Title : FlexRisc Platform

CEG 4912 – Computer Engineering Design Project I
Fall 2024
School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa
Professeur: Dan Ionescu
Teaching Assistants: Hamideh Ghanadian, Ahmed Hassanein, Haseeb Ur Rehman, Armina Faghihi

Submitted by : Groupe #10
Leader: Hakim Chergui - 300173357
Christopher Krayem - 300212035
Maro Mohamed Sherine Zaher Abdine - 300222317
Simon Marchildon - 300242291
Onel Valery Mezil - 300260630
Adam Taktek - 300110268

Date submitted : 09-13-2024
Date received : 09-13-2024

Reviewed by :
Approved/Disapproved :
Signature (group members) : Chergui Hakim, Maro Abdine, Adam Taktek, Simon Marchildon, Onel Valery Mezil, Christopher Krayem
Signature (faculty) :

Project Rationale :

This project aims to design a "flexible, powerful, cost-effective" FPGA-based RISC-V prototyping platform with an "Arduinolike" approach. By developing a flexible SoC prototyping board focused on supporting Ada development in embedded systems, the platform will bridge the gap between hardware and software domains by providing a robust, adaptable environment that facilitates seamless integration, configuration, and expansion. Featuring a range of critical components, including high IO availability, Persistent RAM, HyperRAM, DDR memory, and cost-effective FPGAs (e.g., Artix7), the system will support a wide array of applications and configurations.

The project will ensure compatibility with open-source tools (e.g., Yosys) for FPGA bitstream creation and develop an infrastructure for SoC deployment, utilizing Ada's package manager to streamline the process. It is important to note that the project will primarily focus on packaging existing and functioning open-source IPs. Simple extension boards will be designed to demonstrate the board's capabilities, with thorough documentation provided, including templates for the EDA of extensions and a pinout sharing system.

This project stands to benefit researchers by offering a cost-effective alternative to current boards (e.g. Arduino), with enhanced flexibility and Ada integration. Developers focusing on safety-critical systems will also benefit from the high-integrity software practices enabled by Ada/SPARK. Additionally, the project will contribute to the growing body of open-source hardware solutions, creating a reusable platform for future developments.

The project is funded by **AdaCore** and overseen by the Chief Engineer **Olivier Henley**.

Project goals :

- Develop the SoCs Prototyping Board: Focus on delivering a cost-effective and highly flexible platform for Ada-based embedded development.
- Integrate multiple components such as a FPGA hosting a RISC-V core, memory modules, IO ports and more, on a single board.
- Create SoC Deployment Infrastructure: Implement an open-source infrastructure for FPGA development, with a focus on Ada integration.
- Design and Document Extension Boards: Create and document simple extension boards to demonstrate and expand the platform's capabilities.
- Showcase a Working Example: Deploy a RISC-V SoC with Ada firmware as a reference implementation to demonstrate the board's full potential.

Project Description

Functional requirements :

- A flexible SoCs prototyping board featuring high IO availability, HyperRAM, DDR memory, FPGA (ECP5/Artix7), multiple voltage rails, and configurable clocks.
- Infrastructure to support FPGA bitstream creation using open-source tools.
- Support for Ada's package manager for SoC deployment.

Components :

- LED (BLUE)
- Programming status LED (RED)
- Power LED (GREEN)
- Xilinx XADC Analog Input Connection
- JTAG header
- LED (ORANGE)
- LED (GREEN)
- LCD Screen or Seven Segment Screen

- Switches and/or Buttons
- Power supply
- Temperature Sensor
- Motor Controller
- Rotary Encoder
- External FPGA Memories: HyperRAM/PSRAM, MRAM, DDR4 SODIMM slot
- IO Matrix Multiplexer for flexible pin assignments
- USB port

Non-functional requirements :

- Modularity: Design should allow seamless integration with extension boards.
- Documentation: Comprehensive documentation, including pinouts, design templates, and user guidelines.
- Cost-Effectiveness: Prioritize components that deliver high performance at a reasonable cost.

Hardware and Software Architecture :

1. Hardware Design

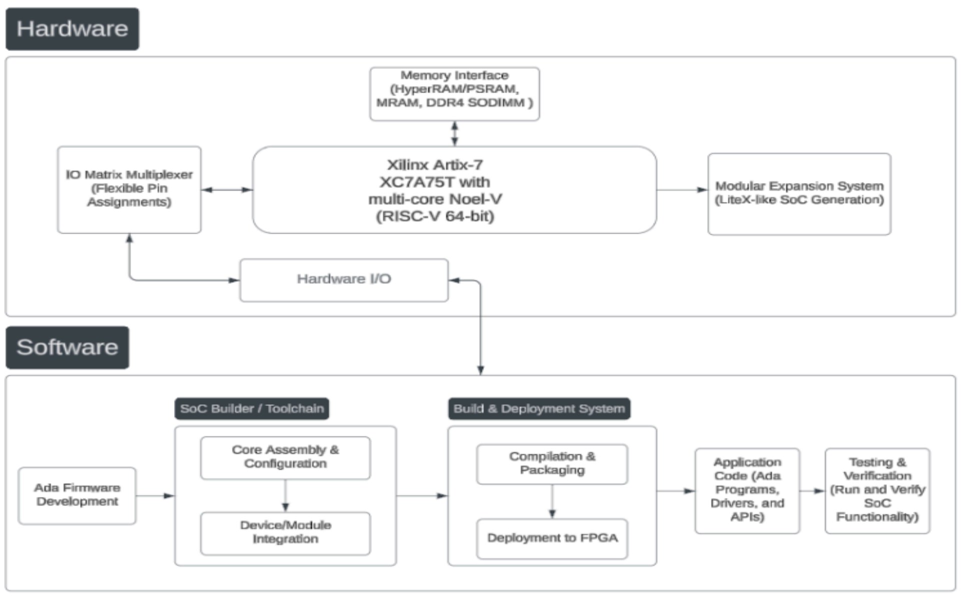
Hardware
Memory Interface (HyperRAM/PSRAM, MRAM, DDR4 SODIMM)
IO Matrix Multiplexer (Flexible Pin Assignments)
Xilinx Artix-7 XC7A75T with multi-core Noel-V (RISC-V 64-bit)
Modular Expansion System (LiteX-like SoC Generation)
Hardware I/O
Software
SoC Builder / Toolchain
Build & Deployment System
Ada Firmware Development
Core Assembly & Configuration
Device/Module Integration
Compilation & Packaging
Deployment to FPGA
Application Code (Ada Programs, Drivers, and APIs)
Testing & Verification (Run and Verify SoC Functionality)

Figure1: Hardware & Software design block diagram

1.1. Schematic Design

- **1.1.1. Selection of components (FPGA, memory, multiplexers)**
  Choose the key components with LCSC Electronics for the platform, including the FPGA (Xilinx Artix-7 XC7A75T), memory types (DDR4 SODIMM, HyperRAM, MRAM), and I/O multiplexers.
  Choose additional components with LCSC Electronics for the resistors, capacitors, inductors and more.
- **1.1.2. Design of the electrical schematic for the board**
  Create a detailed electrical schematic with easyEDA that outlines the connections between components and power distribution.
- **1.1.3. Review and validation of the schematic**
  Conduct thorough reviews to ensure the schematic is correct and meets design requirements, then validate it before proceeding.

1.2. PCB Footprint Design

- **1.2.1. Component placement on the PCB**
  Arrange the selected components efficiently on the printed circuit board (PCB), considering factors like signal integrity and power distribution and thermal management.
- **1.2.2. Routing of traces for connections (FPGA, memory, I/O)**
  Route the electrical traces between components, focusing on maintaining signal integrity and minimizing interference.
- **1.2.3. Verification and validation of the PCB design**
  Review and validate the PCB layout, checking for errors or potential design flaws before sending it to fabrication.

1.3. Fabrication and Assembly

- **1.3.1. PCB fabrication by an external supplier**
  Send the validated PCB design to a manufacturer for fabrication.
- **1.3.2. Soldering of components (FPGA, memory, connectors)**
  Assemble the board by soldering the remaining components (those not soldered by the manufacturer), including the FPGA, memory modules, and external connectors.

- **1.3.3. Visual inspection and assembly testing**
  Perform a detailed inspection of the assembled PCB and conduct initial tests to ensure correct assembly and component placement.

1.4. FPGA & RISC-V Processor Programming

- **1.4.1. Download and integration of the RISC-V processor (Noel-V)**
  Program the FPGA with the RISC-V processor, ensuring it is properly integrated into the design.
- **1.4.2. Implementation of memory controllers (DDR4, HyperRAM, MRAM)**
  Develop and integrate memory controllers to interface with the external memory modules.
- **1.4.3. Configuration of the I/O multiplexer**
  Set up the I/O multiplexer to manage flexible input/output connections.
- **1.4.4. Generation of the FPGA bitstream**
  Compile the design and generate the bitstream to program the FPGA with the complete system configuration.

1.5 Testing and Verification of the design

- **1.5.1. Test the whole pcb prototype**
  Power the pcb and evaluate what works.
- **1.5.2. Conduct experiments**
  If the card does not work as expected, conduct experiments:
  - **Netlist Verification**: Compare the netlist generated from the PCB design with the schematic to ensure all connections are correct
  - **Continuity Test**: Ensure there are no breaks in the signal paths.
  - **Short Circuit Detection**: Verify that there are no unintentional short circuits between power and ground or between signal nets.
  - Evaluate the time delay of clock signals on the card
  - Measure the power rails at different distances from the source, to measure the voltage drops.

2. Software Development (Firmware Ada & SoC)

2.1. Ada Firmware Development

- **2.1.1. Development of firmware for FPGA configuration**
  Design and implement the firmware responsible for configuring the FPGA and initializing its internal logic using GNAT Community.
- **2.1.2. Development of the I/O multiplexing matrix management**
  Create the firmware that controls and manages the I/O matrix, enabling flexible pin assignments.
- **2.1.3. Development of the memory management**
  Create the firmware that manages the interaction to the external memory modules like: HyperRAM, Persistent RAM, DDR memory
- **2.1.4. Optimization of the firmware for SoC and hardware modules**
  Fine-tune the firmware for efficient interaction between the SoC, FPGA, and connected hardware components.

2.2. SoC Development

- **2.2.1. SoC design using LiteX or another SoC generator**
  Utilize LiteX or a similar tool to design a system-on-chip (SoC) architecture tailored for the RISC-V core and FPGA resources**.**
- **2.2.2. Implementation of VHDL IP blocks for controllers**
  Develop and integrate VHDL intellectual property (IP) blocks that manage memory controllers and other peripherals.
- **2.2.3. Verification of RISC-V processor integration into the SoC FPGA**
  Ensure the correct integration of the RISC-V processor within the SoC, ensuring compatibility and functionality.
- **2.2.4. Hardware/software interoperability testing**
  Conduct thorough tests to confirm smooth communication between the hardware and software components of the SoC.

2.3. Debugging and Software Validation

- **2.3.1. Unit testing of the firmware**
  Perform detailed unit tests on the Ada firmware to ensure each function operates as intended.
- **2.3.2. Validation of the controller functionalities (DDR4, I/O)**
  Test and validate the performance of memory controllers (DDR4, HyperRAM) and I/O management features.

- **2.3.3. Bug fixing and optimization**
  Identify, debug, and resolve any issues encountered during testing, optimizing the firmware for performance and reliability.

Work Breakdown Structure (WBS)

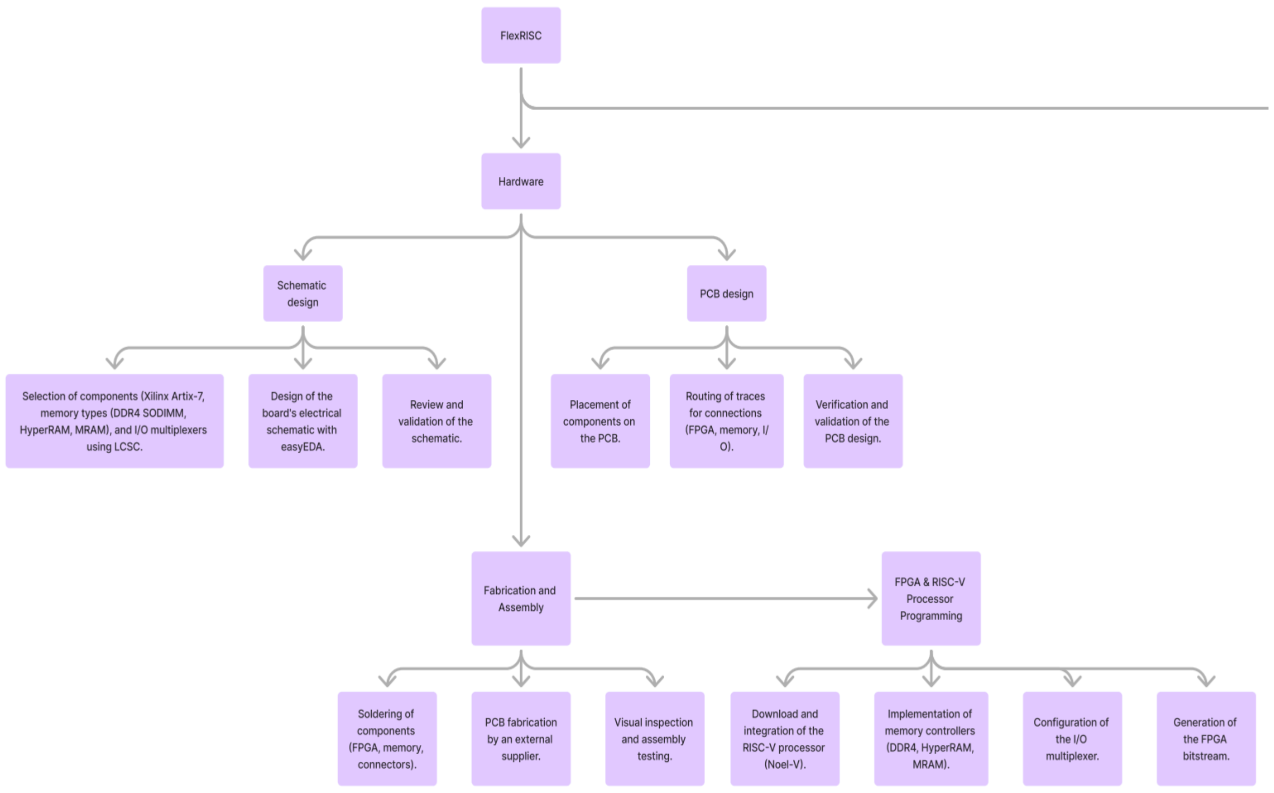
FlexRISC
Hardware
Schematic design
PCB design
Selection of components (Xilinx Artix-7, memory types (DDR4 SODIMM, HyperRAM, MRAM), and I/O multiplexers using LCSC.
Design of the board's electrical schematic with easyEDA.
Review and validation of the schematic.
Placement of components on the PCB.
Routing of traces for connections (FPGA, memory, I/O).
Verification and validation of the PCB design.
Fabrication and Assembly
FPGA & RISC-V Processor Programming
Soldering of components (FPGA, memory, connectors).
PCB fabrication by an external supplier.
Visual inspection and assembly testing.
Download and integration of the RISC-V processor (Noel-V).
Implementation of memory controllers (DDR4, HyperRAM, MRAM).
Configuration of the I/O multiplexer.
Generation of the FPGA bitstream.

Figure 2: Hardware section WBS

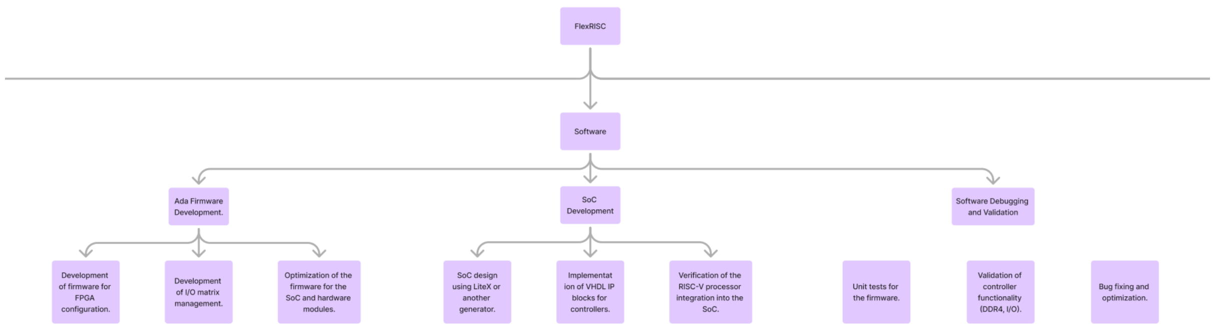
FlexRISC
Software
Ada Firmware Development.
SoC Development
Software Debugging and Validation
Development of firmware for FPGA configuration.
Development of I/O matrix management.
Optimization of the firmware for the SoC and hardware modules.
SoC design using LiteX or another generator.
Implementation of VHDL IP blocks for controllers.
Verification of the RISC-V processor integration into the SoC.
Unit tests for the firmware.
Validation of controller functionality (DDR4, I/O).
Bug fixing and optimization.

Figure 3: Software section WBS

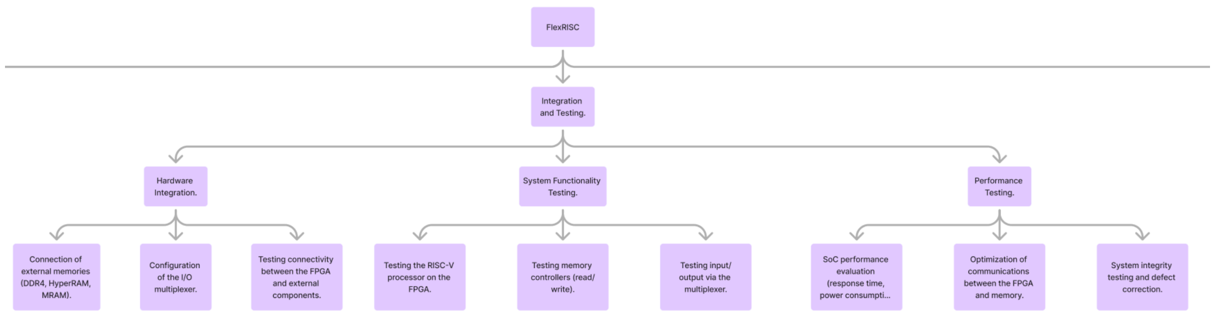
FlexRISC
Integration and Testing.
Hardware Integration.
System Functionality Testing.
Performance Testing.
Connection of external memories (DDR4, HyperRAM, MRAM).
Configuration of the I/O multiplexer.
Testing connectivity between the FPGA and external components.
Testing the RISC-V processor on the FPGA.
Testing memory controllers (read/ write).
Testing input/ output via the multiplexer.
SoC performance evaluation (response time, power consumpti...
Optimization of communications between the FPGA and memory.
System integrity testing and defect correction.

Figure 4: Testing and Integration section WBS

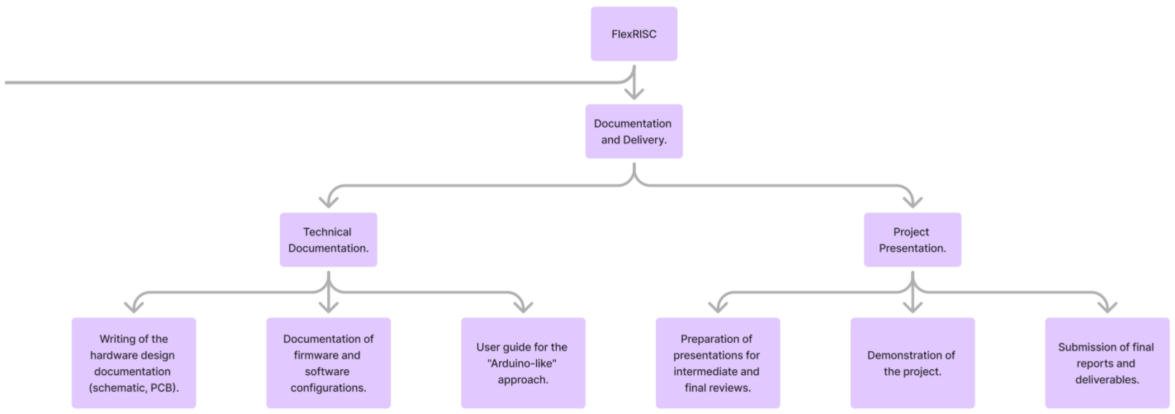
FlexRISC
Documentation and Delivery.
Technical Documentation.
Project Presentation.
Writing of the hardware design documentation (schematic, PCB).
Documentation of firmware and software configurations.
User guide for the "Arduino-like" approach.
Preparation of presentations for intermediate and final reviews.
Demonstration of the project.
Submission of final reports and deliverables.

Figure 5: Documentation and Delivery section WBS

Gantt Chart

- **Phase 1 (Weeks 1-3)**: Planning
- **Phase 2 (Weeks 4-6)**: Research
- **Phase 3 (Weeks 7-10)**: Initial design and component selection.
- **Phase 4 (Weeks 11-16)**: Board layout, prototyping, and testing.
- **Phase 5 (Weeks 17-20)**: SoC deployment infrastructure and Ada integration.
- **Phase 6 (Weeks 21-25)**: Final documentation, demonstration, and testing.
- **Phase 7 (Weeks 26-33)**: Software Infrastructure (SoC builder: automatic assembly of cores and devices/modules)
- **Phase 8 (Weeks 34-40)**: Design of two extension boards, each with its own Ada firmware

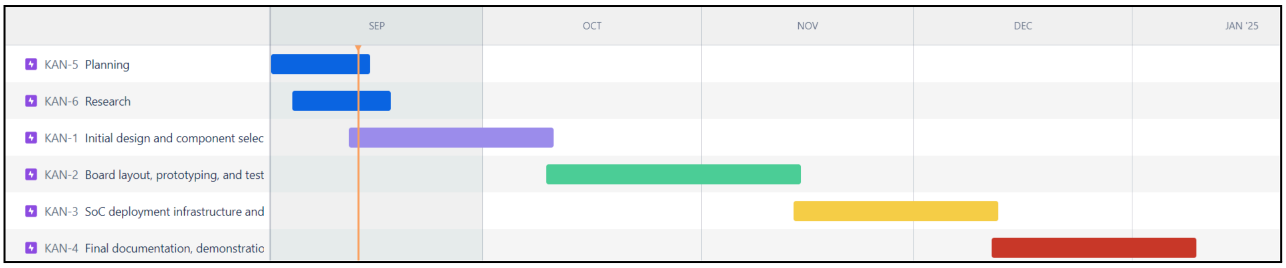

Figure 6 : Gantt Chart for first semester

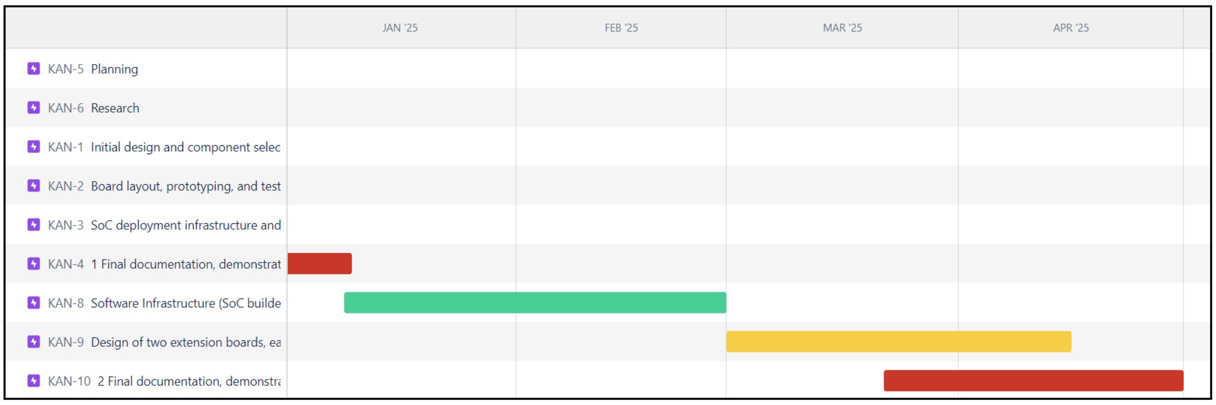

Figure 7 : Gantt Chart for second semester

Personal/Professional Expectations

This project provides an opportunity to deepen our knowledge of FPGA design, SoC architecture, and **Ada-based embedded systems development**. Professionally, we aim to enhance our skills in hardware-software co-design, we hope to develop stronger teamwork and project management skills through collaboration on an advanced real-world system.

Project Implementation Phase

The project will follow **SCRUM methodology**, involving:

- **Sprint Planning**: Outline goals for each cycle.
- **Daily SCRUM Meetings**: To discuss progress and tackle blockers.
- **Sprint Review**: End-of-sprint reviews to adjust the plan.

The first SCRUM will focus on setting up the **board layout**, **component selection**, and establishing a clear architecture.

Specifications and Formal Requirements

- **High IO availability** with flexible routing.
- **Component integration** of HyperRAM, DDR4 memory, and Artix-7 FPGA.
- **Support for open-source tools** like Yosys for bitstream creation.
- **Modular design** for extension boards.
- **Ada's package manager** integration for SoC deployment.