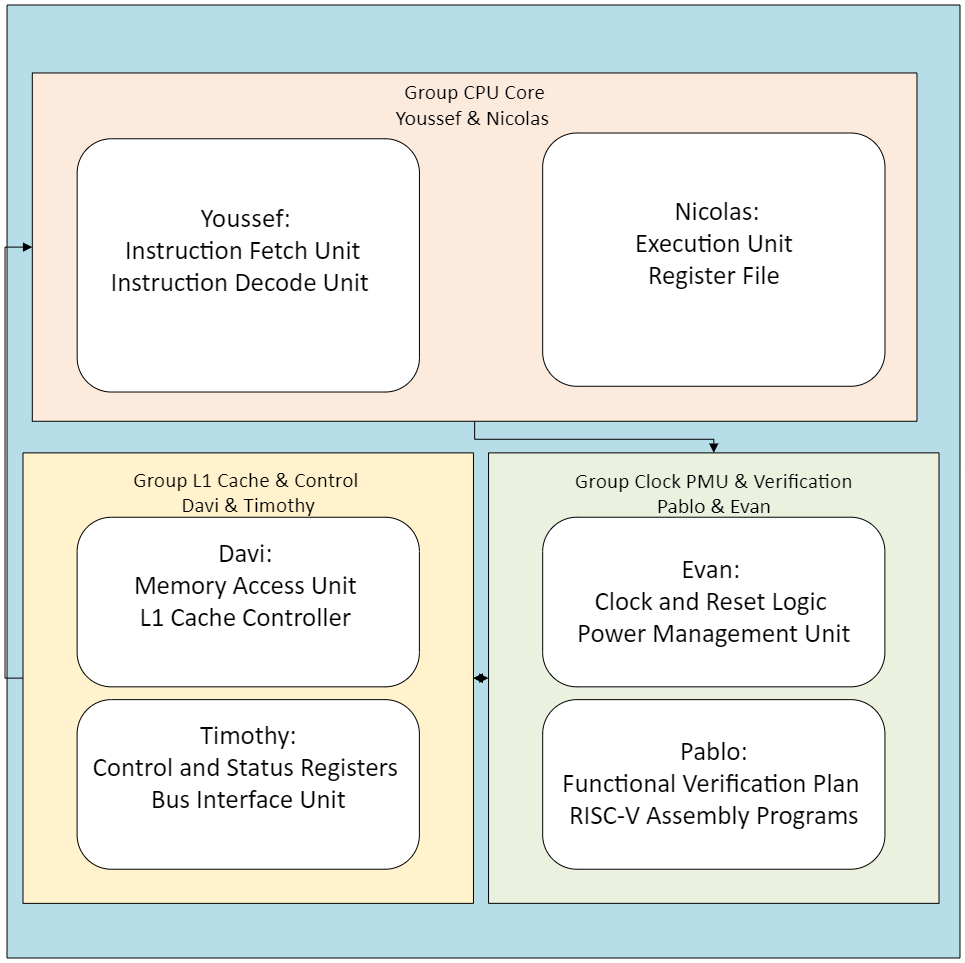
The aforementioned products reveal some of the common aspects of commercially available RISC-V SoCs. First, they appear to be well-suited for signal processing, so they are a natural choice to pair with sensors for the purposes of data logging. Second, RISC-V SoCs typically operate at clock speeds between 100 MHz and 400 MHz and are able to operate within 100 milliwatts for the lower clock speeds. Third, instruction-level parallelism using pipelines is always implemented in order to achieve higher throughput, however, superscalar designs are uncommon for this range of processors.

**2.5 - Engineering Specifications (Table)**

***Table 2.2: Requirements and Specifications***

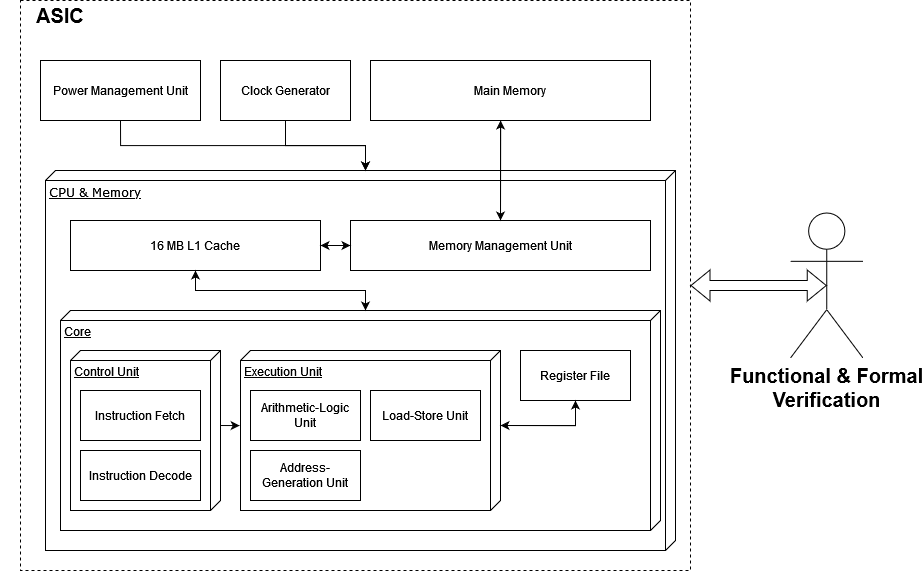
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Requirement** | **Specification** | **Description** | **Priority** |
| **1** | Processing Speed | ≥ 100 MIPS | The design should be capable of processing at least 100 MIPS for the required application. | High |
| **2** | RV32I Compatibility | ≥ 38 unique instructions | The design should be able to handle at least 38 unique instructions, excluding memory ordering and environment calls. | High |
| **3** | Low Power Consumption | ≤ 1W | The design should be efficient and meet all specifications with less than 1W of power consumption. | High |
| **4** | Cache size | ≥16 MB | The design should incorporate an L1 cache of at least the specified size. | High |
| **5** | Process Node | ≤ 45nm | The design should be implemented with a process size of at most 45nm, this will allow for better performance and efficiency than a larger size. | High |
| **6** | Operating Voltage Range | 0.95 V - 1.25 V | The design should operate correctly within its permitted voltage range. | High |
| **7** | Operating Temperature Range | 0 °C - 125 °C | The design should operate correctly within its permitted temperature range. | High |
| **8** | IR Drop | ≤ 5% | The design should not allow voltage drops below 5% of the expected voltage. | High |
| **9** | Timing Failure Count | 0 Faults | The design should not have any timing faults over any operating period. | High |

**2.6 - Hardware Diagram**

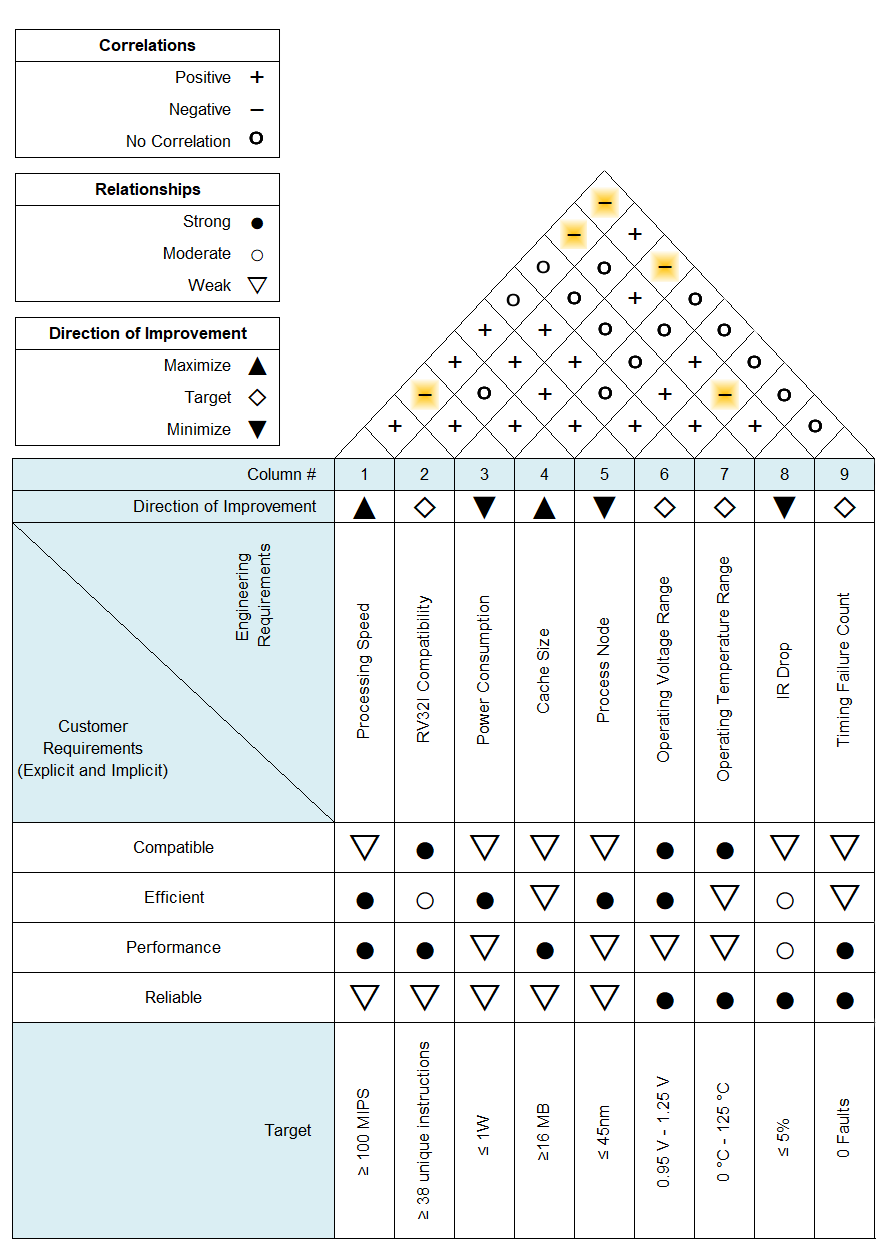
***Table 2.3: Hardware Diagram***

**2.7 - Software Diagram**

***Table 2.4: Software Diagram***

******

**2.8 - House of Quality**



**Chapter 10 - Administration Content**

**10.1 Budget and Financing**

This project does not require any hardware, and any software not currently accessible to UCF students will be provided by Northrop Grumman.

**10.2 Project Milestones**

***Table 10.2.1 Phase 1: Fall semester***

|  |  |  |
| --- | --- | --- |
| Week 1; September 9 -13 | Meeting With Northrop Grumman team | * Student team meets with NG team * Q&A project goals, objectives and expectations * Review project timeline and deliverables * Setup GitHub for all members |
| Weeks 2 - 3; September 23- October 4 Deadline: September 27, 2024 | Initial Design Specs and Block Diagram | * Detailed project specifications * Development of the High-level block diagram of ASIC design * Reassess block responsibilities to team |
| Week 4 - 6; | RTL Design Commencement | * Reassess coding standards and documentation * Begin writing HDL code for the RISC-V base instructions set * Review RTL among team members |
| Week 7; | Functional Verification Plan | * Develop verification plan using SystemVerilog and UVM at the center * Create testbenches and test cases for RTL functionality verification * Identify metrics to track verification progress |
| Week 8 -10; | RTL Design Finalization | * Completion of RTL for all modules * Perform code reviews and fix identified issues (bugs) * Make sure all blocks have proper integration and the design demonstrates effectiveness |
| Week 11 - 13 | Functional Verification Start | * Use planned functional simulations testbenches on the completed RTL design * NG labeled goal: Achieve initial functional verification with a focus on reaching 85% code coverage. * Identify and address any bugs or discrepancies. |
| Week 14 | Project Review | * Progress presentation to Northrop Grumman and advisors * Presentation must include RTL design, verification results and challenges faced and overcome * Discuss project plans for next phase * Receive feedback for adjustments for the project plan for the next phase |
| Week 15 -16 | Synthesis and Preliminary Timing Analysis | * Synthesize the RTL design into a gate level netlist using Cadence (Requires training) * Learn and perform preliminary static timing analysis to identify timing issues * Prepare Place and route design for physical design in |

***Table 10.2.2 Phase 2: Spring Semester***

|  |  |  |
| --- | --- | --- |
| Week 1 - 4 | Place and Route Process | * Begin physical design process using Cadence * Perform floorplanning, placement and routing of the up to date synthesized netlist * Ensure that the design meets area, power, and timing constraints * Work closely with advisors and NG engineers to achieve optimal results via recommendations |
| Week 5 - 6 | Design Rule Checks and Layout vs Schematic | * Perform Design Rule Checks to ensure manufacturability * Perform Layout Versus Schematic checks to verify layout correctness * Address any violations |
| Weeks 7 - 8 | Final Timing and Power Analysis | * Reperform extensive static timing analysis to ensure the design meets timing requirements * Conduct IR drop and electromigration analysis using Cadence * Optimize power distribution and ensure the design operates within specified power limits |
| Week 9 - 10 | Design for testability Insertion and Testing Plan | * Insert Desing for Testability (DFT) structures, such as scan chains and BIST. * Develop a detailed testing plan for post-fabrication validation. * Simulate the design with real clock delays using gate-level simulations (Stretch Goal). |
| Week 11 - 12 | Final Design Sign Off and Tapeout Prep | * Ensure all Design Rule Checks, Layout Versus Schematic, timing and power checks are clean (for the most part) and signed off * Prepare the GDSII file and other required documentation for tapeout * Review the design with NG team and advisors for approval |
| Week 13 | Final Project Review and Documentation | * Prepare final presentation with final design, verification results, and tapeout preparation * Complete and finalize all project documentation which includes the design report, verification reports, and lessons learned by all individual members. |
| Week 14 | Mock Tapeout | * Perform a mock tapeout of the design to simulate the actual tapeout process (Brainstorm and discuss further on this). * Ensure everything is ready for the tapeout;this includes all files and docs * Conduct review of project process * Meet with NG team and advisors |
| Week 15 - 16 | Project Presentation | * Final presentation of the project to NG team, advisors and class * Highlight achievements, challenges, and experience as a whole |

**10.3 Work Distributions (table, primary and secondary)**

***Table 10.3.1: Group 1 members and responsibilities***

|  |  |
| --- | --- |
| **Group 1** | **Responsibilities** |
| **Youssef Samwel (ECE)**  **Team 1 lead** | Instruction Fetch Unit (IFU) |
| Instruction Decode Unit (IDU) |
| Develop logic to fetch instructions from memory. |
| **Nicolas Sayed (ECE)**  **Team 1 member** | Develop Execution Unit and Register File |
| Design the ALU |
| Implement Logic to update registers |

***Table 10.3.2: Group 2 members and responsibilities.***

|  |  |
| --- | --- |
| **Group 2** | **Responsibilities** |
| **Davi Dantas (ECE)**  **Team 2 lead** | Designing the Memory Access Unit |
| Ensure load/store functions properly |
| Implementation of cache policies |
| **Timothy Ogg (CPE)**  **Team 2 member** | Design Control and Status Registers |
| Focus on Bus Interface |

***Table 10.3.3: Group 3 members and responsibilities.***

|  |  |
| --- | --- |
| **Group 3** | **Responsibilities** |
| **Evan Kasky (CPE)**  **Team 3 lead** | Designing the Clock and Reset Logic |
| Designing Power Management |
| Implement Clock Generation |
| **Pablo Rodriguez** **(CPE)**  **Team 3 member** | Develop functional verification |
| Write testbenches to verify each individual component |
| Write RISC-V assembly program to test CPU functionalities. |

**Appendix A - Reference**

**“Arm Cortex-M4 Processor Technical Reference Manual Revision R0p1.” Accessed September 5, 2024.** [**https://developer.arm.com/documentation/100166/0001/Introduction/About-the-processor**](https://developer.arm.com/documentation/100166/0001/Introduction/About-the-processor)**.**

**“The RISC-V Instruction Set Manual Volume I - Version 20240411”**

**Accessed September 5, 2024.**[**https://github.com/riscv/riscv-isa-manual/releases/download/20240411/unpriv-isa-asciidoc.pdf**](https://github.com/riscv/riscv-isa-manual/releases/download/20240411/unpriv-isa-asciidoc.pdf)

**“Greenwaves GAP8 IoT Product Brief”**

**Accessed September 5, 2024.**[**https://greenwaves-technologies.com/wp-content/uploads/2021/04/Product-Brief-GAP8-V1\_9.pdf**](https://greenwaves-technologies.com/wp-content/uploads/2021/04/Product-Brief-GAP8-V1_9.pdf)

**“ESP32-C3 Series Datasheet Version 1.8”**

**Accessed September 5, 2024.**

[**https://www.espressif.com/sites/default/files/documentation/esp32-c3\_datasheet\_en.pdf**](https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf)

**“Muxup. Commercially Available RISC-V Silicon”**

**Accessed September 5, 2024.**

[**https://muxup.com/2023q1/commercially-available-risc-v-silicon**](https://muxup.com/2023q1/commercially-available-risc-v-silicon)