Digital ASIC Fabrication

Design Document

sdmay25-28

Client & Faculty Advisor: Dr. Henry Duwe

Camden Fergen

John Huaracha

Nicholas Lynch

Calvin Smith

Levi Wenck

sdmay25-28@iastate.edu

sdmay25-28.sd.ece.iastate.edu

Revised: December 7th 2024

Version 1.3

Executive Summary

There are not many ways for students to experience, learn, and participate in Digital ASIC design. Thankfully, due to the introduction of open-source tools and designs such as Caravel Harness and OpenLANE, ASIC design is achievable for undergraduate students. Our project will be leveraging these tools to make a digital ASIC of our own. We aim to create a RISC-V processor that supports customizable instructions programmed by the user. We hope our project can help students learn about Computer Hardware and Digital IC design by providing a hands-on digital IC that students can experiment with and learn from. We hope our project can also serve as a jumping off point for new students who may want to try digital IC design.

To complete our project, we are utilizing the open-source tools and designs from Efabless. We are using a CGRA integrated into a RISC-V core to extend the RISC-V ISA. There have been a handful of decisions made for our design: We have chosen to use the PicoRV32 as our RISC-V core, DFFRAM for On-Chip Memory, and have completed a high-level design for our project. We have put time into learning all of the open-source tools provided by Efabless to ensure we can finish our project by the end of May.

We are currently working on finishing the CyGRA and other modules, this will make sure that we are on track to complete our design. The goal for our CGRA is to accelerate custom instructions that use Taylor expansions. This will help us create custom instructions that manage trigonometric functions, exponentials, logarithms, and more. After all modules are completed, we will move on to integrating them into a top-level design and begin testing. After we fully test and confirm functionality, we will set our project to be fabricated in April.

Learning Summary

Development Standards & Practices Used

EEE 1754-1994: IEEE Standard for a 32-bit Microprocessor Architecture

IEEE 1364-2001: IEEE Standard Verilog Hardware Description Language

IEEE 1364.1-2002: IEEE Standard for Verilog Register Transfer Level Synthesis

Summary of Requirements

- Project must be compatible with the Efabless process
- Entire project must be open source
- Function as a RISC-V processor when provided with RISC-V instructions
- Support custom instructions defined by the user
- Stores and runs programs provided by the user
- HDL used is Verilog

Applicable Courses from Iowa State University Curriculum

- CPRE 2810 — Digital Logic
- CPRE 2880 — Embedded Systems
- CPRE 3810 — Computer Organization and Assembly Level Programming
- CPRE 4870 — Hardware Design for Machine Learning
- CPRE 4880 — Embedded Systems Design
- EE/CPRE 3300 — Integrated Electronics
- EE/CPRE 4650 — Digital VLSI Design

New Skills/Knowledge acquired that was not taught in courses

Skills:

- ASIC Chip Design
- Chip Fabrication
- How to use open-source tools

Tools:

- Caravel Harness
- CocoTB
- Magic DRC
- Netgen LVS
- OpenROAD
- Vivado

Table of Contents

Introduction 8

Problem Statement 8

Intended Users 9

Chip Forge Club Member 9

Hardware Students 9

Professors 9

Professionals 9

Requirements, Constraints, And Standards 9

Requirements & Constraints 9

Engineering Standards 10

Project Plan 11

Project Management/Tracking Procedures 11

Task Decomposition 11

Project Proposed Milestones, Metrics, and Evaluation Criteria 13

Project Timeline/Schedule 14

Risks and Risk Management/Mitigation 16

Personnel Effort Requirements 17

Other Resource Requirements 17

Design 18

Design Context 18

Broader Context 18

Prior Work/Solutions 19

Technical Complexity 21

Design Exploration 21

Design Decisions 21

Ideation 22

Decision-Making and Trade-Off 22

Proposed Design 24

Overview 24
Detailed Design and Visuals 25
Functionality 29
Areas of Concern and Development 29
Technology Considerations 30
Design Analysis 30
Testing 30
Unit Testing 30
Interface Testing 30
Integration Testing 31
System Testing 31
Regression Testing 31
Acceptance Testing 31
Results 31
Implementation 32
Professional Responsibility 34
Areas of Professional Responsibility/Codes of Ethics 34
Four Principles 36
Virtues 37
Closing Material 39
Conclusion 39
References 40
Appendices 40
Team 41
Team Members 41
Required Skills Sets 41
Skills Set Covered by the Team 41
Project Management Style Adopted by the Team 41
Initial Project Management Roles 42

Team contract 42

List of Symbols and Definitions

Efabless - Open-source fabrication company that will fabricate our chip, and provide us with design resources/tools

Caravel Harness - Provided wrapper around our design which includes an SoC

ASIC - Application-Specific Integrated Circuit

RISC-V – Open-source ISA

PicoRV32 - A Size-Optimized RISC-V CPU

Chip Forge – ISU Club focused on developing and bringing up ASICs

CGRA – Coarse Grained Reconfigurable Architecture, a reconfigurable architecture that operates on coarser granularity than traditional reconfigurable architectures such as FPGA

SPI - Serial Peripheral Interface

PCB - Printed Circuit Board

FPGA - Field Programmable Gate Array, is a reprogrammable integrated circuit

Verilog HDL - Verilog Hardware Description Language

SoC - System on chip

SkyWater 130nm - Fabrication process used by Efabless

User Area - Our design space within the Caravel Harness

Management Area - Part of the Caravel Harness that includes management utilities, SoC, and logic analyzer probes

OpenLane - The collection of open-sourced tools provided by Efabless

Introduction

Problem Statement

Processors are limited by their defined instruction sets. If you want to perform or test an operation that is not supported in the instruction set, you will have to redesign the processor and refabricate it to support that operation (expensive and time-consuming), approximate the operation by using multiple instructions (slow and/or inaccurate), or avoid doing that operation entirely. These options are not acceptable in many situations. There are no current processors that are suited to learning processor design due to the lack of low-level intractability and modification of most processors, forcing students and professors to utilize other options. To solve this, we aim to design a RISC-V processor that supports the use of a custom instruction defined by the user, which would allow a user to directly program the processor to support any desired instruction. We aim for this instruction to be as customizable as possible, allowing the user high flexibility when programming their instruction.

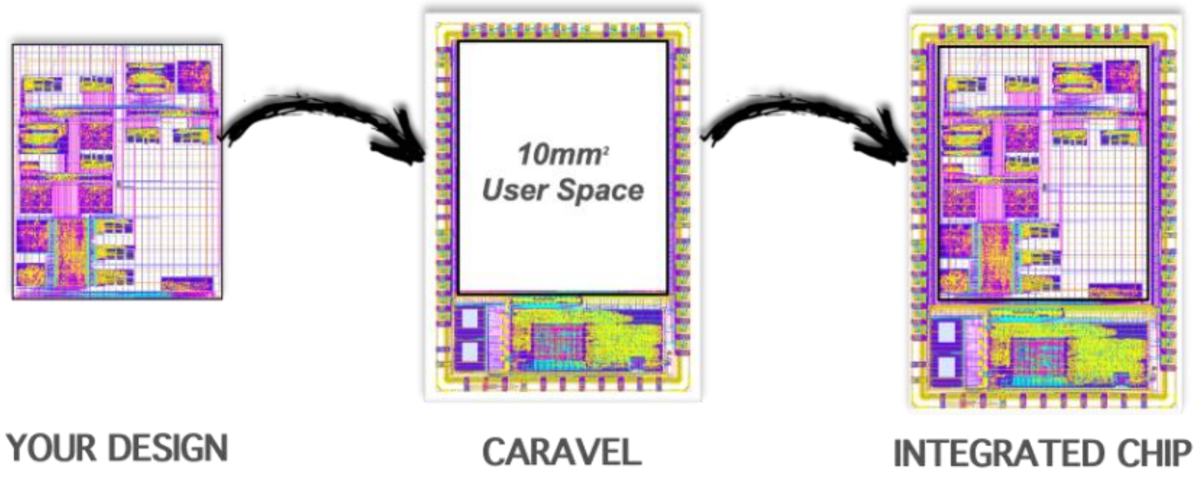


Figure 1: Caravel Chip Design

To design our ASIC, we will use Efabless's Caravel and OpenROAD. Caravel is a harness we can insert our project into. Caravel provides a management area controlled by a VexRISCV processor and includes logic analyzers, interrupt pins, a wishbone bus, a clock, and a reset. We will use these to aid in our design which will be placed in the user space. OpenROAD is a compilation of tools that generates a layout from Verilog files and performs DRC, LVS, and STA tests.

Intended Users

Chip Forge Club Member

Chip Forge is a student organization at Iowa State University dedicated to designing analog and digital ASICs. Each member has an interest in designing, testing, and/or fabricating ASICs. The club uses Caravel, so students need resources to help learn about it. Our project will provide students with an interactable ASIC to learn about the Caravel chip and serve as a potential jumping-off point for their projects.

Hardware Students

Hardware students encompass all students learning about digital hardware design in classes like CPRE 2810, CPRE 3810, or CPRE/EE 4650. These students need ways to learn about ISAs and test new instructions for a processor. The ways students learn about these topics is through software and FPGAs. Our project will provide a way for students to learn these things on a physical processor, which will aid in the education of many hardware students. Our project will also aid future students who use the Efabless process and Caravel chip in future senior design projects.

Professors

Professors are interested in instructing their students and need new teaching methods. Our project provides an interactive way to teach students about RISC-V, ISAs, and microprocessors. Our project will expand professors' teaching options and allow them to instruct their students more effectively.

Professionals

Professionals are interested in accelerating computation and finding new ways to solve computational problems. Our modifiable instruction will allow users to circumvent specific problems that may be difficult to solve using traditional processors due to instructions not being natively supported on hardware. Our modifiable instruction will allow for hardware acceleration of many tasks, reducing the cycles needed to perform those tasks, therefore cutting down the time spent executing those tasks.

Requirements, Constraints, And Standards

Requirements & Constraints

Functional requirements:

- Function as a RISC-V processor when provided with RISC-V instructions.

- Support custom instructions defined by the user.
- Custom instructions should only be executed when called (should not execute custom instructions when provided standard RISC-V instructions).
- Stores and runs programs provided by the user.

Technical requirements:

- HDL used is Verilog
- Custom instruction execution should not slow down the processer.
- Programming a new instruction should take minimal time.
- Max clock frequency of 40 MHz (**constraint**).
- The microcontroller is programmed using C.
- Design should pass LVS and DRC test before sending off to be fabricated.
- The RISC-V processor used must be open-source.
- Tapeout in the 130 nm skywater process

User experiential requirements:

- Product should be user-friendly to program custom instructions and load in programs.
- Product should provide a wide range of settings that cater to different user experience levels.
- It should be easy to test custom instructions and programs.
- Custom instruction assembly code should be structured and loaded like a standard RISC-V instruction.

Physical:

- Must function at room temperature (≈20°C).
- User project must be 3mm x 3.6mm to fit in user project area (**constraint**).
- Must use I/O pins provided by project wrapper (**constraint**).

Engineering Standards

Engineering standards are important to adhere to when designing products. Standards ensure that your product is consistent with the industry, allowing easier use for users and others in the industry that may work on your product later. For our project, we will be using standards laid out by IEEE.

<u>EEE 1754-1994: IEEE Standard for a 32-bit Microprocessor Architecture</u>

Since we are designing a 32-bit Microprocessor, it is important for our project to adhere to the pre-established standards from IEEE. We be careful to ensure that we adhere to this standard when adding our custom function unit.

<u>IEEE 1364-2001: IEEE Standard Verilog Hardware Description Language</u>

The Efabless process requires the use of Verilog for our design. Using this, we will ensure our Verilog code adheres to the industry standard.

<u>IEEE 1364.1-2002: IEEE Standard for Verilog Register Transfer Level Synthesis</u>

Our project will use RTL synthesis to translate our Verilog code to a hardened design. Using this, we can write code that best works with RTL synthesis to ensure correctness and efficiency.

<u>Wishbone Bus</u>

Our project will use the WISHBONE protocol as implemented by Efabless to communicate between devices on chip and in the management area, the Efabless process has this protocol implemented as a non-option, with slight variations from the OpenCores WISHBONE.

<u>SPI Protocol</u>

The SPI Protocol de facto standard is a serial communication bus that was developed by Motorola in the 1980s with a master-slave configuration that is commonly used in SD cards, it consists of four logic signals; CS, SCLK, MOSI, MISO. For our project the SPI Protocol will be used to communicate between off-chip memory and on-chip memory (the slaves) via a memory interface which acts as the master.

Project Plan

Project Management/Tracking Procedures

Our team will be using an agile management style for project planning. This allows our team to have structured goals with clear deadlines and milestones to reach, while also ensuring our team is provided with the flexibility needed to accommodate any unexpected difficulties that may arise during development.

Our team will track the progress through communication on Microsoft Teams and Discord as well as shared files in the SharePoint associated with the Teams. Additionally, we will be using GitLab for version control of code base and tracking any specific issues found in the code.

Task Decomposition

Our project is split up into the following tasks which will be completed in sequential order:

1. Project Prep
    a. Setup virtual machines to complete design work on
    b. Setup Gitlab repos to hold source code and GitLab modules
    c. Become familiar with the Efabless tools
        i. OpenLane
        ii. Caravel
    d. Successfully harden and verify a test design
2. November Chip Design
    a. Decide on a design/component to place onto to chip framework
    b. Harden chosen design
    c. Pass precheck and verify functionality
    d. Work with dec24-12 to integrate our design into their chip
3. Project Design
    a. High level chip design
        i. Basic overview
        ii. Determine how RISC-V core will interface with memory
            1. Design cache system to interface with off chip memory
        iii. CyGRA integration
        iv. Interface with management area
    b. RISC-V core
        i. Determine available open IP designs
        ii. Choose best open IP for our project
        iii. Test harden the softcore to ensure compatibility
    c. Accelerator design (CyGRA)
        i. Determine acceleration use case
        ii. Design and verify hard coded accelerator
        iii. Backport work on hard coded accelerator to CGRA
    d. RISC-V ISA extension
        i. Add basic instruction for testing
        ii. Integrate CyGRA into a custom instruction
    e. PICO test plan
        i. Create toolflow for testing
            1. Behavorial
            2. C code
        ii. FPGA testing flow
    f. Management area interface
        i. Determine how interrupts are supplied to processor
        ii. Determine how management area will interface with on chip memory

4. Integration
    a. Combine memory with RISC-V core
    b. Connect CyGRA to RISC-V core
    c. Test off chip memory
    d. Integrate wishbone with management core
5. Testing/Verification
    a. Ensure full functionality of RISC-V core
    b. Ensure implemented instruction works as expected
    c. Ensure memory interface is working correctly
6. Documentation
    a. Design documentation
    b. Design presentation
    c. Website design
    d. Bring up planning

Project Proposed Milestones, Metrics, and Evaluation Criteria

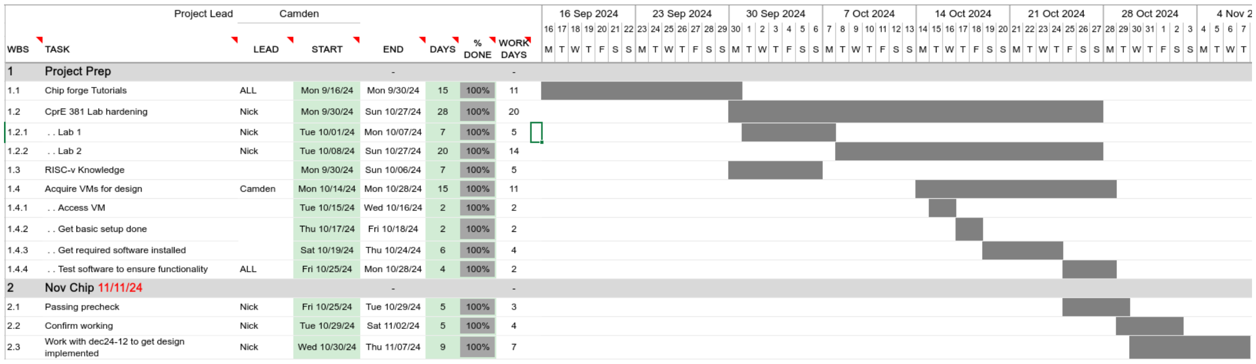
Our project's milestones are closely related to the main task sections listed above. Each of the milestones will be measured using the following metrics:

- Milestone 1: Complete tooling setup
    - Each member of the team is able to fully harden a design and complete GL simulation on an example project.
    - Each member can create a Verilog module and complete the steps above
    - Each member is aware of how to use the virtual machines to harden a design
- Milestone 2: Small chip design
    - Using a previously designed datapath from a CPRE class, harden and verify the design
    - Work with dec24-12 team to integrate our simple design into their framework
- Milestone 3: High level design and RISC-V core
    - Determine which RISC-V softcore best fits our use case
    - Ensure the chosen RISC-V core works in the Efabless tooling
    - Design high level of chip with included RISC-V core
- Milestone 4: Accelerator/CyGRA design
    - Determine the specific application to accelerate
    - Develop simple Verilog module to accelerate use case
    - Backport work to a CGRA design
- Milestone 5: System Integration
    - Combine the CyGRA and the RISC-V core

    - Include on chip memory
    - Wishbone bus integration to management core
- Milestone 6: Overall Design Testing
    - Ensure the RISC-V core can interface with both on and off chip memory
    - Ensure the management core can write to the memory and reset the RISC-V core
    - Ensure the custom instruction for the CyGRA works as intended
- Milestone 7: Submit Design to Efabless
    - Place project in public efabless repository
    - Finish all hardened and precheck and submit to efabless by April Deadline
- Milestone 8: Complete Documentation
    - Complete all project documentation ensuring readability
    - Finish any bring up documentation for the chip

Project Timeline/Schedule

Our project is broken up into 6 major parts; Project prep and setup, November chip deadline, Project design, Integration, Testing, and Documentation. One of the first deadlines is the November chip deadline (Figure 1). This included a full test of our team's knowledge of the efabless tooling as well as our ability to work with another design team to integrate our design into their multi-design framework.

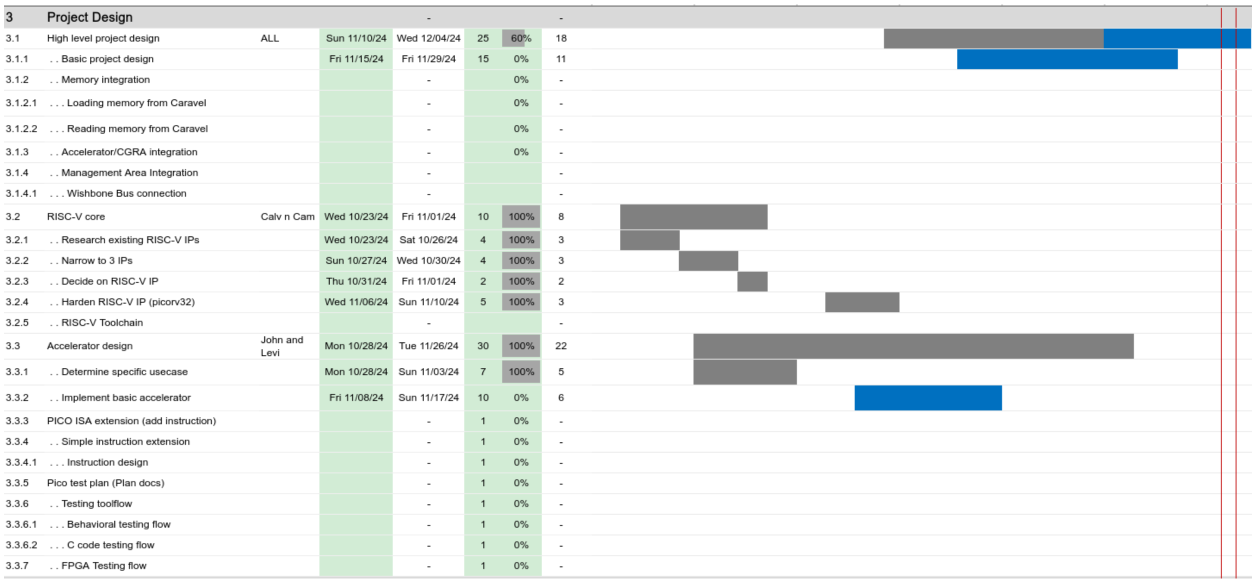


Project Lead Camden

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	Project Prep			-			-
1.1	Chip forge Tutorials	ALL	Mon 9/16/24	Mon 9/30/24	15	100%	11
1.2	CprE 381 Lab hardening	Nick	Mon 9/30/24	Sun 10/27/24	28	100%	20
1.2.1	. . Lab 1	Nick	Tue 10/01/24	Mon 10/07/24	7	100%	5
1.2.2	. . Lab 2	Nick	Tue 10/08/24	Sun 10/27/24	20	100%	14
1.3	RISC-v Knowledge		Mon 9/30/24	Sun 10/06/24	7	100%	5
1.4	Acquire VMs for design	Camden	Mon 10/14/24	Mon 10/28/24	15	100%	11
1.4.1	. . Access VM		Tue 10/15/24	Wed 10/16/24	2	100%	2
1.4.2	. . Get basic setup done		Thu 10/17/24	Fri 10/18/24	2	100%	2
1.4.3	. . Get required software installed		Sat 10/19/24	Thu 10/24/24	6	100%	4
1.4.4	. . Test software to ensure functionality	ALL	Fri 10/25/24	Mon 10/28/24	4	100%	2
2	Nov Chip 11/11/24			-			-
2.1	Passing precheck	Nick	Fri 10/25/24	Tue 10/29/24	5	100%	3
2.2	Confirm working	Nick	Tue 10/29/24	Sat 11/02/24	5	100%	4
2.3	Work with dec24-12 to get design implemented	Nick	Wed 10/30/24	Thu 11/07/24	9	100%	7

Figure 2: Project Prep and November Chip

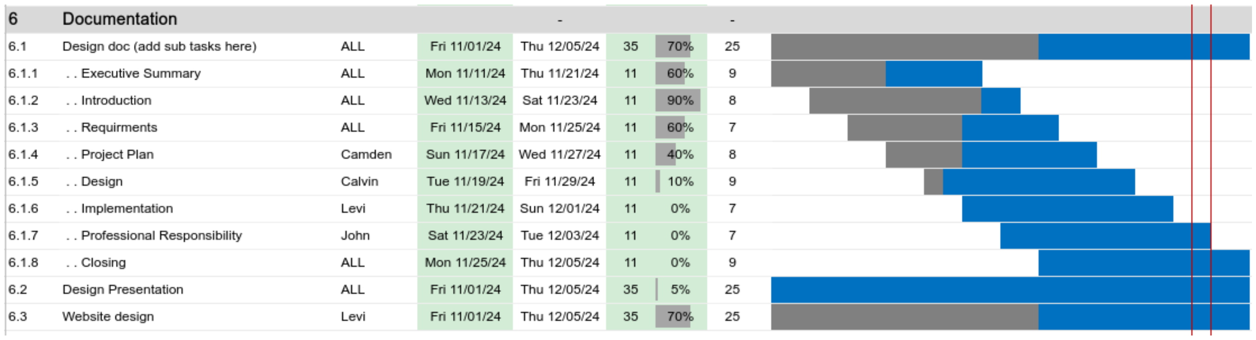
Our next deadline is focused on the full design and submission to efabless. This must be completed before April 11th, 2025, to ensure that we have time to fix any issues as they arise.



3	Project Design			-			-
3.1	High level project design	ALL	Sun 11/10/24	Wed 12/04/24	25	60%	18
3.1.1	. . Basic project design		Fri 11/15/24	Fri 11/29/24	15	0%	11
3.1.2	. . Memory integration			-		0%	-
3.1.2.1	. . . Loading memory from Caravel			-		0%	-
3.1.2.2	. . . Reading memory from Caravel			-		0%	-
3.1.3	. . Accelerator/CGRA integration			-		0%	-
3.1.4	. . Management Area Integration			-			-
3.1.4.1	. . . Wishbone Bus connection			-			-
3.2	RISC-V core	Calv n Cam	Wed 10/23/24	Fri 11/01/24	10	100%	8
3.2.1	. . Research existing RISC-V IPs		Wed 10/23/24	Sat 10/26/24	4	100%	3
3.2.2	. . Narrow to 3 IPs		Sun 10/27/24	Wed 10/30/24	4	100%	3
3.2.3	. . Decide on RISC-V IP		Thu 10/31/24	Fri 11/01/24	2	100%	2
3.2.4	. . Harden RISC-V IP (picorv32)		Wed 11/06/24	Sun 11/10/24	5	100%	3
3.2.5	. . RISC-V Toolchain			-			-
3.3	Accelerator design	John and Levi	Mon 10/28/24	Tue 11/26/24	30	100%	22
3.3.1	. . Determine specific usecase		Mon 10/28/24	Sun 11/03/24	7	100%	5
3.3.2	. . Implement basic accelerator		Fri 11/08/24	Sun 11/17/24	10	0%	6
3.3.3	PICO ISA extension (add instruction)			-	1	0%	-
3.3.4	. . Simple instruction extension			-	1	0%	-
3.3.4.1	. . . Instruction design			-	1	0%	-
3.3.5	Pico test plan (Plan docs)			-	1	0%	-
3.3.6	. . Testing toolflow			-	1	0%	-
3.3.6.1	. . . Behavioral testing flow			-	1	0%	-
3.3.6.2	. . . C code testing flow			-	1	0%	-
3.3.7	. . FPGA Testing flow			-	1	0%	-

Figure 3: Project Design

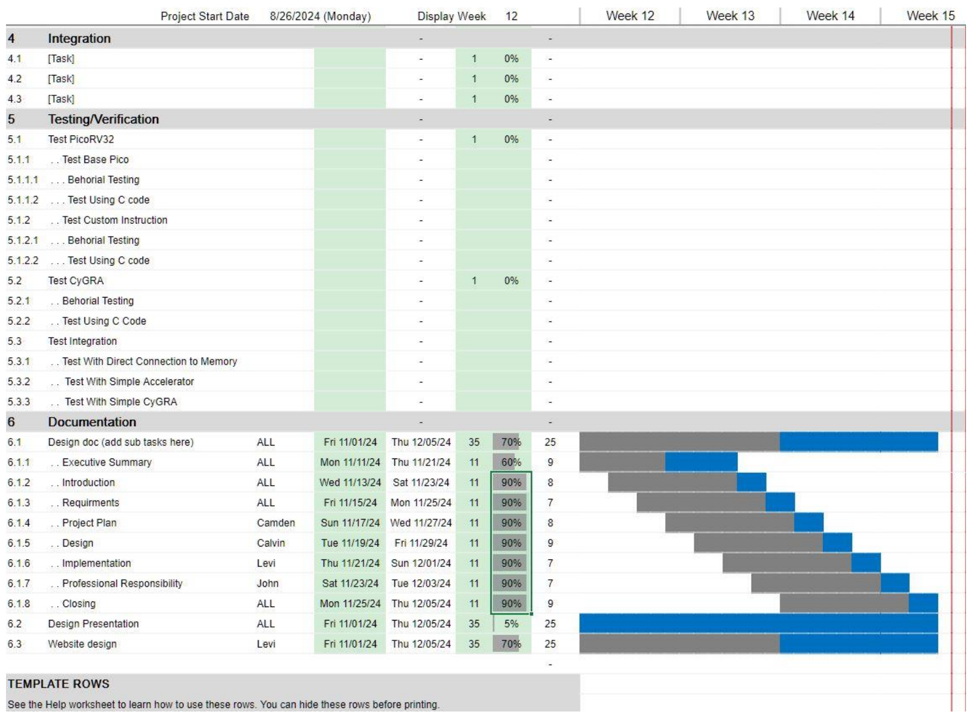
As we progress through the project, more items will be filled out and deadlines set as needed. Our gantt chart is mostly focused on the immediate future with less detailed objectives filling out the rest to give a rough outline of what still needs to be completed as seen in figure 2 and figure 3.



6	Documentation			-			-
6.1	Design doc (add sub tasks here)	ALL	Fri 11/01/24	Thu 12/05/24	35	70%	25
6.1.1	. . Executive Summary	ALL	Mon 11/11/24	Thu 11/21/24	11	60%	9
6.1.2	. . Introduction	ALL	Wed 11/13/24	Sat 11/23/24	11	90%	8
6.1.3	. . Requirments	ALL	Fri 11/15/24	Mon 11/25/24	11	60%	7
6.1.4	. . Project Plan	Camden	Sun 11/17/24	Wed 11/27/24	11	40%	8
6.1.5	. . Design	Calvin	Tue 11/19/24	Fri 11/29/24	11	10%	9
6.1.6	. . Implementation	Levi	Thu 11/21/24	Sun 12/01/24	11	0%	7
6.1.7	. . Professional Responsibility	John	Sat 11/23/24	Tue 12/03/24	11	0%	7
6.1.8	. . Closing	ALL	Mon 11/25/24	Thu 12/05/24	11	0%	9
6.2	Design Presentation	ALL	Fri 11/01/24	Thu 12/05/24	35	5%	25
6.3	Website design	Levi	Fri 11/01/24	Thu 12/05/24	35	70%	25

Figure 4: Documentation

Figure 4 represents some of the future objectives that will need to be completed. They are left dateless since they require previous objectives to be completed first.



Project Start Date 8/26/2024 (Monday) Display Week 12

4	**Integration**			-			-
4.1	[Task]			-	1	0%	-
4.2	[Task]			-	1	0%	-
4.3	[Task]			-	1	0%	-
5	**Testing/Verification**			-			-
5.1	Test PicoRV32			-	1	0%	-
5.1.1	. . Test Base Pico			-			-
5.1.1.1	. . . Behorial Testing			-			-
5.1.1.2	. . . Test Using C code			-			-
5.1.2	. . Test Custom Instruction			-			-
5.1.2.1	. . . Behorial Testing			-			-
5.1.2.2	. . . Test Using C code			-			-
5.2	Test CyGRA			-	1	0%	-
5.2.1	. . Behorial Testing			-			-
5.2.2	. . Test Using C Code			-			-
5.3	Test Integration			-			-
5.3.1	. . Test With Direct Connection to Memory			-			-
5.3.2	. . Test With Simple Accelerator			-			-
5.3.3	. . Test With Simple CyGRA			-			-
6	**Documentation**			-			-
6.1	Design doc (add sub tasks here)	ALL	Fri 11/01/24	Thu 12/05/24	35	70%	25
6.1.1	. . Executive Summary	ALL	Mon 11/11/24	Thu 11/21/24	11	60%	9
6.1.2	. . Introduction	ALL	Wed 11/13/24	Sat 11/23/24	11	90%	8
6.1.3	. . Requirments	ALL	Fri 11/15/24	Mon 11/25/24	11	90%	7
6.1.4	. . Project Plan	Camden	Sun 11/17/24	Wed 11/27/24	11	90%	8
6.1.5	. . Design	Calvin	Tue 11/19/24	Fri 11/29/24	11	90%	9
6.1.6	. . Implementation	Levi	Thu 11/21/24	Sun 12/01/24	11	90%	7
6.1.7	. . Professional Responsibility	John	Sat 11/23/24	Tue 12/03/24	11	90%	7
6.1.8	. . Closing	ALL	Mon 11/25/24	Thu 12/05/24	11	90%	9
6.2	Design Presentation	ALL	Fri 11/01/24	Thu 12/05/24	35	5%	25
6.3	Website design	Levi	Fri 11/01/24	Thu 12/05/24	35	70%	25
							-

TEMPLATE ROWS

See the Help worksheet to learn how to use these rows. You can hide these rows before printing.

Figure 5: Integration, Testing/Verification, and Documentation

Risks and Risk Management/Mitigation

Due to our open-ended project, there are a few risks that are quite different from other groups.

One major risk that we may face is a project that is unfeasible due to the amount of time. We are highly unlikely to run into this problem due to our guidance from Dr. Duwe, as he has given us advice about how to plan our work and when we propose something too difficult, we are advised to propose something more feasible. Though it is unlikely that we will run into this problem, if we were to run into this very late into our project, it would result in major consequences such as our project not having a complete design. To mitigate this risk, we have come up with a way to fall back on if we were to run into this problem.

Another major risk that we could face is time delays, due to us not being too familiar with Efabless or caravel before this project, there is a high chance that we will be set back with problems that were not expected. Though it would not result in major consequences, if we are consistently setback major problems could result later down the line which we would like to avoid. To mitigate the risk of it having a major impact on our project, we have planned to use an agile management style which would help us address these issues more effectively. Another way to mitigate the risk would be just making sure that we have clear communication within our group to make sure that we are moving at a good pace and not stuck somewhere for a long time.

Personnel Effort Requirements

Our list of tasks is a constantly developing and evolving entity that we cannot put expected work hours towards each task, especially since said numbers can be skewed by factors such as the number of sessions and length of said sessions that have varying levels of efficiency, instead we have a listed number of workdays and/or story points as found in the AGILE workflow.

Task	**Projected Hours**
Workflow Tools/Setup	40 Hours
November Chip Design	20 Hours
Project Design	100 Hours
Integration	60 Hours
Testing/Verification	40 Hours
Documentation	40 Hours

Table 1: Projected Hours

Other Resource Requirements

Tools and environments to work in that are beneficial to larger team productivity are immensely advantageous in this project, as our team has gotten in contact with ETG and currently created 1 VM (soon to be 2) that we can work collaboratively on remotely where our environment has identical variables and promotes easy sharing of data.

Design

Design Context

Broader Context

When we were designing and planning our project, we thought about accessibility to a niche subject that is filled with proprietary technology/IP as such we decided to use only Open-Source tools, so our final deliverable contributes as an educational piece that can help others break into hardware design.

List relevant considerations related to our project in each of the following areas:

Area	**Description**	**Example**
Public Health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., the solution is implemented in their communities)	Our product gives students hands-on experience with a real processor fabricated on an IC rather than software simulations and FPGAs. The programmable aspect of the project can help students test designs and learn Hardware design.
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Our project adds to the list of open-source designs that the global electrical and computer engineering community can use and expand upon.
Environment	What environmental impact might your project have? This can include indirect effects, such as deforestation or	Custom instruction can make some processes take less instructions and less energy, which can add up when done many times.

	unsustainable practices related to materials manufacture or procurement.	
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Our product is open source which lets anyone use it. This saves people from having to develop a design of their own, which saves them money.

Table 2: Broader Context

Prior Work/Solutions

Nios® V/g General purpose Processor from Intel

- Processor Overview: https://www.intel.com/content/www/us/en/docs/programmable/683632/24-3/processor-87132.html
  - 32-bit RISC-V processor
  - FPGA implementation of processor
  - Come with Quartus® Prime Pro Edition
- Custom Instruction Overview: https://www.intel.com/content/www/us/en/docs/programmable/773194/current/processor-custom-instruction-overview.html
  - Processors support non-branching custom instructions
  - Selected by a mux choosing between the ALU and Custom Instruction Unit during the execution state of the pipeline.
- Benefits and Drawback compared to our design
  - Benefits:
    - Supports 32 Custom Logic Blocks
    - Has an Integer Multiplication and Division Unit
    - Has a floating-point unit
    - Pipelined
    - Faster frequency max on popular FPGAs

- Drawbacks:
  - FPGA based
    - Require user to have an FPGA
    - Larger Area
  - Can not fabricate as an IC
  - Cannot re-program custom instruction during run time
  - Not free (requires Quartus® Prime Pro Edition)

<u>Arm Custom Instructions</u>

- Paper: https://armkeil.blob.core.windows.net/developer/Files/pdf/white-paper/arm-custom-instructions-without-fragmentation-whitepaper.pdf
  - Available with Cortex –M33, Cortex-M55, and Cortex-M85
  - Arm Architecture
  - Programable Instruction
  - Used in the Execution stage of the pipeline
- Benefits and drawbacks compared to our solution
  - Benefits
    - Available on Arm processors
      - More features
      - Faster
    - Custom Instruction solution more intricate
      - Can pipeline custom instruction
  - Drawbacks
    - Neither Arm nor Arm Custom Instructions is open source
    - Complicated to use

Sources with citations:

[1] "4. Nios® V/g Processor," Intel, 2024. https://www.intel.com/content/www/us/en/docs/programmable/683632/24-3/processor-87132.html (accessed Dec. 08, 2024).

[2] "1. Nios® V Processor Custom Instruction Overview," Intel, 2023. https://www.intel.com/content/www/us/en/docs/programmable/773194/current/processor-custom-instruction-overview.html (accessed Dec. 08, 2024).

[3] J. Yiu, "Innovate by Customized Instructions, but Without Fragmenting the Ecosystem," 2021. Accessed: Dec. 08, 2024. [Online]. Available:

https://armkeil.blob.core.windows.net/developer/Files/pdf/white-paper/arm-custom-instructions-without-fragmentation-whitepaper.pdf

Technical Complexity

Our project has multiple complex components that need to work together. Finishing the project will require the following tasks.

- Creating a Wishbone Slave interface to communicate write data to the User Project Area.
- Creating a CGRA that can be reconfigured and used to implement custom instructions.
- Creating a memory interface that can manage memory between a cache and external memory.
- Creating an SPI Main interface that can communicate with external memory.
- Integrating all components into a final design and having everything work with the PicoRV32 to allow users to create custom instructions and run programs.

Design Exploration

Design Decisions

For our project, we needed to make several design decisions listed below.

- Which processor ISA will we use
    - The ISA determines what instructions our processor will support and how those instructions are formatted in memory. The ISA could affect how we can program and call a custom instruction.
- Which open-source implementation of the ISA will we use
    - For this project, we will be using an open-source implementation of an ISA to focus on implementing the custom instruction rather than the design. We want to choose a processor design that allows the easy implementation of custom units within the processor. We also need to consider the size the processor will take, the speed at which it can execute instructions, and how memory is read.
- What design will we use to implement our custom instruction
    - We need to decide on a digital circuit that can have instructions programmed into it. The circuit needs to be small, fast, and flexible to ensure the unit functions as expected.

One important consideration that we had to keep in mind when making these above decisions was our initial constraints through Efabless. Each of the selections in these categories must be compatible with each other and supported through Efabless, this was the primary reason we weren't considering the MIPS ISA (even though that's what we have the most experience with through ISU), even though in the end we would've likely chosen the RISC-V ISA from its broader support in industry.

Ideation

When looking for an open-source processor design, we had some suggestions from our advisor and found options online. Below are the designs we found and considered.

- Rocket Chip
    - RISC-V 64 bit
    - Feature Rich
    - Not optimized for size
    - Written in Chisel
- Neorv32
    - RISC-V 32 bit
    - Designed as a Microcontroller
    - Not optimized for size
    - Written in VHDL
- Vex RISC-V
    - RISC-V 32 bit
    - High customizability
    - Not optimized for size
    - Written in SpinalHDL
- CVA6
    - RISC-V 64 bit
    - Support UNIX-like operating systems
    - Not optimized for size
    - Written in System Verilog
- PicoRV32
    - RISC-V 32 bit
    - Wishbone interface
    - High clock frequency
    - Size optimized
    - Written in Verilog

Decision-Making and Trade-Off

When deciding between the different RISC-V soft-core designs we came up with a few criteria to rank the various designs we found. The criteria are as follows:

- Instruction Set

  - This criterion was used to separate the processors based on which instruction architecture they supported, either 32 or 64 bit
  - It was also further used to specify and determine what processor supported the most of a given instruction set, i.e. RV32I vs RV32IM
- Written Language
  - This criterion was used as for the Efabless process, we needed to submit our design/create our design in Verilog
- Features
  - This criterion was used to specify what type of extra features each processor included in its base package. Things like the ability to run Linux or if it contained a wishbone interface were added here
- Size Optimization
  - This criterion was based on if the GitHub repository for each processor specifically mentioned its size/space optimization. This was important since the user area on the Efabless caravel is limited to 2.92 mm x 3.52mm (10mm^2), and we require space not only for the base design of the CPU but also for our additional accelerator.

Based on the criteria listed above and considering the features listed in the previous section. We chose PicoRV32 because it is written in Verilog (avoiding having to generate Verilog from something else, but simply doing it directly), contains a wishbone interface, and is optimized to be smaller. The size of our implementation is a major concern as the smaller our base design is, the more room is left for our accelerator and adding additional things like on-chip memory.

Proposed Design

Overview

Our high-level design integrated into the Caravel harness is shown below in figure 3.

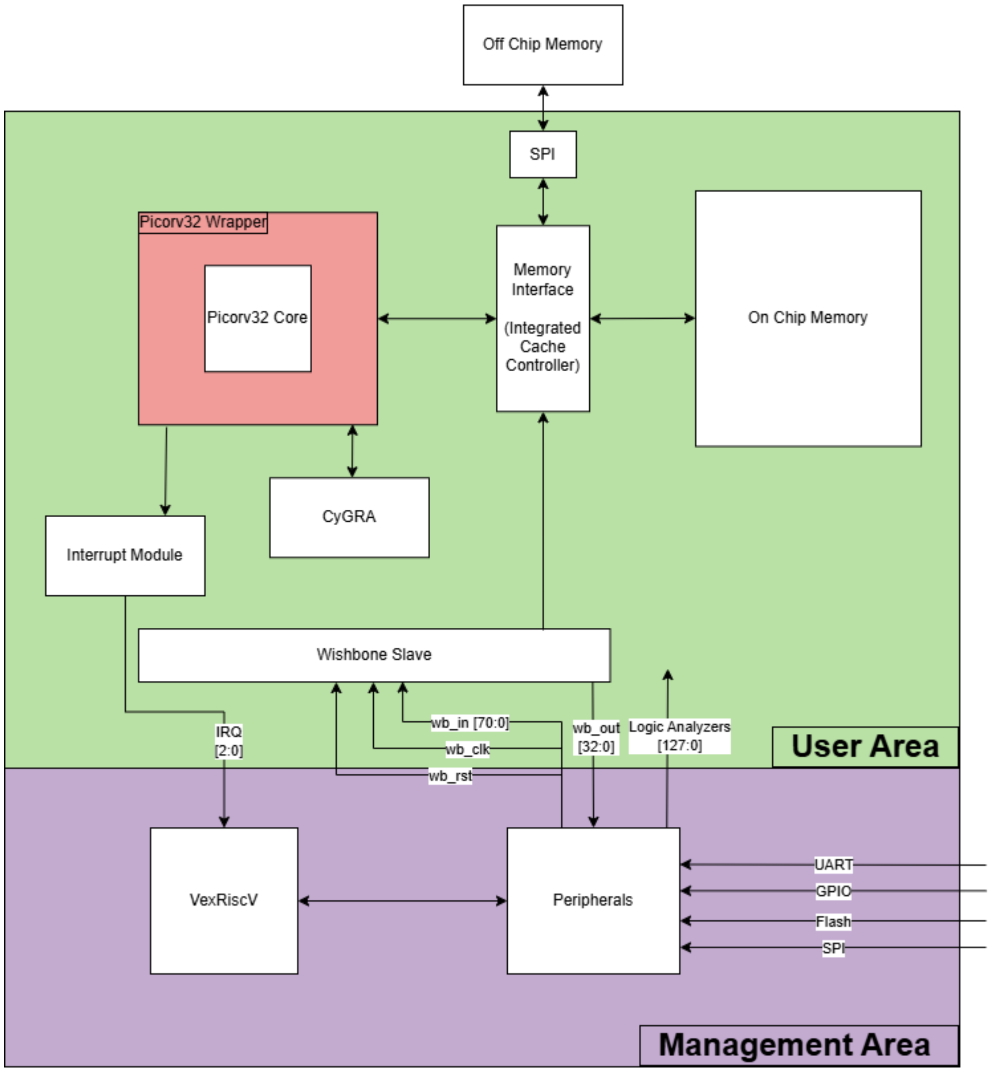


Figure 6: High-Level Design

The high-level design consists of two main areas, the Management Area, which is included with the Caravel harness, and the User Area, where our modules will be placed. The User and Management area communicate through the following buses, Wishbone, which provides clock, reset, and buses from communicating between the User and Management Area, Logic Analyzers, which allows us to probe certain parts of the project for testing, and Interrupt signals, which allows the User Project area to run code on the VexRISCV when needed. The Management Area has flash inputs, which allows programs to be loaded onto the VexRISCV. The Management and User Areas also share 34 GPIO ports, which can be set to either an input or outputsdepending on what is needed.

Detailed Design and Visuals

PicoRV32

The PicoRV32 is a size optimized RISC-V 32-bit processor that we chose for our project. The PicoRV32 is not pipelined, meaning the processor can only run one stage at a time. A simplified datapath of the PicoRV32 with a CGRA is shown below in figure 4.

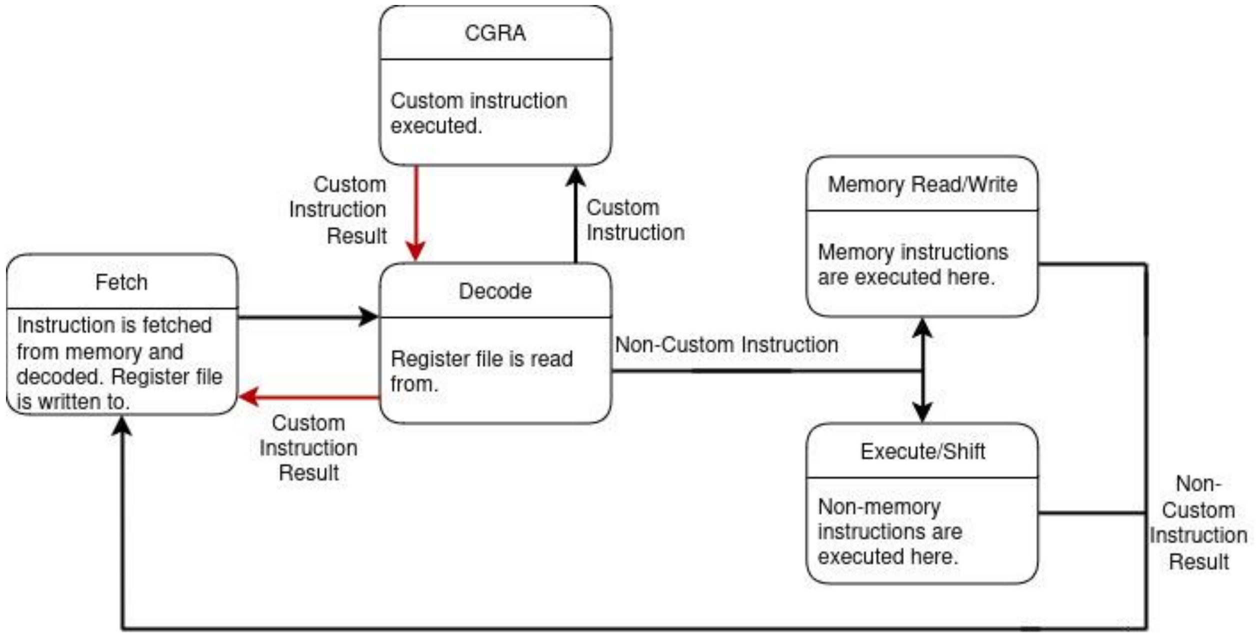

CGRA
Custom instruction executed.
Custom Instruction Result
Custom Instruction
Memory Read/Write
Memory instructions are executed here.
Fetch
Instruction is fetched from memory and decoded. Register file is written to.
Decode
Register file is read from.
Non-Custom Instruction
Custom Instruction Result
Execute/Shift
Non-memory instructions are executed here.
Non-Custom Instruction Result

Figure 7: PicoRV32 Datapath

The PicoRV32 can run all standard 32-bit RISC V instructions but also supports custom instruction through a co-processor interface which activates when an unknown instruction is decoded. This is where are programmable unit will interface with and be used to execute a custom instruction. Below describes the dataflow of the PicoRV32 with a CGRA.

The first stage of the processor is the Fetch stage which does three things.

1. Fetches the next instruction from memory.

2. Decodes the fetched memory and sets control signals.

3. Write back the result of the previous instruction to the register file if needed.

The next stage of the processor reads from the register file and can go to one of five stages.

1. Memory Read: if the decoded instruction reads from memory.

2. Memory Write: if the decoded instruction writes to memory.

3. Shift: if the decoded instruction is a shift instruction.

4. Execute: if the instruction is not a Memory Read, Memory Write, or Shift.

5. CGRA: if the instruction is a custom instruction.

Once the previous stage finishes, each stage will go to the Fetch Stage. The Memory Read, Write, Shift, and Execute stages directly go to the Fetch stage. But the CGRA stage goes to the Decode Stage, then to the Fetch stage. Below is a chart showing the CPI for non-custom instruction provided by the PicoRV32 readme and their runtimes based at 40 MHz (clock speed provided by Caravel).

Instruction	Jump and link	ALU reg + Immediate	ALU reg + reg	Branch (not taken)	Memory load	Memory store	Branch (taken)	Indirect jump	Shift operations
CPI	3	3	3	3	5	5	5	6	4-15
Runtime	75 ns	75 ns	75 ns	75 ns	125 ns	125 ns	125 ns	150 ns	100ns-375 ns

Note: The memory figures are assuming single cycle read/write memory which is unrealistic for our project. A cache hit may produce a CPI and runtime like above, but a cache hit would result in a high CPI and slow runtime due to SPI being only able to transfer one bit at a time.

We are aiming for the custom instruction to take 3-30 CPI depending on the instruction, meaning that the runtime would range from 75 ns to 750 ns.

The PicoRV32 will write and read memory to the Memory Interface, which is described later in this section. The PicoRV32 will also signal interruptions when certain actions need to be taken by the VexRISCV.

CyGRA

Below is a basic outline of the currently planned CyGRA design. We will have an instruction parse unit for basic decoding of whether it receives a configuration instruction or an operation instruction.

- Configuration instruction will set the configuration registers which are the select bits for the mux input, as well as what operation the ALU executes

- Operation instruction will simply pass the register data to every input, and set interface validity once the instruction is done.

Currently we plan to support signed and unsigned addition and multiplication operations for fixed point and integer data types.

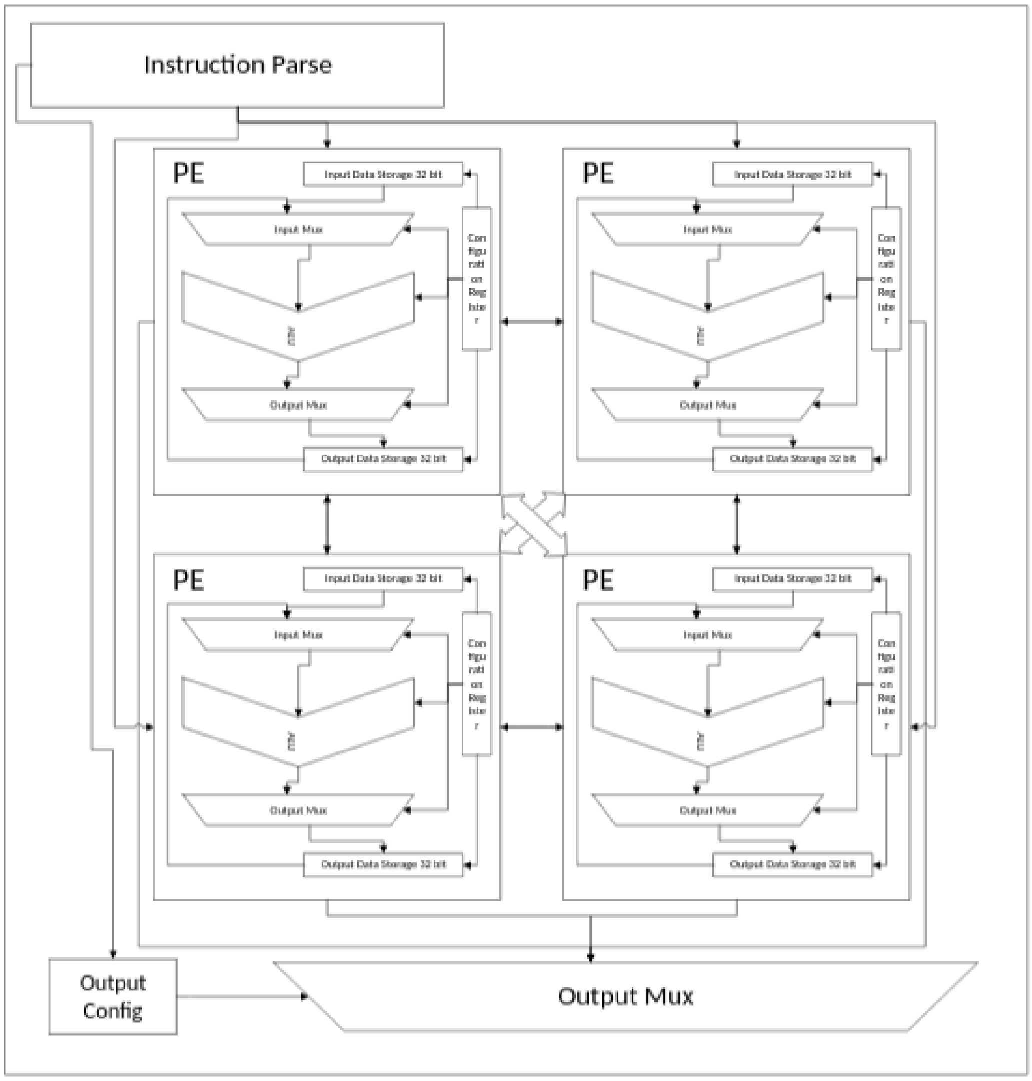


Figure 8: CyGRA Design

<u>On-Chip Memory</u>

For On-Chip Memory we are using a pre-hardened open-source 512x32 bit DFFRAM provided by OL-DFFRAM. The DFFRAM is shown below in figure 6.

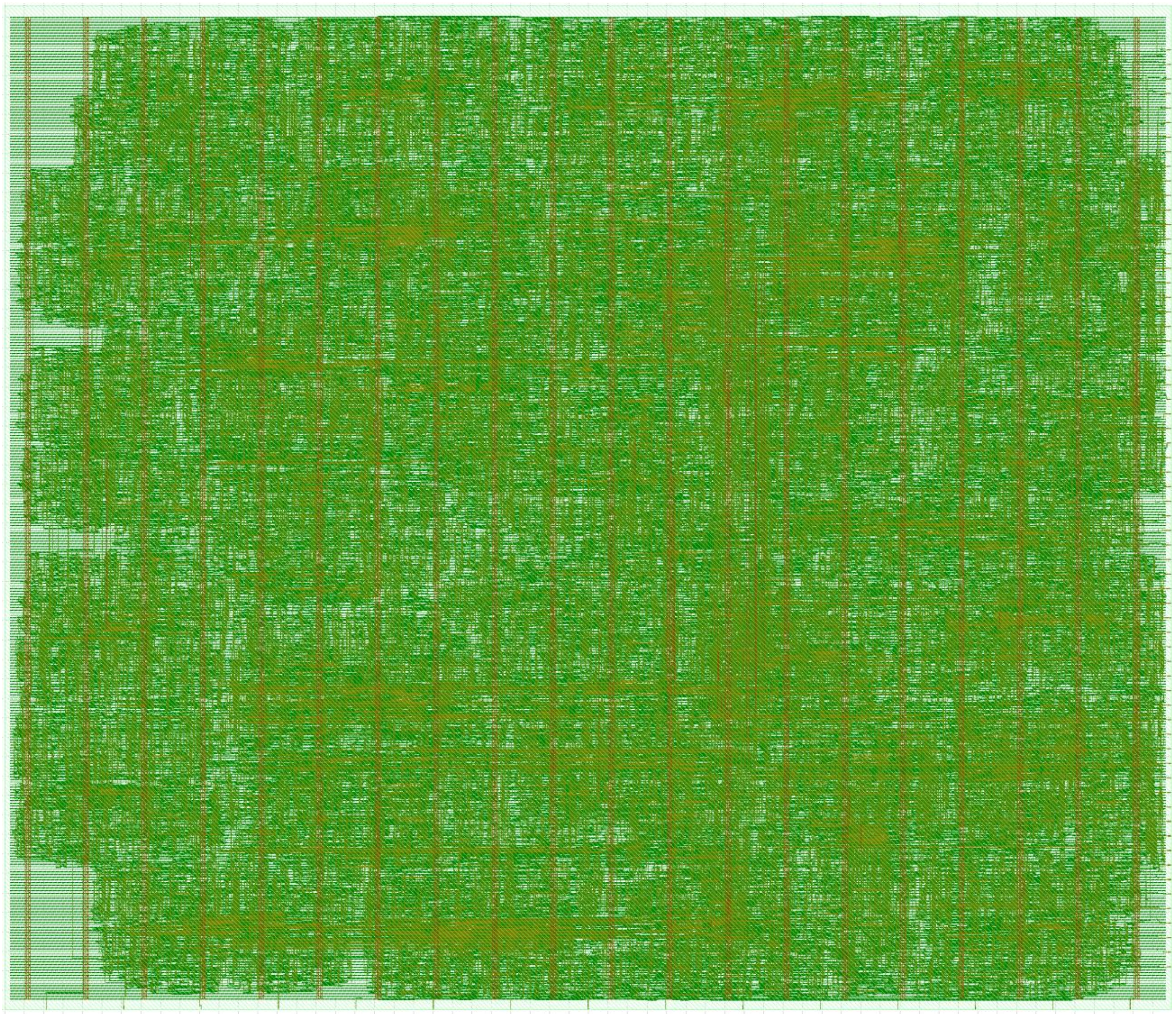


Figure 9: On-Chip Memory

The On-Chip memory is a single cycle read/write and is designed to function at 40 MHz. We may use several of these depending on how much space we have after we implement the rest of our modules. The On-Chip memory serves as cache memory for the PicoRV32.

For on-chip memory, we have also been looking into other memory solutions. The main one we have been looking at is OpenRAM, which can generate memory with parameters entered by the user. The memory OpenRAM generates is SRAM, which is smaller than DFFRAM, so we could fit more of it on our project.

<u>Other Components</u>

Off-Chip memory: Off chip SPI memory that communicates with SPI Main component described below. It has a much larger capacity than the on-chip memory but much slower read and write times.

SPI Main: Used to send and receive data with off-chip memory. Will consist of shift registers that send data bit by bit to the off-chip memory.

Interrupt module: Processes interrupt requests from the User Project Area and interprets them for the VexRISCV processor.

Memory Interface: Used as memory I/O and functions as a cache controller. It will detect when there is a cache hit and miss and will be responsible for loading off-chip memory to on-chip memory when caching.

Wishbone Slave: Used by the Management Area to write instruction memory to the Memory Interface. This memory is used to execute instructions on the PicoRV32.

Functionality

The user will load instruction memory into the VexRISCV through the flash ports of the Caravel harness. This instruction memory will be written to the Memory Interface through the Wishbone Slave. While the instruction memory is being written to, the PicoRV32 will be executing noop instruction. After writing memory, the PicoRV32's program counter will reset, and it will begin executing the written instructions.

To program the CyGRA, there will be custom instruction(s) that the user will use to write to the CyGRA's configuration registers. To execute a programmed custom instruction the user will call another instruction. These instructions will be abstracted into a C library for easy use.

Areas of Concern and Development

Our current design satisfies all specifications as laid out by our client Dr Duwe. The design implements an RISC-V soft-core that will allow both inexperienced and newer members to learn about how the Efabless caravel and supporting board works as well as allow more experienced members to learn about application acceleration with the inclusion of the custom accelerator unit.

The primary concern for our design is full functionality and completion before the April Efabless date. Beyond that, there is some concern with the CyGRA unit because it has not been implemented in Verilog and tested for functionality and size.

We will address this by working as quickly as possible to get a working prototype to perform RTL tests on. Doing this, we can debug issues quickly and see where our design may need to be modified. We will also quickly create and test individual components, so they are ready to be integrated into our design when needed. Ideally, we want all our modules coded and tested by March of next year so we can make changes if needed.

Technology Considerations

The Efabless design process requires the use of certain open-source tools. The primary tool we use is OpenLane, which is used to harden designs. These tools are well-documented but not user-friendly. This can make working with the tools difficult at times. However, Efabless has scripts and make files that do most of the work, which helps working with the tools.

Design Analysis

Currently, we have a high-level design planned out varying progress on all the modules for the high-level design. We have a completed RISC-V processor that needs to be tested in Caravel and in-chip memory selected. We have a high-level design for the CyGRA that needs to be implemented in Verilog and tested. The rest of our modules have been decided on recently and need more time to be designed, implemented, and tested. So far there has been no indication that our project will be impossible to finish before the April deadline for submitting our design.

Testing

Unit Testing

All the modules in our design will have at least one test that ensures the correct behavior of each of our components. These tests will be written as Verilog testbenches. We will test each component as they are implemented. The test will then be conducted using OpenLane tools.

Interface Testing

There are a few interfaces that we will be testing, this testing will make sure that the communication between the user area and the management area is correct. These tests ensure that we are correctly communicating over a given protocol.

Interfaces to test:

- Wishbone Slave
- Memory Interface
    - *If* implementing Off-Chip Memory
        - Cache Controller
        - SPI

Integration Testing

These tests will be written both as Verilog testbenches as well as C code. These tests will make sure that our components work as desired when they are connected. The integration test will also make sure that the RISC-V Instructions continue to work as expected. The PicoRV32 core comes with pre-existing testbenches that verifies that the RISC-V instructions are working as expected; we plan on making use of these testbenches to save time.

System Testing

The C code will enable us to program the Management Area, ensuring that user project can properly interface with the various chip utilities. It is critical that we properly test the communication between the Management Area and User Area because it is what will be used during the chip bring-up.

Regression Testing

Our integration testing will make sure that changes to our project do not change the desired behavior. Along with continually using our integration tests, we will make sure that the individual components are tested. We will retest our components whenever any changes are made. If there is a change in the desired behavior, we will make sure that it is known, and we will update the test.

Acceptance Testing

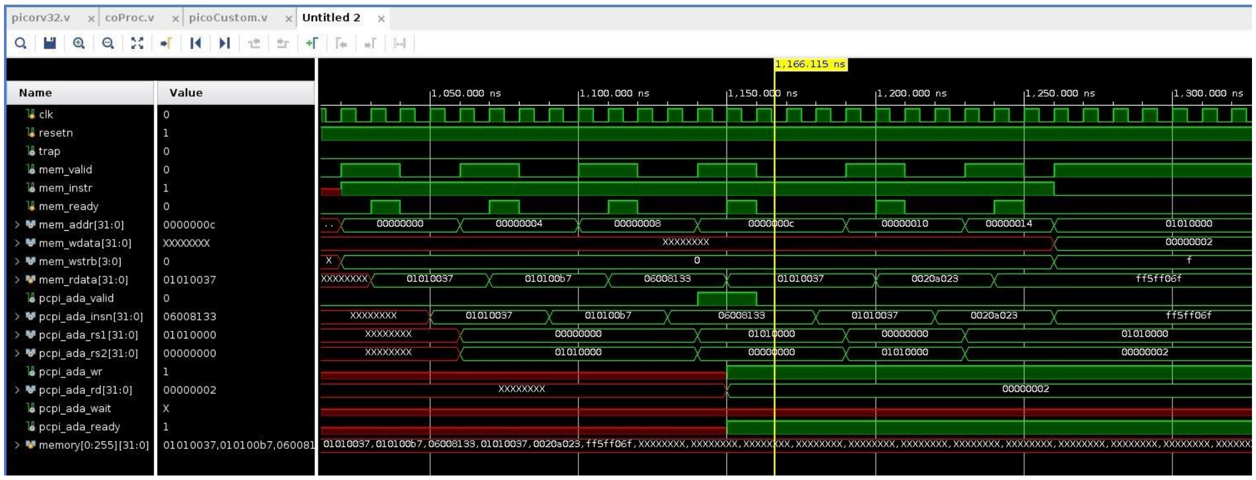
Our acceptance testing will ensure that we are meeting the desired metrics that we mentioned earlier. Meeting these metrics are important to make sure that the CyGRA is having the expected impact. We will harden our design and then ensure that the user area passed precheck to confirm that the layout matches the Verilog design. Efabless precheck will run DRC and LVS tests to make sure that there will be no issues during fabrication.

Results

The results from our tests will be in the form of waveforms, these results will be used to make sure that we are meeting our metrics. The waveforms will be used to perform regression testing to ensure that changes to our project do not have an impact on the output.

Implementation

So far, we have been able to run a pre-defined custom instruction on the PicoRV32 when simulating. Below is a simulation of an R-type instruction named ada, which performs unsigned addition of all bytes in a word. Figure 7 below shows the waveform of ada with RS1 set as 0x0101000 and RS2 set as 0x0000000.

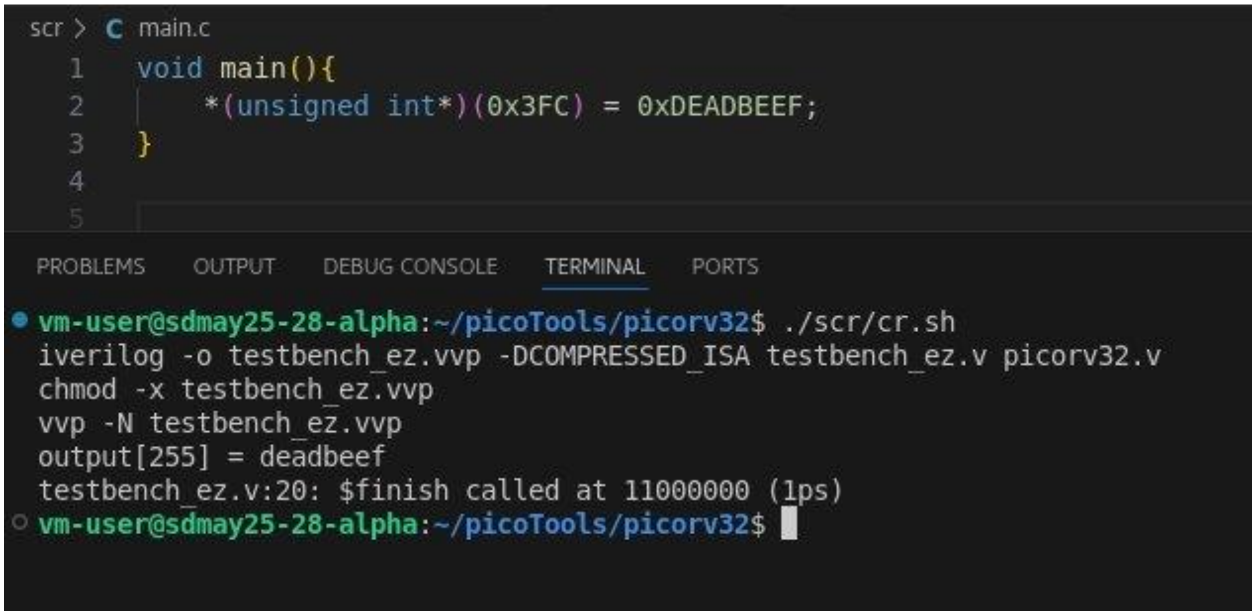


Name	Value
clk	0
resetn	1
trap	0
mem_valid	0
mem_instr	1
mem_ready	0
mem_addr[31:0]	0000000c
mem_wdata[31:0]	XXXXXXXX
mem_wstrb[3:0]	0
mem_rdata[31:0]	01010037
pcpi_ada_valid	0
pcpi_ada_insn[31:0]	06008133
pcpi_ada_rs1[31:0]	01010000
pcpi_ada_rs2[31:0]	00000000
pcpi_ada_wr	1
pcpi_ada_rd[31:0]	00000002
pcpi_ada_wait	X
pcpi_ada_ready	1
memory[0:255][31:0]	01010037,010100b7,060081

Figure 10: ada Waveform

With RS1 value of 0x0101000 being inputted into the ada module, we get the value 2 as output which we expect.

We have also been able to get basic C code running on the PicoRV32. Figure 8 below shows the PicoRV32 assigning the value 0xDEAFBEEF to address 1020 of memory.



scr > main.c

```c
void main(){
    *(unsigned int*)(0x3FC) = 0xDEADBEEF;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
vm-user@sdmay25-28-alpha:~/picoTools/picorv32$ ./scr/cr.sh
iverilog -o testbench_ez.vvp -DCOMPRESSED_ISA testbench_ez.v picorv32.v
chmod -x testbench_ez.vvp
vvp -N testbench_ez.vvp
output[255] = deadbeef
testbench_ez.v:20: $finish called at 11000000 (1ps)
vm-user@sdmay25-28-alpha:~/picoTools/picorv32$
```

Figure 11: PicoRV32 Running Code

We can see that DEADBEEF gets assigned to address 255, which is byte 1020 of memory which is what we expect.

We also have been able pass a design with the PicoRV32 and in-chip memory through pre-check, which means it has passed DRC, LVS, and STA tests and could be fabricated. The wrapper with both modules is shown below in figure 8.

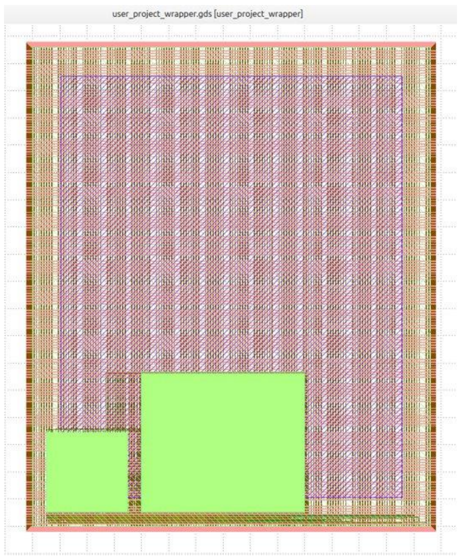


Figure 12: Hardened Wrapper with PicoRV32 and DFFRAM

The smaller green square at the bottom left is the PicoRV32 which is .6mm x .6mm and to the right of that is the 512x32 bit DFFRAM which is about 1mm x 1 mm.

Professional Responsibility

Areas of Professional Responsibility/Codes of Ethics

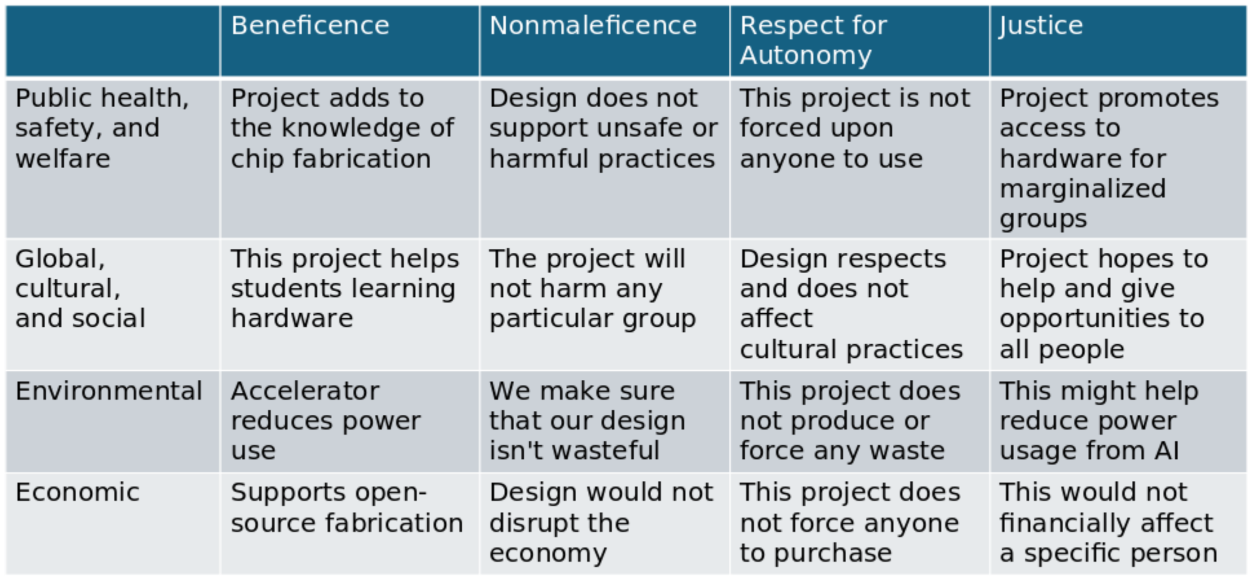
Area of Responsibility	Definition	**Corresponding IEEE Ethics Code**	**Team interaction**
Work Competence	Our definition of work competence is making sure that we are working to the best of our abilities and that we are being honest with each other if we are not completely confident in our abilities to perform a task.	**6.** to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience or after full disclosure of pertinent limitations	Our team ensured that we carried out each task to the best of our ability given time constraints and tried not to promise anything we couldn't deliver and addressed limitations
Financial Responsibility	Our definition of financial responsibility is making sure that we are not misusing funds for our project and making sure that we are making it properly open source so that others can learn from our work	**4.** To reject bribery in all its forms	Our team ensured that our project was completely open source and at the minimum cost to anyone wanting to replicate our work and were transparent with our client and stakeholders on the cost of production or desired components
Communication Honesty	Our definition of communication honesty is making sure that we are being honest with how and what our project can perform as well as being honest with our team members and our client/advisor	**3.** To be honest and realistic in stating claims or estimates based on available data	We maintained clear and honest communication with all stakeholders ensuring our technical reports and progress were reported accurately

Health, Safety, Well-being	Our definition of health, safety, and well-being is making sure that we are making a project that will improve people's lives through learning and not negatively affect others	**1.** to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, to protect the privacy of others, and to disclose promptly factors that might endanger the public or the environment	Our team focused on the technical development of our project and disclosed any information that could be problematic in a digital environment and were conscious in the cost and use of resources
Property Ownership	Our definition of property ownership is making sure that our project is properly being used and that we are properly handing our project over once we finish it	**9.** To avoid injuring others, their property, reputation, or employment by false or malicious action	Our team has focused on ensuring that our project and deliverable will persist long after the end of this project to serve as an available educational tool for future students
Sustainability	Our definition of sustainability is making sure that we are not misusing funds or negatively affecting the environment more than needed (due to the fabrication process)	**1.** To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment	Our team focused on ensuring that our project is sustainable by making it open source as well as by making this a resource for students to use for years to come
Social Responsibility	Our definition of sustainability is making sure that our project is used to help others and that everyone will be able to use our project no matter age, race, gender, etc.	**2.** to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems;	We have as a team troubleshooted at every turn in our project and done research to make each component open source and picked IP that is projected to be supported well into the future

One area of responsibility we are doing well in is communication. We keep constant contact with each other and exchange information and progress frequently. Communication is the bedrock of a good team and is indicative of strong performance because efficient communication leads to good task delegation and progress.

One area of responsibility we could improve is social responsibility. Currently our project is very technical and difficult to learn from, this could be unfair to those with less resources and less experience. To address this, we can make sure our project has documentation that properly explains the technologies that are being used and provides resources for them to use.

Four Principles



	Beneficence	Nonmaleficence	Respect for Autonomy	Justice
Public health, safety, and welfare	Project adds to the knowledge of chip fabrication	Design does not support unsafe or harmful practices	This project is not forced upon anyone to use	Project promotes access to hardware for marginalized groups
Global, cultural, and social	This project helps students learning hardware	The project will not harm any particular group	Design respects and does not affect cultural practices	Project hopes to help and give opportunities to all people
Environmental	Accelerator reduces power use	We make sure that our design isn't wasteful	This project does not produce or force any waste	This might help reduce power usage from AI
Economic	Supports open-source fabrication	Design would not disrupt the economy	This project does not force anyone to purchase	This would not financially affect a specific person

Figure 13: Four Principles Table

One important broad context-principal pair for our project is Economic – Beneficence. We all want to contribute something to the knowledge of Computer Engineering and to contribute and support the open-source community, and our project is a good way to do both. We will ensure that our project remains open source to anyone that wishes to use it.

One broad context-principal pair that our project will be lacking in is environment and Nonmaleficence. We cannot ensure that our project isn't wasteful, someone could theoretically use the CyGRA is a way that is inefficient and therefore wasteful of power. However, we do think that the potential power that can be saved by an efficiently used CyGRA will outweigh the negatives for inefficient use.

Virtues

Our team believes that the three most important virtues are Integrity, Communication, and Respect. Integrity to us means to upload honesty for the team no matter what as well as be transparent with both each other and our advisor and client to build trust and respect. Communication helps us ensure we are always in talks with each other about the project and where we are so everyone can get a good sense of progress. Respect means for us to be kind and understanding of others in the event of difficulties or other reasons and to always be understanding.

Camden:

Virtue Demonstrated -– Communication

To support communication and trust I keep in contact with the team and lead team meetings to help foster a successful and friendly environment for the whole team, so everyone feels safe to talk about whatever they need, good or bad.

Virtue Undemonstrated -– Self-discipline

One of the things I have struggled with both in this project and in life is self-discipline. Overall, I don't feel as if I have put in as much time as I have both wanted to and should, leaving my teammates on the hook for a bit more work than I would prefer. I want to be the best teammate I can and hope to improve on this next semester.

John:

Virtue Demonstrated – Honesty

One virtue that I have demonstrated is honesty. I have been honest with my team members; I believe that this is important because it shows that I am respectful to those on my team. I also want my team to see me as trustworthy and that I do what I say. I have shown this virtue through our meetings where I am honest with what I have completed and the reasonings for not completing something. I am also honest when I do not completely understand something, this makes sure that I am understanding and not just pretending.

Virtue Undemonstrated – Self-discipline

One virtue that I have not demonstrated is self-discipline, at least not enough of it. I have not been putting enough time into working on senior design as I would like. I do think I have been slacking a bit on my contributions and feel like it is disrespectful to my team not to do more. I will make sure that I not only contribute more time to senior design, but also that I work earlier in order to be able to contribute more and to give us more time to fix issues.

Nicholas:

Virtue Demonstrated – Diligence

One virtue that I have demonstrated is diligence. If I have a task I need to do, I will always strive to work hard towards completing that task. Through the beginning of the project, I spent most of most time out of everyone learning the Efabless tools and spending long periods of time trying to resolve issues I had while learning so I could help my teammates with it in the future. I was able to complete a full project by the beginning of November, which helped give me a good knowledge of Caravel and OpenLANE before anyone else in my group.

Virtue Undemonstrated – Being Collaborative

One virtue that I have not demonstrated is being collaborative. Out of everyone in the group, I spent most of my time alone doing my own thing, which had some benefits but is outweighed by not having any else able to work on your tasks. I hope to fix this by communicating my progress and teaching my other group members more about my work so they can help me complete tasks in the future.

Calvin:

Virtue Demonstrated – Humility

Throughout the beginning of the design process, we have run into many difficulties when it has come to making decisions about the scope of our project and the design itself. As a rule, I tend to have my head in the clouds in relation to ideas, and I tend to be very stubborn about changes that others suggest for these ideas. Because I find myself lacking in this area, humility is very important for me to be able to work in a team and client driven environment. Throughout this first semester I have made an active effort to maintain humility regarding making decisions as a team, as well as readily accepting feedback from our advisor and client as to our direction

Virtue Undemonstrated – Clarity

When you are designing a product that other people in the future will learn from, or that they might find use in, you have to keep in mind the ease of use from their perspective. Your code cannot be inscrutable if you want others to find value in it. So far, I have been lax on demonstrating clarity due to all the code I’m writing being test code, but going forward into the next semester, I hope to make an active effort toward all production code being well organized and readable, with clear documentation and comments.

Levi:

Virtue Demonstrated – Adaptability

As our project has progressed our project has changed and adapted and the perceived skillsets and assumed roles of some people in our group have changed. For myself I started off as a client interaction lead, then a integration lead, and now that role is refined more to a SPI/Serial Communications lead (WISHBONE & SPI) and maintain our website. I think I have managed well with picking up different hats and not dying on a hill of needing to play a specific part. I originally wanted to be the Caravel lead but after that role was already taken, I found a new role that still lets me interact with caravel in ways I want to, but contributing something else the team will value more.

Virtue Undemonstrated - Self-discipline

One virtue I haven't displayed well this semester is self-discipline. This semester has been the busiest semester I've had in college, and the group I am working with for the project is extremely competent and capable. It has been easy during this project to after a long week of school, to not focus on senior design for the weekend because I know it will all be fine. I think I could take more initiative in the next semester and use more of my free time to work on this project to help my teammates and let them know how much I value them and give them time to slack off in turn.

Closing Material

Conclusion

For our project, we have put in a lot of work learning the Efabless tools, planning, and design. So far, we have made good progress in achieving our goals for the project. We have gotten a design through the Efabless process. We also have a RISC-V processor picked out and we are able to execute a pre-defined custom instruction on it using a co-processor interface. This is a large part of our project; it makes good progress in achieving our goals. Now we need to implement a user-defined instruction unit and include everything in a top-level Caravel wrapper.

Our main objectives for the spring are to complete and test all our modules, integrate these modules into a top-level design, test the top-level design, and send our project to be fabricated. We feel we are on pace to get our project done by the April submission deadline.

References

CGRA Architecture and Tools | AHA Agile Hardware Project. https://aha.stanford.edu/research/cgra-architecture-and-tools. Accessed October 11, 2024.

Efabless Caravel "Harness" SoC — Caravel Harness Documentation. https://caravel-harness.readthedocs.io/en/latest/. Accessed September 20, 2024.

"PicoRV32 - A Size-Optimized RISC-V CPU." GitHub, https://github.com/YosysHQ/picorv32. Accessed October 30, 2024.

Todd, Dillon. "Tightly Coupling the PicoRV32 RISC-V Processor with Custom Logic Accelerators via a Generic Interface." All Theses, May 2021, https://open.clemson.edu/all_theses/3552. Accessed November 15, 2024.

Appendices

Criteria	**Rocket Chip**	**PicoRV32** **(Chosen)**	**Neorv32**	**VexRiscV**	**CVA6**
Instruction Set	RV64I + IMAFDC	RV32I + MC	RV32I + Zicsr Zifencei	RV32I + M	RV64I + MAC
Written Language	Chisel	Verilog	VHDL	SpinalHDL (Scala)	SystemVerilog
Features	Everything + ROCC	-Multiplication -Wishbone Master interface -Compressed Instructions -Axi interface	-CSR instructions -Instruction-fetch fence	-Multiplication -Wishbone ready	-Linux -Privilege levels -

		-PCPI interface			
Size optimization	No	Yes	No	No	No

Table 3: RISC-V Core Decision Matrix

DFFRAM GitHub repository: https://github.com/efabless/OL-DFFRAM

Team

Team Members

1. Camden Fergen
2. John Huaracha
3. Nicholas Lynch
4. Calvin Smith
5. Levi Wenck

Required Skills Sets

- Digital VLSI
- Hardware Design
- Teamwork

Skills Set Covered by the Team

- Hardware Design
- Team software development
- Electrical knowledge
- Embedded systems
- Agile
- CI/CD
- Analog and Digital VLSI
- Verilog/VHDL

Project Management Style Adopted by the Team

- Agile

Initial Project Management Roles

- Camden Fergen – DevOps and Project Lead
- John Huaracha – Testing Lead
- Nicholas Lynch – Harden and Verification Lead
- Calvin Smith – Accelerator Design Lead
- Levi Wenck – Communication Interfaces Lead

Team contract

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings
   - Team Meeting: Tuesdays, 5:30pm, Senior Design Lab/TLA
     - other meetings are arranged as needed via Discord
   - Meeting with Duwe: Friday, 3:00pm, Durham 353
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
   - Discord - Internal
   - Email, Teams - External
3. Decision-making policy (e.g., consensus, majority vote):
   - Majority vote
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
   - Once the team meeting starts, John will keep track of time. We'll use an excel sheet

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
   - Be there at least by 6:00, notify 24 hours in advance if you can't make it
   - Be a team player
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
   - Ensure all assigned tasks are completed on time. If you are unable to make a deadline, let the team know in advance.
   - All tasks must be completed in full by the deadline.

3. Expected level of communication with other team members:

   - Communicate a lot, there is no such thing as over communicating
   - Let people know what you are working on and when you run into problem when needed/when decisions are made

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

   - We expect everyone to commit to the team and to the project
   - Even if a decision is made which you are against, you should be putting in full effort
   - Get your tasks done on time and communicate if you won't be able to

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

   - Camden Fergen – DevOps and Project Lead
   - John Huaracha – Testing Lead
   - Nicholas Lynch – Harden and Verification Lead
   - Calvin Smith – Accelerator Design Lead
   - Levi Wenck – Communication Interfaces Lead

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

   - Be a nice person
   - Be an understanding person
   - Do good work
     - Documentation/comments

3. Strategies for recognizing the contributions of all team members:

   - Using some sort of project management tool (Jira, Gitlab, etc.)

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

   Calvin Smith:

   - Hardware Design
   - Team software development
   - Basic electrical knowledge

- Embedded systems

Camden Fergen:

- Digital design/VHDL (CPRE 381)
- Software development and low level coding (C, Assembly)
- Embedded system integration
- CI/CD

Levi Wenck:

- General Software Development skills & CI/CD (AGILE)
- Embedded Systems experience & network analysis (Internships)
- RTL focused CprE Degree

John Huaracha:

- CI/CD & Agile
- Embedded Systems
- VLSI (EE 330)
- VHDL & Verilog

Nicholas Lynch:

- Low Level Programming
- Basic Analog and Digital VLSI design
- VHDL & Verilog design

2. Strategies for encouraging and supporting contributions and ideas from all team members:

   - Be open to ideas
     - Understand that not everyone knows everything

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?)

   - Write/document complaint(s) as to help define what the issue is (Privately/Publicly)
   - Request a team meeting
   - Communicate the issue with all team members

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    - Have Fun
    - Have a prototype finished
2. Strategies for planning and assigning individual and team work:
    - Coordinate tasks during team meetings or on discord.
3. Strategies for keeping on task:
    - Holding other teammates accountable for their work.
    - Setting deadlines for all tasks.

Consequences for Not Adhering to Team Contract

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

- Have a team meeting to discuss any issues with everyone present

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

- Contact course instructors to find a resolution

****

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) John Huaracha	DATE	09/17/2024
2) Levi Wenck	DATE	09/17/2024
3) Nicholas Lynch	DATE	09/17/2024
4)Calvin Smith	DATE	09/17/2024
5) Camden Fergen	DATE	09/18/2024