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: May 3rd, 2025

Version 1.5

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 an open-source coarse-grained reconfigurable architecture named the CyGRA, attached to an open-source microcontroller. 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 also hope our project can also serve as a jumpstart 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 co-processor integrated into a microcontroller to extend an open-source ISA. There have been a handful of decisions made for our design: We have chosen to use the PicoRV32 as our microcontroller core, DFFRAM for on-chip memory, as well as a cache system to extend the memory size and utilize off-chip memory. The goal for our CyGRA co-processor is to accelerate certain instructions, as well as allow users to define their own configurations to potentially accelerate their own instructions. We also have experience with the open-source tooling provided by Efabless, so in the event an opportunity arises we can get our chip fabricated.

Update: As of March 2nd, 2025, Efabless has shutdown. Because of this, we currently do not have a plan for getting the chip fabricated. Regarding our senior design project, this has little change in terms of our project design, and we will continue to use the Efabless tooling, but we no longer have any access to tech support from Efabless in the event we run into issues. Additionally, we now no longer need to have the project ready for Efabless's April 21st deadline, which will allow us more time to design and test our system

Learning Summary

Development Standards & Practices Used

IEEE 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 Skywater 130nm process
- Entire project must be open source
- Function as a microcontroller when provided with standard 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
- Memory system
- Reconfigurable architecture
- Using open-source tools

Tools:

- Caravel Harness
- CocoTB

- Magic DRC
- Netgen LVS
- OpenLANE/OpenROAD

Table of Contents

List of Symbols and Definitions

Efabless – Currently shutdown open-source fabrication company. Has provided many open-source tooling as well as the Caravel Harness

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

SoC - System on chip

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, who have also created a tool for testing caravel projects on an Xilinx Artix FPGA

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

FPGA - Field Programmable Gate Array, reconfigurable integrated circuit

SPI - Serial Peripheral Interface

PCB - Printed Circuit Board

Verilog HDL - Verilog Hardware Description Language

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

Embench-iot – The collection of open-sourced benchmarks based on Bristol/Embecosm Embedded Benchmark Suite for modern embedded systems.

Introduction

Problem Statement

Processors are constrained by their defined instruction sets. If an operation isn't supported, designers face three options: redesign and refabricate the processor (which is costly and time consuming), approximate the operation using multiple instructions (which can be inefficient or inaccurate), or avoid the operation altogether. In many cases, none of these options are practical.

Currently, few research processors are suitable for learning processor design because most lack low-level accessibility and are difficult to modify. This forces students and educators to rely on alternative methods, which can limit hands-on learning.

To address this, we aim to design a system-on-chip (SoC) that allows users to define and implement custom instructions directly in the processor. Our goal is to make these instructions highly customizable, giving users maximum flexibility to tailor processor behavior to their needs. Additionally, because our team is interested in reconfigurable computing, this project will enable us to explore that field further by designing a dedicated reconfigurable computing unit.

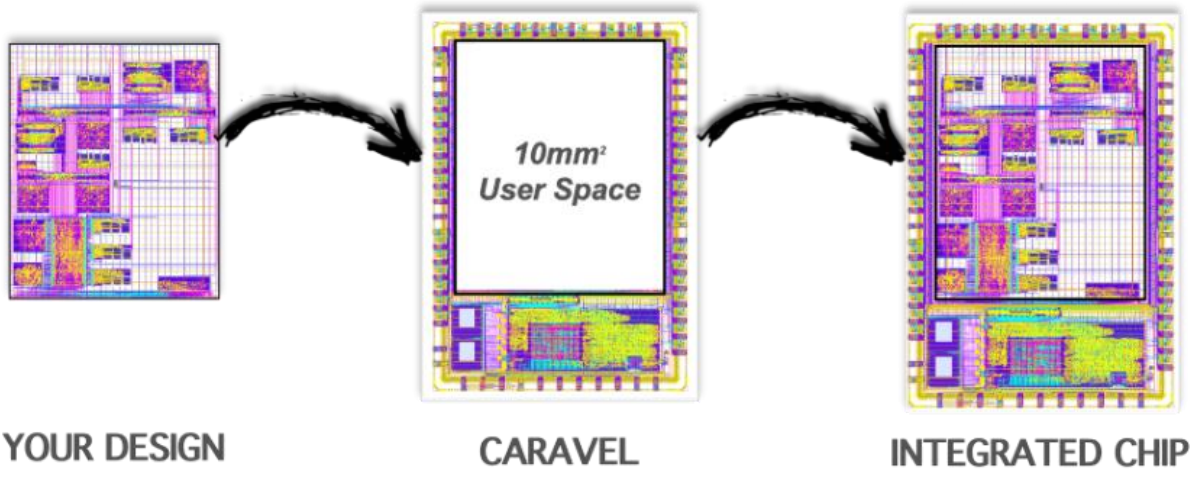

Figure 1: Caravel Chip Design

For our ASIC design, we will use Efabless's Caravel platform along with the OpenLANE tooling. Efabless's Caravel will act as a harness that we will integrate our project into. This provides us with a management area that is powered by a VexRISC-V processor, including features such as logic analyzers, interrupt pins, and a wishbone bus including clock and reset signals, which we will use to support our design. OpenLANE is a collection of open-source tooling that we will use to generate the physical layout of our chip from the Verilog

code. Additionally, it also runs additional checks such as DRC (Design Rule Check), LVS (Layout Versus Schematic), and STA (Static Timing Analysis).

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 Efabless's Caravel and open-source tooling, 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 reconfigurable ISAs and complete SoCs. 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 open-source ISAs, and microprocessors. Our project will expand professors' teaching options and allow them to instruct their students more effectively.

Requirements, Constraints, And Standards

Requirements & Constraints

Functional requirements:

- Function as a microcontroller when provided with normal ISA instructions.
- Support custom instructions defined by the user.
- Custom instructions should only be executed when called (should not execute custom instructions when provided standard ISA 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 ISA softcore used must be open-source.
- Chip tapeout in the SkyWater 130 nm 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 ISA instruction.

Physical:

- Must function at room temperature (≈20°C).
- User project must be 3mm x 3.6mm (10mm^2) to fit in the 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.

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

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.

IEEE 1364-2001: IEEE Standard Verilog Hardware Description Language

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

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

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.

Wishbone Bus

Our project will use the wishbone protocol as implemented by Efabless to communicate between the user area and the management area. Efabless's process has this protocol implemented as a non-option, with a slight variation from the OpenCores wishbone protocol.

SPI Protocol

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 linked to an SPI 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 softcore 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-source IP designs
        ii. Choose the best open-source IP for our project

            iii. Test hardening 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 CyGRA
        d. RISC-V ISA extension
            i. Add basic instruction for testing
            ii. Integrate CyGRA into a custom instruction
        e. PicoRV32 test plan
            i. Create toolflow for testing
                1. Behavioral
                2. C code
            ii. FPGA testing flow
                1. Compile embench-iot benchmarks
                2. Run as baseline to compare to CyGRA acceleration
        f. Management area interface
            i. Determine how interrupts are supplied to processor
            ii. Determine how the management area will interface with on chip memory
4. Integration
    a. Combine memory system with softcore
    b. Connect CyGRA to the softcore
    c. Test memory system
    d. Integrate wishbone with management core
5. Testing/Verification
    a. Ensure full functionality of microcontroller
    b. Ensure implemented instruction works as expected
    c. Ensure memory interface is working correctly
    d. Run FGPA tests of full system and embench-iot benchmarks
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 data path 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 softcore
    - Determine which open-source ISA softcore best fits our use case
    - Ensure the chosen softcore works in the Efabless tooling
    - Design high level of chip with included softcore
- Milestone 4: Accelerator/CyGRA design
    - Determine the specific application to accelerate
    - Develop simple Verilog module to accelerate use case
    - Backport work to a CyGRA design
- Milestone 5: System Integration
    - Combine the CyGRA and the softcore
    - Include on chip memory
    - Wishbone bus integration to management core
- Milestone 6: Overall Design Testing
    - Ensure the softcore can interface with both on and off chip memory through cache system
    - Ensure the management core can write to the memory and reset the softcore
    - Ensure the custom instruction for the CyGRA works as intended
- Milestone 7: Synthesize Layout
    - Synthesize each component individually
    - Synthesize high level design
    - Perform gate level tests
- 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 2). 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

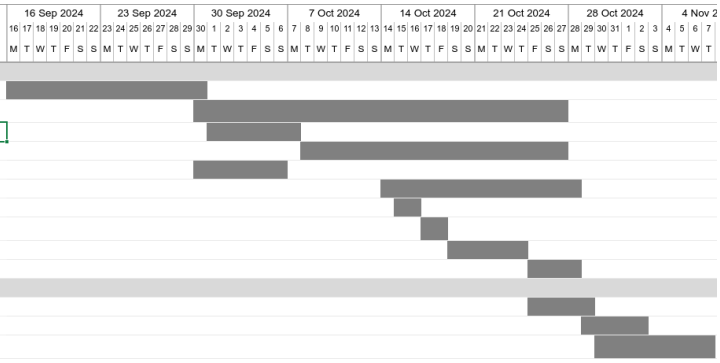

Figure 2: Project Prep and November Chip

Our next deadline is focused on the full design and chip tapeout/submission.

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

Our Gantt chart has been 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

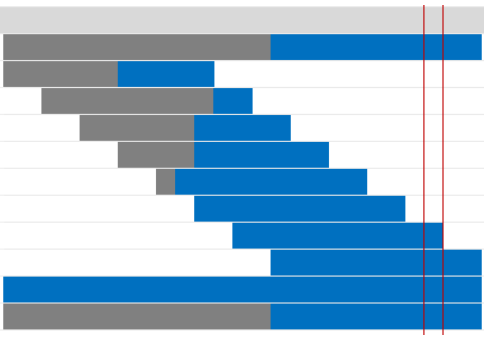

Figure 4: Documentation

Figure 4 represents some of our documentation objectives that (at the time) were to be completed. You can see that they are staggered as each section built upon the last.

Project Start Date 8/26/2024 (Monday) Display Week 12 Week 12 Week 13 Week 14 Week 15

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 face is that our project may become unfeasible due to time constraints. However, this risk is fairly minimal due to the guidance that our advisor, Dr. Duwe, gives us during our weekly meetings. This ensures that we will end up with a complete project by the end of the year. While unlikely, if the feasibility of our project is at

risk later in the project, we plan to mitigate this by developing a fallback plan to ensure that we can deliver a functional product in the event of major setbacks.

Another significant risk we face is the potential for time delays, as before this project we were not familiar with Efabless's Caravel or OpenLANE tools. This lack of experience increases the likelihood of unexpected challenges that could slow our progress. While minor setbacks alone may not have any serious consequences on our timeline, persistent delays could lead to larger problems later on. To mitigate this risk, we are adopting an agile management approach, allowing us to address issues more effectively as they arise. Additionally, we are prioritizing clear and consistent communication within our team to ensure we keep steady progress and quickly resolve roadblocks.

The largest challenge we've faced so far is the time required to complete certain components, the most notable of which is the CyGRA, which has proven to be much more complex than anticipated. Additionally, the SPI master unit took us longer than expected to adapt to our project which caused delays in integration and testing. To mitigate this, we have split our team into multiple leads so during any delay, other members can continue with other modules of the chip to keep us on track.

Personnel Effort Requirements

Our task list is constantly evolving, making it difficult to assign expected work hours to dedicated tasks. Time estimates can be easily skewed by factors such as the number of work sessions and length of time spent by each member, which can vary in efficiency depending on the day.

Task	**Projected Hours**	**Actual**
Workflow Tools/Setup	40 Hours	40 Hours
November Chip Design	20 Hours	30 Hours
Project Design	100 Hours	200 Hours
Integration	60 Hours	100 Hours
Testing/Verification	40 Hours	Ongoing (currently>50)
Documentation	40 Hours	Ongoing (currently>40)

Table 1: Projected Hours

We have completed design and individual testing of all our modules, and are currently working on integration of major components such as the CyGRA into our microcontroller. We underanticipated how complex our CyGRA would be, which has been a major contributing factor to our design time.

Other Resource Requirements

Tools and work environments that are beneficial to larger team productivity have been very helpful in this project. As such, we contacted ISU's ETG (Electronic and Technology Group) that helps support senior design teams to create a virtual machine to work on. This has allowed us to work collaboratively remotely and ensure our environment has identical variables to reduce errors relating to the setup of the tools we use. We have used this virtual machine to build the RISC-V toolchain as well as generate benchmarks from embench-iot.

Beyond this, we have also accessed and used the tools provided by the ISU ChipForge club. This has allowed us to run system tests on an Artix FPGA board in Durham, which has provided valuable feedback on our design and development.

Design

Design Context

Broader Context

One of our main goals in designing this project was to keep the technology accessible. To support this, we used only open-source tools and made all our designs open-source, allowing others to expand and improve upon them. This ensures that our final deliverable serves as a valuable education and research resource to help other students and teams gain entry into the field of hardware design.

Below is a table of some considerations we made when designing and developing our project:

Area	Description	Example
Public Health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups?	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,	Our project adds to the list of open-source designs that the global electrical and

	practices, and aims of the cultural groups it affects?	computer engineering community can use and expand upon.
Environment	What environmental impact might your project have?	Custom instruction can make some processes take less instructions and less energy, which can add up when done many times.
Economic	What economic impact might your project have?	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

        - Required 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)

Arm Custom Instructions

- 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.

- Deciding on an open-source ISA softcore to serve as the main compute unit
- Creating a CyGRA (Coarse-Grained Reconfigurable Architecture) co-processor to implement custom instructions
- Designing a memory interface and caching unit to manage on- and off- chip memory
- Implementing a SPI master interface to communicate with the off-chip SRAM
- Modify a Wishbone slave interface to enable communicate and data transfer to the user project area (including writing processor code to memory)
- Integrate all components into a complete system, allowing users to configure the CyGRA to execute custom instructions alongside the main softcore

Design Exploration

Design Decisions

For our project, we made several important design decisions, outlined below:

- Using an open-source ISA
    - Selected the RISC-V ISA
    - Simple, fully open-source, widely adopted
    - Commonly used and taught in universities and research settings
- Selecting the PicoRV32 softcore
    - Relativity easy to integrate into our design
    - Includes a built-in co-processor interface, simplifying integration of the CyGRA
    - Size-optimized, occupying only 0.6 x 0.6 mm^2 within the SkyWater 130nm process
- Implementing a Coarse-Grained Reconfigurable Architecture (CyGRA) for custom instructions
    - Complex module to design
    - Challenging to integrate effectively into the overall system
    - New area for our team, requiring new research and learning

An important consideration throughout our design process was ensuring that all chosen components and designs were fully compatible with the Efabless Caravel platform and toolchain. Every decision had to account for both smooth integration between components and support within the Caravel wrapper. This was the main reason we did not

consider the MIPS ISA, despite our team having experience with it through various ISU classes.

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	Neorv32	Vex RISC-V	CVA6	**PicoRV32**
- RISC-V 64 bit - Feature rich - Not size optimized - Written in chisel	- RISC-V 32 bit - Designed as a microcontroller - Not size optimized - Written in VHDL	- RISC-V 32 bit - Highly customizable - Not size optimized - Written in SpinalHDL	- RISC-V 64 bit - Support for UNIX-like operating systems - Not size optimized - Written in System Verilog	- RISC-V 32 bit - Wishbone interface and high clock frequency - Size optimized - Written in Verilog

Table 3: Comparison of various softcores that were considered for our project

Decision-Making and Trade-Off

When deciding between the different RISC-V softcore designs we established a few criteria to rank the designs we found. The criteria are as follows:

- Instruction Set
  - Used to categorize processors based on their supported architecture (32-bit or 64-bit)
  - Determine which processors supported the most extensions of their given ISA (i.e. RV32I vs RV32IM)
- Written Language
  - Since the Efabless process and Caravel requires designs in Verilog, we prioritized processors written in Verilog for ease of integration
- Features
  - Assessed additional capabilities included in the base design, such as Linux compatibility or built-in Wishbone interface
- Size Optimization
  - Considered whether the processor's repository specifically mentioned size or space optimization – critical to our design as the Efabless Caravel has a limited user area of 2.92 mm x 3.52 mm ($10mm^2$), which must fit both the processor, memory, and custom co-processor

After consideration, we decided to use the PicoRV32 as our RISC-V softcore. The reason that we decided on this was for a couple of reasons. First off, it was written in Verilog which

allows us to easily integrate it into the Efabless Caravel. Secondly, it is a size optimized design, which is important considering the space available in the user area, allowing us to dedicate more space to other components. Lastly, it includes a Wishbone interface, which we can use to integrate with the management area of the Caravel.

Final Design

Overview

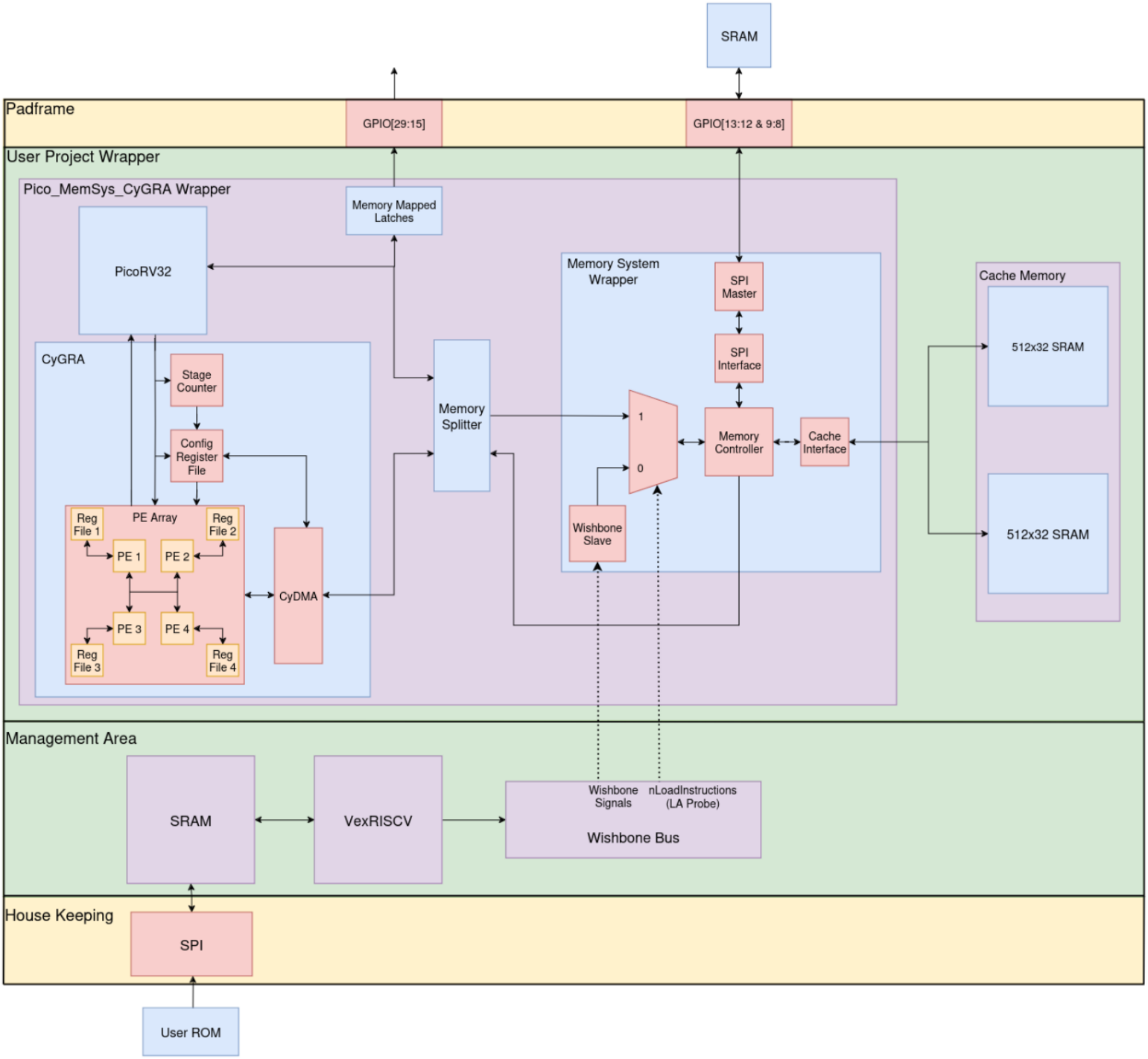

Figure 6: High-level design contained within the user area

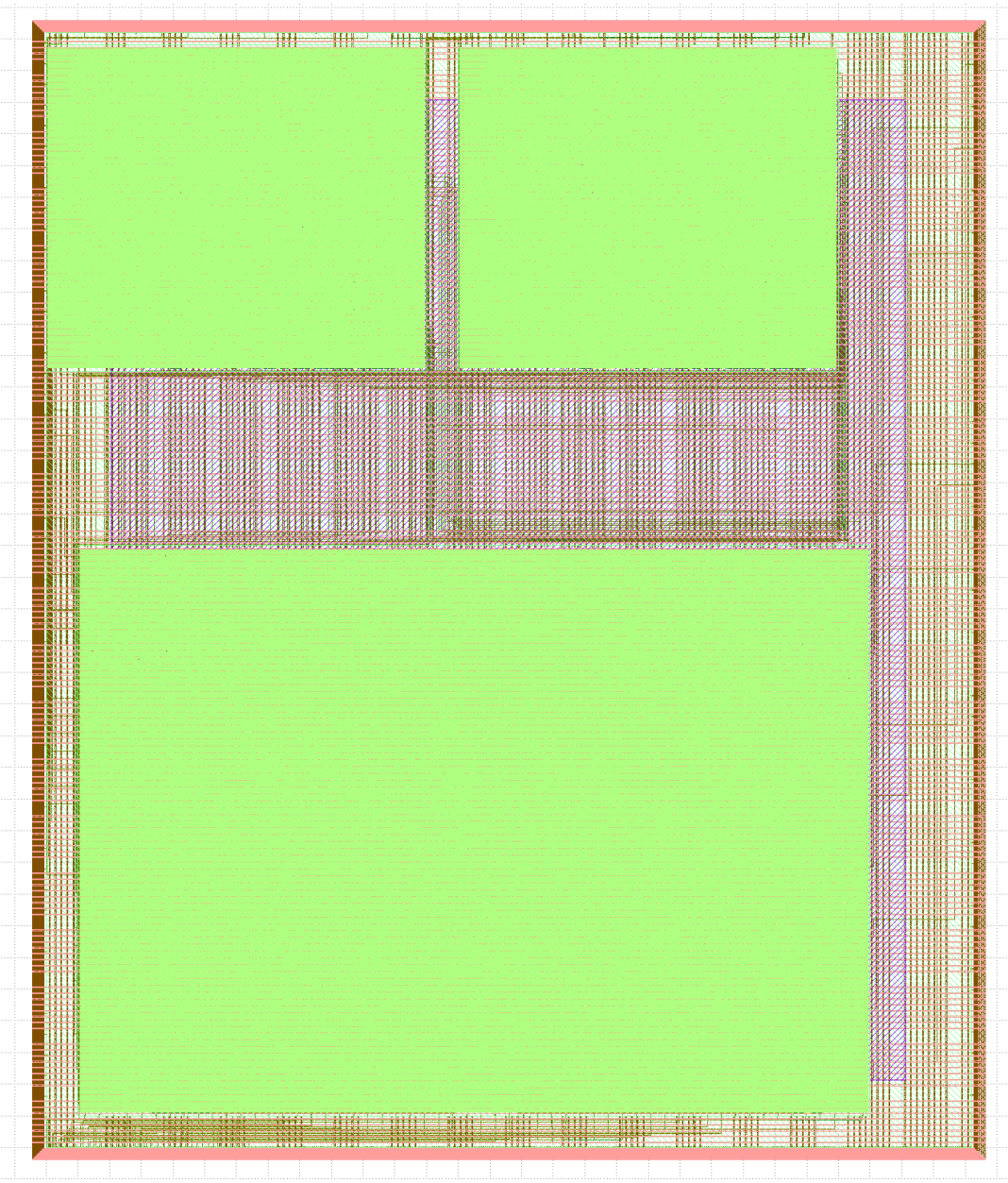

Figure 7: Synthesized Layout of Circuit

The high-level design (as seen in figure 6) consists of two main areas, the management area, and the user project area. The management area is provided as is from Efabless and contains a VexRISCV processor integrated with a Wishbone which provides the clock, reset, and buses for communication into the user project area. Additionally, the management area provides logic analyzers so that after fabrication you can still probe sections of the user area for testing to ensure things are working as expected. The management area also has flash inputs which allows the user to load programs onto the VexRISCV processor.

The user project area is a ~10mm^2 section where our design is placed into. It has access to 34 GPIO ports, allowing us to set them as either input or output depending on what is needed, such as for the off-chip memory as seen in figure 6. Within the user area is the PicoRV32, CyGRA, memory system, and cache memory.

Figure 7 shows the synthesized layout of the user_project_wrapper. The larger rectangle on the bottom is the wrapper containing the PicoRV32, Memory System, and CyGRA. The wrappers area is 2.5 mm x 1.8mm which is 4.5 mm^2. The two smaller rectangles on the top are the DFFRAM modules, they both have an area of about 1mm^2.

Detailed Design and Visuals

PicoRV32

The PicoRV32 is a size optimized RISC-V 32-bit processor, which we have chosen to be the main compute unit for our design. The PicoRV32 is a non-pipelined processor, meaning that it only runs a single stage at a time (as shown in figure 8).

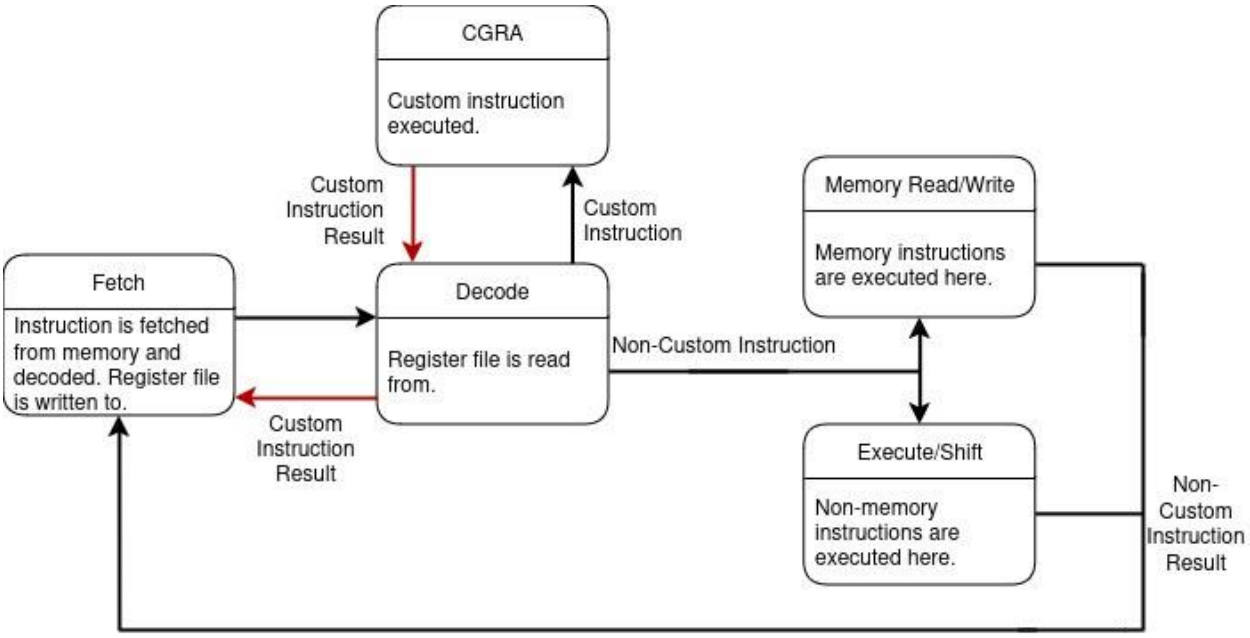

Figure 8: PicoRV32 Datapath

This processor supports the full set of standard 32-bit RISC-V instructions but also allows for custom instructions through a co-processor interface, which triggers when an unknown instruction is decoded. We leverage this interface to seamlessly integrate the CyGRA with minimal overhead into the main processor. Below is an example of how the pipeline operates with the CyGRA:

1. Fetch stage
    a. Fetch the next instruction from memory
    b. Decode the fetched instruction and set control signals
    c. Write back the results of the previous instruction to the register file (if needed)
2. Decode stage – Splits into one of five stages (listed below)
    a. Memory read: for load instructions
    b. Memory write: for store instructions
    c. Shift: for shift operations
    d. Execute: for all other standard instructions
    e. CyGRA: for custom instructions handled by the CyGRA

Once a stage is completed, control returns to the Fetch stage. The memory Read and Write, Shift, and Execute stages all transition directly back to Fetch. However, the CyGRA stage routes through an additional Decode stage before returning to Fetch. Below is a table showing the cycles per instruction (CPI) for standard (non-custom) instructions, as documented in the PicoRV32 readme, along with their runtimes based on a 40 MHz clock speed (the max clock speed provided in 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

Table 4: Expected CPI and runtime for the PicoRV32
Note: The memory figures above assume 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 miss would result in a high CPI and slow runtime due to SPI being only able to transfer one bit at a time

Our aim for custom instructions is to take 3-30 CPI depending on which instruction is being executed, meaning that the runtime would range from 75ns to 750ns.

The PicoRV32 writes and reads memory over our implemented memory interface (described later in this section).

CyGRA

The CyGRA is our custom coprocessor consisting of an array of 4 processing elements arranged in a 2 by 2 grid that is capable of reading and writing directly to and from memory. Each processing element contains its own register file, containing 32 registers, and this helps lower the amount of DMA or DMO commands that are necessary. There is a stage

counter which keeps track of the amount of stages/cycles that will be necessary for the command to work appropriately.

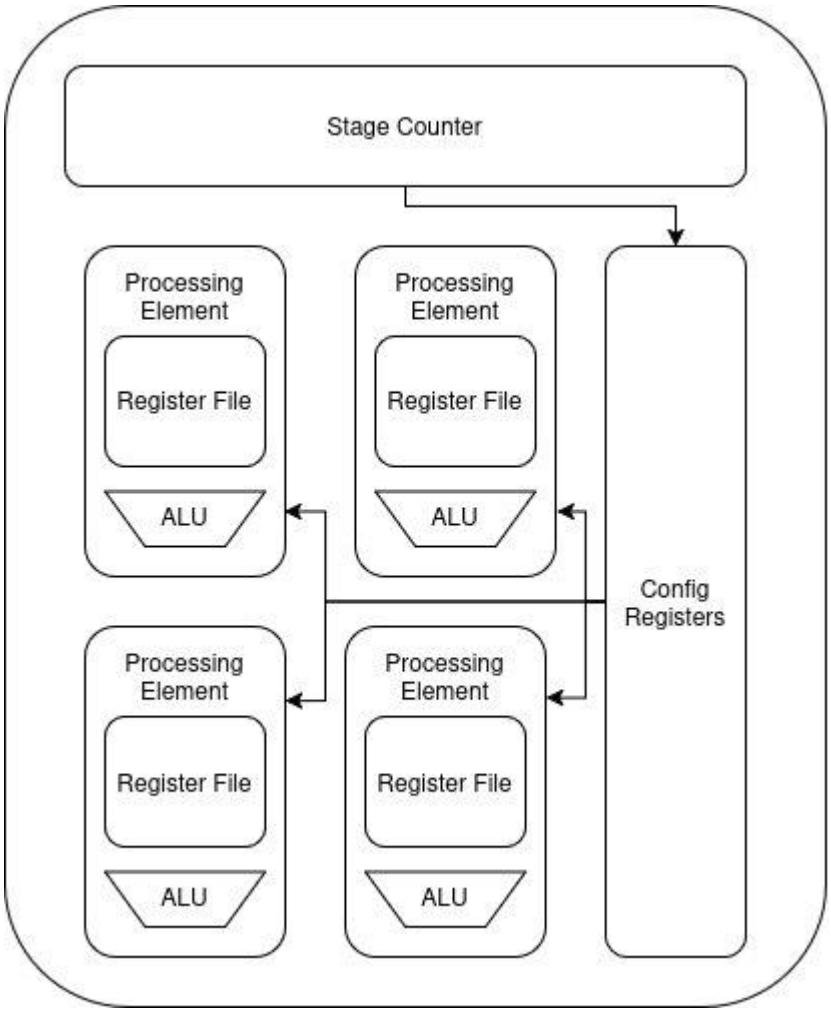

Figure 9: CyGRA

Processing Elements

Each processing element consists of an ALU, register file, and input muxes capable of integer arithmetic as well as data passthrough. Each processing element is capable of 4 operations: addition, subtraction, multiplication and passthrough. Passthrough is used to enable shuffling of data between register files.

Memory System

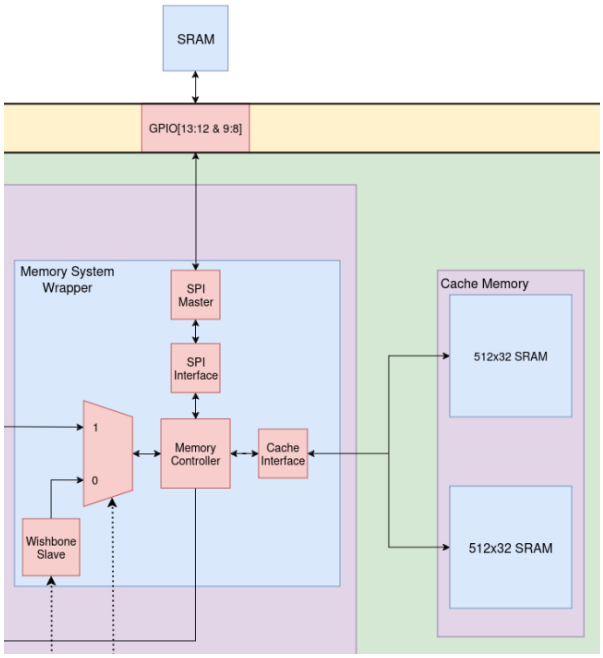

Figure 10: Memory subsystem

The Memory System provides interfaces for the Wishbone Slave, PicoRV32, and CyGRA to interact with. The Memory System decides which component has access to memory. If nLoadInstructions is low, then only the Wishbone Slave access memory. If nLoadInstructions is high, then either the PicoRV32 or CyGRA can access memory. The PicoRV32 has priority over the CyGRA (If both try to access memory, the PicoRV32 will be given access.)

The Memory System manages memory between the On-chip cache memory and Off-Chip SRAM. The Memory System uses a write-through cache with a write allocation policy, meaning the cache will get written on all writes and read misses. This results in considerable performance improvements when running programs.

On-chip memory

Consists of two pre-hardened open source 512x32 DFFRAM modules. This works as cache memory for higher performance. It works in tandem with the Off-chip memory to ensure a large but fast memory system.

Off-chip memory

512 Kbit SRAM PMOD module that communicates with the design over SPI. This acts as the RAM as seen in a typical system, allowing us to load larger programs without using the entire user space for memory

SPI master unit

The SPI master unit is a modified Efabless SPI master unit, designed to allow the memory controller to read and write, over SPI, to the off-chip SRAM PMOD module.

Wishbone slave

Reads data sent from the Management Area. The Management Area sends instruction memory and other data needed for programs to run on the PicoRV32. This includes programming the memory for the PicoRV32 to run programs.

Functionality

The user can write a C program, which can be compiled to hex by the make_firmware.py python script. This hex can be flashed to the Caravel chip using ChipForge tools (documentation on how to do this is provided in the ChipForge GitLab).

After flashing the hex firmware, instruction memory will be written to the user area memory using the Wishbone bus. During this process, the PicoRV32 remains idle, waiting for nLoadInstructions to go high. Once the VexRISCV completes writing, nLoadInstruction goes high and the PicoRV32 begins reading instructions from memory and executes them.

Our CyGRA is designed so that the user first loads a configuration, then inputs data, and can repeatedly execute the same configuration with different data without reloading the configuration. A configuration consists of the dataflow of where each processing element receives its data from, which address in its own register file that it writes to, and what operation it performs on the input data. Combined with the ability to specify multiple stage configurations, where for each clock cycle that the CyGRA executes, a new configuration is used, complex dataflows such as linear algebra operations or FFT are facilitated.

We have created a file that includes how to encode your own functions and common functionality and computations that we can use. For example, we have a function called mac(), this will allow anyone unfamiliar with the project to perform their own multiply and accumulate function without knowing anything about the CyGRA itself. We also have DMA, DMO, CFG, and COM commands which with the proper inputs will create custom instructions for the user to use, without little to no knowledge of the CyGRA. All these functions can be used in a user's own program to help accelerate whatever they desire.

Areas of Challenge

One of the major challenges we faced during development was the CyGRA. Initially, it was difficult to determine how to structure it as we weren't sure which functions we wanted to accelerate. Even after identifying our target functions, we struggled to find the best design approach. We quickly learned that aiming for a perfect solution from the start was

unrealistic; progress came through iteration and experimentation. Over the course of development, we revised the CyGRA design four or five times to better meet our needs and improve various features such as useability and complexity. This iterative process was essential, and we learned that the most important step is often just getting started even if the initial design isn't perfect.

This iterative process also gave us valuable opportunities to receive feedback from our advisor, Dr. Duwe. With each new design iteration we presented, he provided us with insightful feedback and pointed us towards additional resources to refine and strengthen our approach. This cycle of improvement continued until we arrived at our final design that we are very proud of.

Overall, we successfully met our client's goals by delivering a non-trivial design that enables students to engage hands-on with bringing up a new chip for the ISU ChipForge club, while also exploring advanced technologies like Coarse-Grained Reconfigurable Architecture.

Technology Considerations

The Efabless design process requires the use of specific open-source tools, with OpenLANE being the primary tool. While these tools are well-documented, they are not particularly user-friendly, which can make them challenging to use at times. Fortunately, Efabless provides scripts and configuration files that automate much of the process, allowing us to complete most tasks without needing to delve deeply into the inner workings of each individual tool.

Testing

Our overall testing plan is illustrated in Figure 11. In summary, we begin by running testbenches for each individual module using Questasim. Once these pass, we proceed to RTL simulation of the fully integrated system, where we run system-level tests using Cocotb to verify correct functionality. Next, we perform FPGA testing to confirm the design operates correctly on physical hardware; this stage also allows us to test GPIO functionality, such as interfacing with the SRAM PMOD module. Finally, we conduct gate-level tests of the complete system to ensure the chip's hardening process did not introduce any errors.

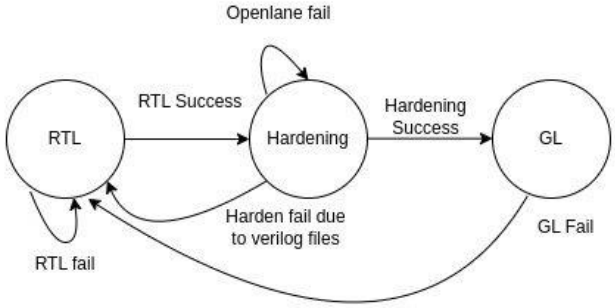

Figure 11: A state diagram showing the testing flow we used during the project

Unit Testing

- PicoRV32
  - Tested with QuestaSim testbenches
  - Verified reading and executing instructions and programs in RTL simulation
    - Working merge and bubble-sort
- CyGRA
  - Tested with QuestaSim
    - Validate custom instruction execution
    - Test each CyGRA component independently
    - Test entire integrated unit
- Internal Memory
  - Testing using Questa Sim
    - Verify direct read/write operations
    - Verify read/write operations through memory interface
  - Tested instruction fetching and memory reading and writing in RTL simulation

Interface Testing

- Wishbone Slave
  - QuestaSim
    - Writing memory interface instructions
  - RTL simulation
    - Writing a series of memory interface instructions (PicoRV32 instruction memory) from management area.
- Memory Interface
  - QuestaSim

        - Reading/Writing memory into simulated external (non-SPI) and internal memory.
        - The above tested using Wishbone Slave and PicoRV32.
    - RTL simulation
        - Writing instruction memory through the wishbone bus.
        - Reading and Writing memory from external memory using simulated Verilog module of SRAM.
        - Cache operations for all scenarios (Write, Read Miss, and Read Hit)
- SPI Main
    - QuestaSim
        - Reading and writing from simulated SPI Slave.
        - Reading and writing instructions through memory interface.
    - RTL simulation
        - Reading memory based on signals from Memory Interface and simulated SRAM.

Integration Testing

- Wishbone Slave, Memory Interface, SPI Main, External Memory.
    - Path while writing instruction memory from management area.
    - QuestaSim
        - Fully tested writing memory via simulated Wishbone inputs.
    - RTL simulation
        - Fully tested writing instruction memory via Wishbone.
- PicoRV32, Memory Interface, Internal Memory, External Memory, SPI Main.
    - Path involved with the execution of non-custom instructions.
    - QuestaSim (excluding SPI Main memory)
        - Fully tested reading/writing memory and non-memory instructions
    - RTL simulation
        - Tested several example C programs that cover all RISCV32I instructions
- CyGRA, PicoRV32, Memory Interface, Internal Memory, External Memory, SPI Main.
    - Path involved with the execution of custom instructions.
    - RTL simulation
        - Tested MAC and FFT programs using custom instructions

System Testing

- RTL Simulation
    - Tested Mac and FFT programs using custom instructions
- FPGA Testing
    - Running embench-iot benchmarks, we modified them for our system so that the benchmarks will run as standalone files, removed all dependencies to libraries and other things that is not on the FPGA, and created 2 variants for each benchmark, one that is standard and another that has our CyGRA being utilized and executing.

Regression Testing

Our regression testing framework ensures that changes to the project do not unintentionally alter the expected behavior. We leverage CI/CD pipelines for each repository to automatically run regression testing whenever changes are committed. The pipeline executes relevant testbenches for all files, and if any test fails, the issue is flagged in the pipeline and team members are notified via email to prevent further regression. When a change in behavior is intentional, we document it clearly and update the corresponding testbenches to align with the new requirements.

Acceptance Testing

Our acceptance testing ensured that we met the key performance metrics defined earlier, which are critical to verifying that the CyGRA delivers the expected impact. After hardening our design, we will validate that the user area passes Efabless's precheck process, confirming that the physical layout matches the Verilog design. The Efabless precheck also runs DRC and LVS tests to identify any potential fabrication issues, ensuring the design is ready for production.

Results

The results from our tests primarily take the form of waveform outputs, which provide a detailed view of signal behavior. These waveforms were carefully analyzed to verify that the design meets our defined performance and functional metrics, such as correct instruction execution, timing accuracy, and data integrity. Beyond initial validation, waveform comparisons also played a key role in our regression testing framework. By using waveform-based verification, we could see that any changes to the design do not unintentionally later the expected behavior. Automated tools (like CI/CD) are employed,

when possible, to check results of testbenches, allowing us to quickly identify issues and address them before they escalate.

Since the CyGRA took much more time than anticipated to design. We didn't have much time to test its functionality. Currently, we know it works properly for single configuration instructions but are unsure of instructions with more than one configuration. We also haven't had time to do FPGA or GL tests, which are important to verify if our fabricated chip would function properly.

Implementation

We have currently built a product that is able to execute custom instructions as described in our final design sections. We have also synthesized a layout via OpenLANE that could theoretically be fabricated, although the project is not in a state where it should be synthesized due to the lack of FPGA and GL tests. Hopefully, these tests can be performed in the future to verify the functionality of our project so it can be synthesized.

Design Analysis

The design is currently able to run all C programs that don't contain any custom instructions. This has been verified with many C programs. Our CyGRA can execute all single config instructions and some multi-config instructions. We have arrived at this conclusion based on the current testing we have done (MAC and FFT).

Ethics and Professional Responsibility

Our team defines engineering ethics as a commitment to uphold the integrity, transparency, and accountability in all aspects of our project. For our project, professional responsibility extends to producing open-source hardware and software that contributes positively to the academic and engineering community. Throughout the project, we have aimed to act in good faith, communicate openly, and ensure that all decisions we have made are backed by solid engineering principles. Our overarching philosophy is that engineering education should be accessible, ethical, and contribute to the betterment of the broader community.

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

Table 5: Code of Ethics

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

Table 6 6: 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 in 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 being 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 anyone 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

Summary of Progress

Overall, we have successfully completed the majority of milestones set for our project. Although we encountered some setbacks, such as Efabless shutting down, these challenges had minimal long-term impact on our progress. Over the course of the project, we were able to maintain on track and ensure that the project reached a near-final state by the end of the semester.

Over the course of the year, our team became proficient with the Efabless platform and the suite of open-source tooling required to harden and prepare our design for fabrication. We carefully planned and executed the implementation of the chip, integrating the CyGRA to serve as an instruction acceleration unit. In addition, we carried out extensive testing, both in simulation and hardware, to validate the functionality and performance of our design.

Since Efabless shutdown mid project, we were able to take advantage of the extra time to refine our testing procedures to ensure they met a high standard of reliability and thoroughness. We also used this time to further optimize the performance and complexity of the CyGRA unit.

While there is no confirmed plan for our chip to be fabricated, we have ensured that our design is hardened and ready should the opportunity arise in the future.

Value Provided

Our current chip design offers a valuable learning experience for users at all skill levels. With great depth in both hardware, software, and system bring up, this project allows ChipForge and hardware students to gain hands-on experience with some exciting technologies like Coarse-Grained Reconfigurable Architecture, while also deepening their knowledge of processor architecture and ISA extensions, enabled by the CyGRA's integration.

In addition, members of the ISU ChipForge club have the unique opportunity to work on bringing our design to life, should the chip be fabricated. This experience would allow students to get hands on with a new platform they don't have any background on. This will allow them to gain advanced problem-solving skills and broaden their technical skillset. For more advanced members of the club, they can use this chip as an opportunity to learn about reconfigurable architecture and further expand on it in the future as all of our designs are open source.

Next Steps

Although there are currently no plans for this chip to be fabricated during our time working on it, our design leaves room for future improvements and innovations. One key area for enhancement is the CyGRA; by designing a larger and more space-efficient version, future teams/ChipForge could expand on both the performance and capabilities, such as enabling support for more complex instructions or a broader acceleration use case.

We would also like to see GL tests and FPGA tests to confirm the functionality of our design and verify that it is ready to be fabricated. Additionally, we would like to see formal benchmarks run on our design to get an idea of the performance improvements provided by our design and to ensure the CyGRA functions fully as intended.

Ultimately, the most impactful advancement we hope to see in the future is the ISU ChipForge club fully fabricating this chip. Seeing our design and the hard work we have put into it coming to life would be deeply rewarded to use as a team, as well as offering students and the club an invaluable opportunity to engage with a real-world chip bring-up process.

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 -PCPI interface	-CSR instructions -Instruction-fetch fence	-Multiplication -Wishbone ready	-Linux -Privilege levels -
Size optimization	No	Yes	No	No	No

Table 6: RISC-V Core Decision Matrix

Operation Manual

Our operation manual for the CyGRA can be found at the following link (access controlled):

https://iowastate.sharepoint.com/:w:/s/ISUChipFab/Ea79wAZdrZdDgTErzOgcYBsBxC84tl5nymkmQ5gZ1IjFbg?e=WBZLoA

Alternative/Initial Version of Design

- Design using an FPGA
  - Too large to implement chip
- Design without External Memory
  - Initial plan

    - External Memory added to increase memory space
    - Internal memory became cache memory
    - Required the implementation of the memory interface

Relevant code

- DFFRAM GitHub repository: https://github.com/efabless/OL-DFFRAM
- Gitlab group: https://git.ece.iastate.edu/sdmay25-28
- Processor repositories:
    - Caravel project (also contains make_firmware script): https://git.ece.iastate.edu/sd/sdmay25-28/tree/final
    - CyGRA: https://git.ece.iastate.edu/sdmay25-28/accelerator
    - Pico Processor: https://git.ece.iastate.edu/sdmay25-28/pico_processor
    - Memory controller: https://git.ece.iastate.edu/sdmay25-28/memory_controller
    - SPI Unit: https://git.ece.iastate.edu/sdmay25-28/spi_master
- FPGA Test repo: https://git.ece.iastate.edu/sdmay25-28/caravel_fpga_tests
- November test chip: https://git.ece.iastate.edu/sdmay25-28/nov-chip
- Embench-iot repo: https://github.com/embench/embench-iot
- Chip Forge repo (access controlled): https://git.ece.iastate.edu/isu-chip-fab

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
- Regression testing
- Teamwork

Skills Set Covered by the Team

- Hardware Design
- Software development

- Electrical knowledge
- Embedded systems
- Agile project management
- CI/CD
- Analog and Digital VLSI
- Verilog/VHDL

Project Management Style Adopted by the Team

- Agile

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