

# Digital ASIC Fabrication

DESIGN DOCUMENT

sddec22-17

Dr. Duwe & Dr. Huang

[Dawood Ghauri](#) - Researcher / Design Workflow

[Constantine Mantas](#) - Researcher / Team Organization Leader

[Soma Szabo](#) - Researcher / Component Design

[Courtney Violet](#) - Researcher / Testing

[sddec22-17@iastate.edu](mailto:sddec22-17@iastate.edu)

<https://sddec22-17.sd.ece.iastate.edu>

Revised: 12/12/2022

# Executive Summary

## Development Standards & Practices Used

- IEEE Standard VITAL ASIC (Application Specific Integrated Circuit) Modeling Specification - We are designing an ASIC using a modeling standard so this will allow us to more clearly communicate our design to other engineers.
- IEEE Standard Testability Method for Embedded Core-based Integrated Circuits - We are designing an ASIC using a testing standard so this will allow us to more accurately verify our testing process.
- IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA)- This is useful for our ASIC design because it covers ways for integrated circuit designers to analyze chip timing and power consistently across a broad set of electric design automation (EDA) applications.

## Summary of Requirements

- The design is selected and manufactured by eFabless in their Open MPW shuttle program.
- The digital design is able to take in a block header and calculate the next hash of a block by testing various nonce values to compute a valid hash.
- The fabricated ASIC is able to perform the expected tasks as compared to the simulated pre-silicon design.

## Applicable Courses from Iowa State University Curriculum

- EE 330: Integrated Electronics
- CPRE 381: Computer Organization and Design
- CPRE 288: Embedded Systems I

- CPRE 480: Graphics Processing and Architecture

## New Skills/Knowledge acquired that was not taught in courses

- Researching use of new design software using github and community communication channels such as slack.
- The ability to troubleshoot problems whose solutions are much more complex than a simple google search.
- Utilizing Openlane to harden a digital design which allows for place and route of the design to be done digitally and a schematic of the design to be output.
- Utilizing Open-Source hardware development software which often relies on an interplay of numerous libraries and lacks adequate documentation.
- Creating and using a linux docker to run a local dev environment.
- Creating a plan for pre and post-silicon bring-up that highlights how the chip will be tested and debugged.
- Learning hardware design skills and languages such as Verilog and SystemVerilog to create our design.

# Table of Contents

<b>1 Team</b>	11
1.1 Team Members	11
1.2 Required Skill Sets for Your Project	11
1.3 Skill Sets covered by the Team	11
1.4 Project Management Style Adopted by the team	11
1.5 Initial Project Management Roles	12
<b>2 Introduction</b>	12
2.1 Problem Statement	12
2.2 Requirements & Constraints	12
2.3 Engineering Standards	13
2.4 Intended Users and Uses	14
<b>3 Project Plan</b>	14
3.1 Project Management/Tracking Procedures	14
3.2 Task Decomposition	14
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria	15
3.4 Project Timeline/Schedule	17
3.5 Risks And Risk Management/Mitigation	19
3.6 Personnel Effort Requirements	19
3.7 Other Resource Requirements	20
<b>4 Design</b>	21
4.1 Design Context	21
4.1.1 Broader Context	21
4.1.2 User Needs	21
4.1.3 Prior Work/Solutions	22
4.1.4 Technical Complexity	22
4.2 Design Exploration	23
4.2.1 Design Decisions	23

4.3 Proposed Design	23
4.3.1 Design Visual and Description	23
4.3.2 Functionality	26
4.4 Technology Considerations	26
4.5 Design Analysis	27
4.6 Design Plan	27
5 Testing	29
5.1 Testing Process Background	29
5.2 Interface Testing	30
5.3 Integration Testing	30
5.4 System Testing	31
5.5 Acceptance Testing	31
5.6 Security Testing	31
5.7 Results	32
6 Implementation	32
6.1 Original Design	32
6.2 Final Design	33
7 Professionalism	37
7.1 Areas of Responsibility	37
7.2 Project Specific Professional Responsibility Areas	42
7.3 Most Applicable Professional Responsibility Area	43
8 Closing Material	43
8.1 Discussion	43
8.2 Changes Since Cpre 491	44
8.3 Conclusion	44
8.4 Appendices	45
8.4.1 Team Contract	45
8.4.2 Alternative Design	50

8.4.3 Github Link (Full Code Repository)	50
8.4.4 User Manual	50
8.4.5 References	55

## List of figures/tables/symbols/definitions

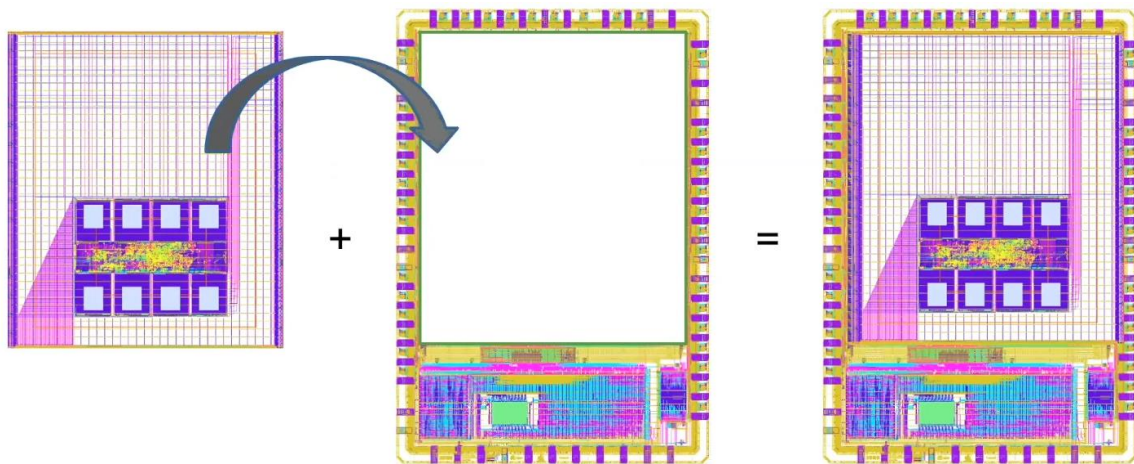


Figure 1. Caravel Process Image. Source: [6]

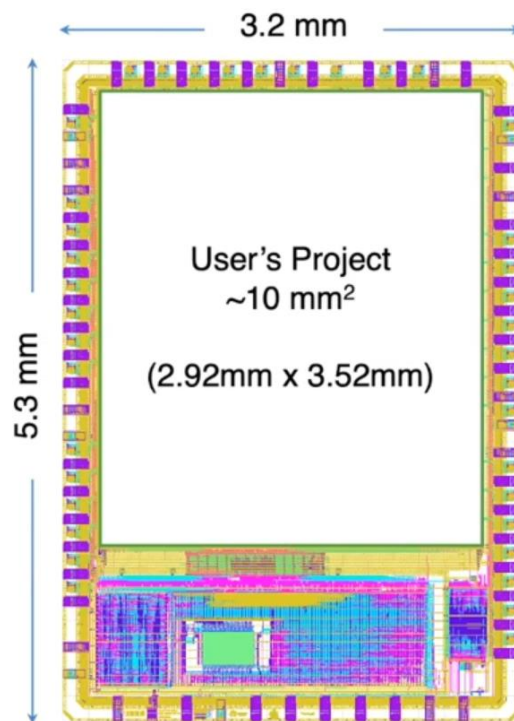


Figure 2. Caravel Harness. Source: [6]

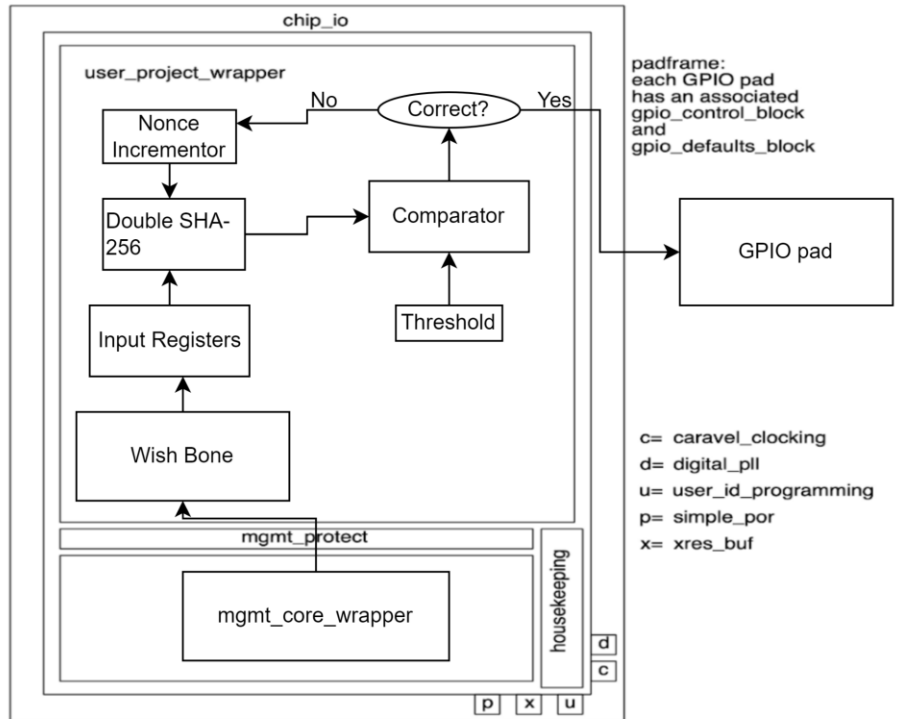


Figure 3. Initial Design Sketch. Source: Adapted from [6]



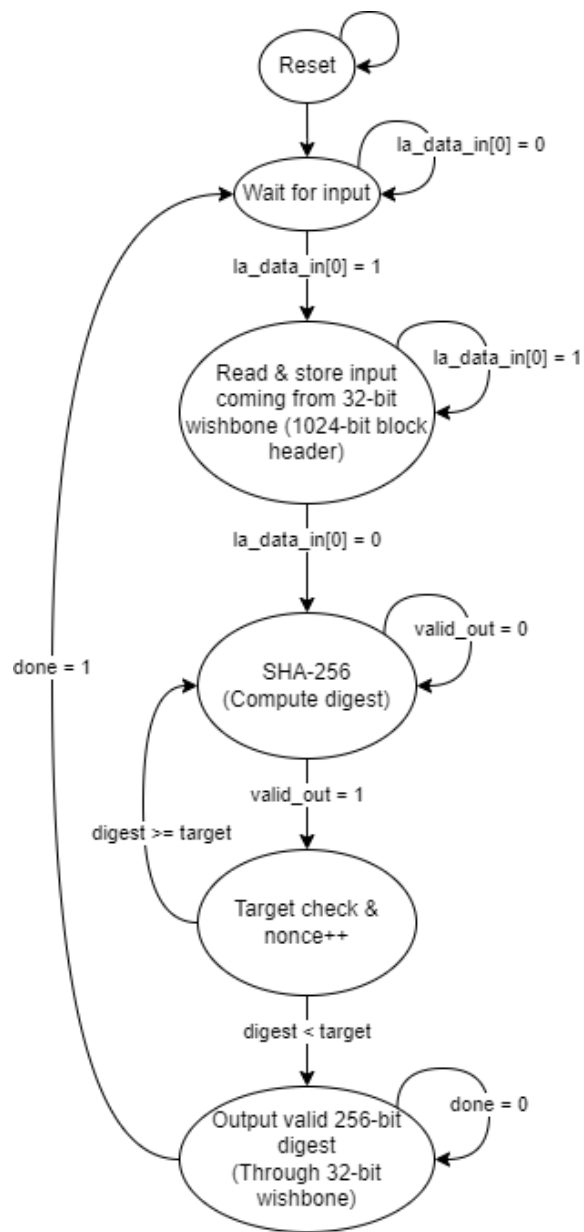


Figure 4. Initial State Machine Design

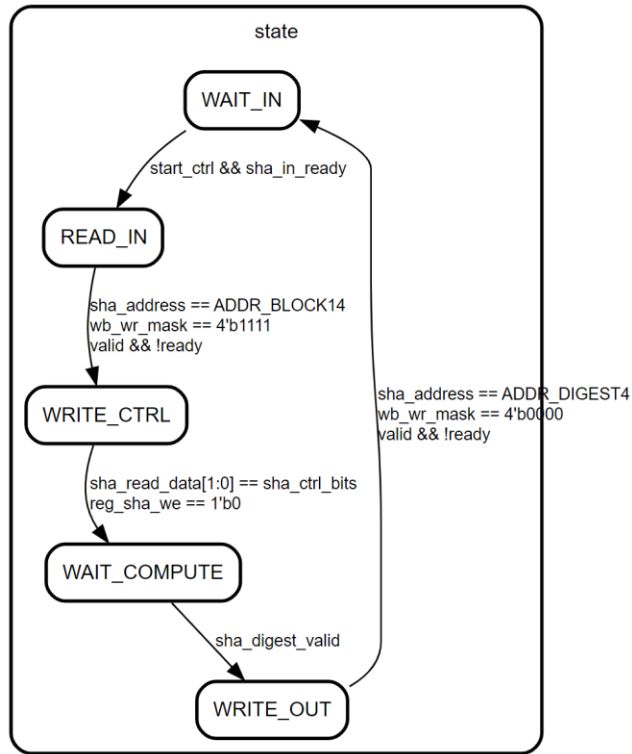


Figure 5. Final State Machine Design

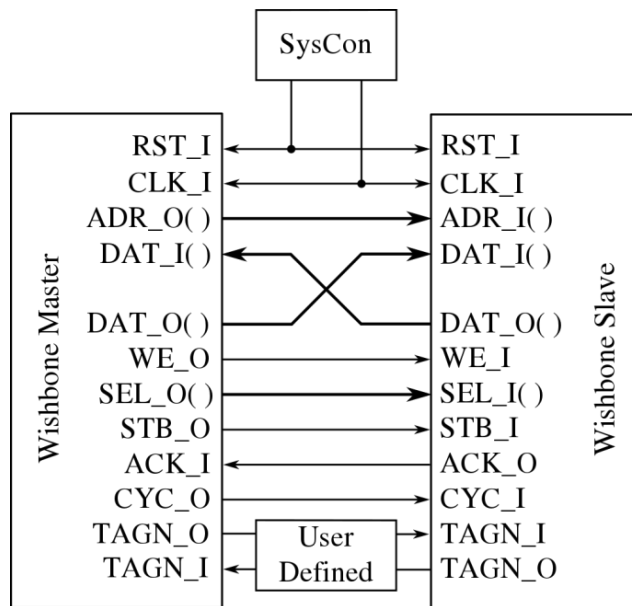


Figure 6. Wishbone Model. Source: [7]

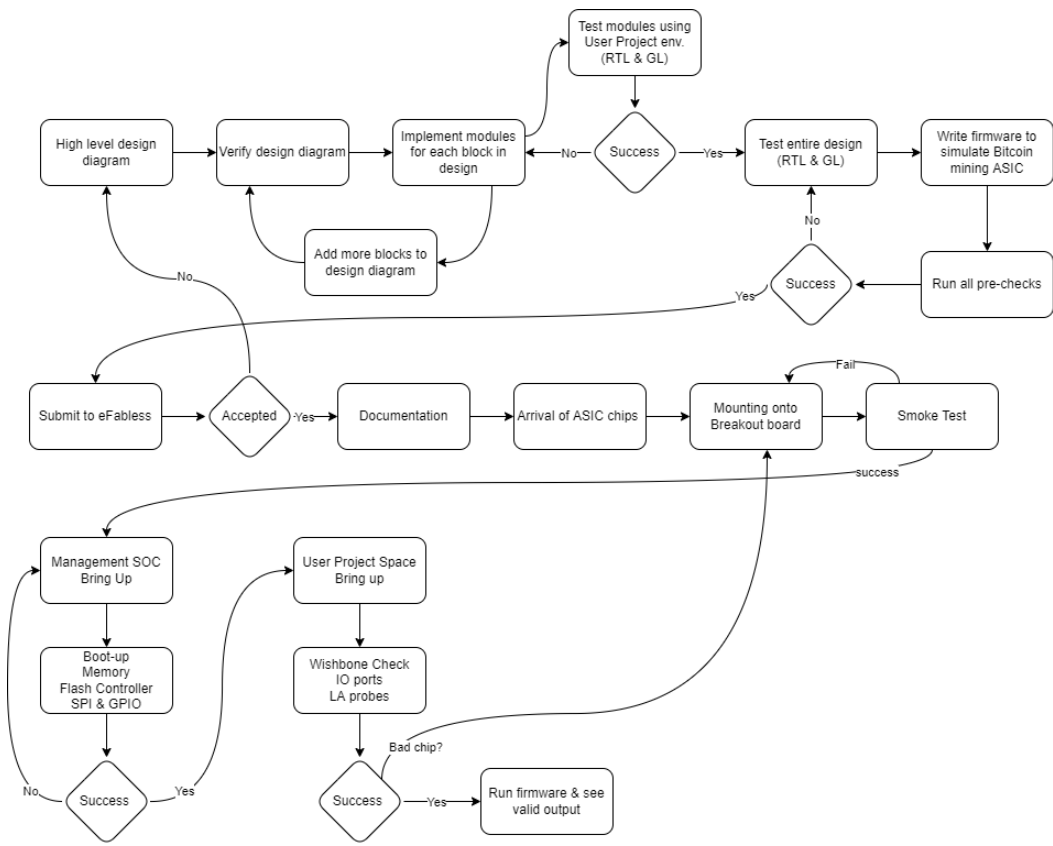


Figure 7. Bring-up Plan

**Ball assignment (6x10 WLCSP)**

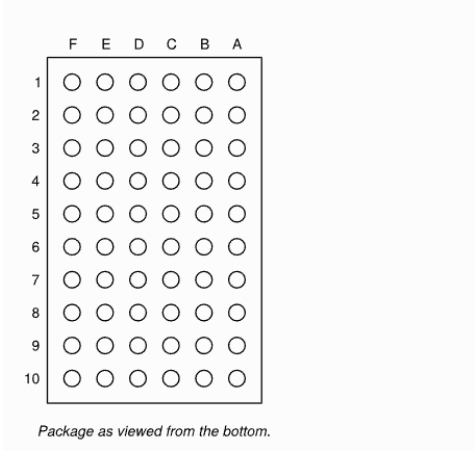


Figure 8. Image of Pinout for ASIC. Source: [8]

# 1 Team

## 1.1 TEAM MEMBERS

[Constantine Mantas](#) - a Computer Engineering major with an interest in hardware design. Through various classwork, internships, and projects, Constantine has gotten experience with various programming languages like Java, C, and Python, and computer hardware design with technologies like ModelSim, Verilog, and VHDL

[Soma Szabo](#) - a Computer Engineering major, pursuing the Master of Engineering concurrent degree program. His classes and work experiences have provided him with knowledge in programming techniques, software languages such as Java, TCL, C/C++, Python, and Dart, as well as hardware description languages such as VHDL and Verilog. He also has experience with computer architecture and related tools such as ModelSim, Vivado, Cadence, and Altium Designer (PCB design).

[Courtney Violet](#) - Is a Computer Engineering major. He has worked through a variety of class work and projects, both for class and personal projects, has gained experience in programming languages such as C, Java, HTML, CSS, computer hardware design, and circuit design. He also has experience with tools such as android studio and ModelSim.

[Dawood Ghauri](#) - a Computer Engineering major with a minor in Physics. His class experiences have developed his knowledge with respect to programming experiences in C, C++, Java, Python, VHDL, and Verilog

## 1.2 REQUIRED SKILL SETS FOR YOUR PROJECT

Some of the skill sets required for this project are:

- Ability to use and debug Verilog/SystemVerilog
- Ability to set up development environment in Linux
- Ability to use C to drive testing for Verilog/SystemVerilog components
- Understand the fabrication process for an ASIC

## 1.3 SKILL SETS COVERED BY THE TEAM

All of the required skills are covered by each team member.

## 1.4 PROJECT MANAGEMENT STYLE ADOPTED BY THE TEAM

The Project management style that we will use for this project is an agile style. We began the project with a waterfall approach when doing the research, but now that we have begun integration of components into the project framework we have begun using an agile methodology where each week we go over tasks that have been completed from the previous week, and then discuss tasks to complete for that week.

### 1.5 INITIAL PROJECT MANAGEMENT ROLES

Role	Team Member
Team Organization	Constantine Mantas
Design Workflow	Dawood Ghauri
Individual Component Design	Soma Szabo
Testing	Courtney Violett

## 2 Introduction

### 2.1 PROBLEM STATEMENT

Our group is designing a process for future students to fabricate a microchip through the Open MPW Shuttle program. To do this, we are creating a bring-up and testing plan for chip fabrication that will provide a streamline process for fabricating an application specific integrated circuit (ASIC). The focus of our design will be to create a test bed for evaluating hardware accelerated hashing implementations and act as pioneers using Open MPW for future students wishing to create custom ASICs.

### 2.2 REQUIREMENTS & CONSTRAINTS

Functional Requirements (Specification):

- Full Chip Simulation passes for RTL and GL (gate-level)
- The hardened Macros are LVS and DRC clean
- The project contains a gate-level netlist for user\_project\_wrapper at verilog/gl/user\_project\_wrapper.v

- The hardened user\_project\_wrapper adheres to the same pin order specified at pin\_order
- The hardened user\_project\_wrapper adheres to the fixed wrapper configuration specified at fixed\_wrapper\_cfgs
- XOR check passes with zero total difference
- The design passes the mpw-precheck

#### Resource Requirements:

- The project repo adheres to the same directory structure in this repo.
- The project repo contains info.yaml at the project root.
- Top level macro is named user\_project\_wrapper.
- Openlane summary reports are retained under ./signoff/

#### Qualitative Aesthetics Requirements:

- Not relevant for our project.

#### Economic/Market Requirements:

- Ensuring all technologies used are open-source. No monetary gain from design.

#### Environmental Requirements:

- As we are not in control of the fabrication process itself (managed by eFabless), there are not any environmental requirements relevant to us.

#### UI Requirements:

- Our design will not have a UI; therefore, UI requirements are not applicable.

### 2.3 ENGINEERING STANDARDS

IEEE Standard VITAL ASIC (Application Specific Integrated Circuit) Modeling Specification - We are designing an ASIC using a modeling standard so this will allow us to more clearly communicate our design to other engineers.

IEEE Standard Testability Method for Embedded Core-based Integrated Circuits - We are designing an ASIC using a testing standard so this will allow us to more accurately verify our testing process.

IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA)- This is useful for our ASIC design because it covers ways for integrated circuit designers to analyze chip timing and power consistently across a broad set of electric design automation (EDA) applications.

## 2.4 INTENDED USERS AND USES

The primary benefactor of the results of our project will be our customers and specifically our advisor and customer Prof. Duwe. His hope is to use the knowledge and example of our project to be able to continuously have teams developing microchips through Efabless.

Another major benefactor of our project will be the preceding team to ours as they will not only use our project as an example for their own project. They will also benefit from our in depth documentation on how to develop within Efabless, but also on our documentation of the bring up process that they will be performing on our chip.

# 3 Project Plan

## 3.1 PROJECT MANAGEMENT/TRACKING PROCEDURES

As we were becoming familiar with the provided design environment, we initially worked in a waterfall workflow. However, once we became familiar with building the environment, we worked in an agile workflow. We started in a waterfall management style as it has allowed us to become more familiar with how to develop within Efabless. Once we all become familiar with it, an agile management style will be easier to work in. This will allow us to quickly move through and test various components as we implement and integrate them into the larger project.

What will your group use to track progress throughout the course of this and the next semester. This could include Git, Github, Trello, Slack or any other tools helpful in project management.

We are currently tracking tasks using a Microsoft Teams task board and working collaboratively on our codebase using Github.

What will your group use to track progress throughout the course of this and the next semester. This could include Git, Github, Trello, Slack or any other tools helpful in project management.

## 3.2 TASK DECOMPOSITION

Design an Open MPW ASIC submission

1. Setup of eFabless dev environment
  - a. Setup Github repo
  - b. Clone caravel\_user\_project template into repo
  - c. Setup local linux environment to build project
  - d. Test baseline/template project to ensure everything works

2. Plan what we will design
  - a. High level system design
  - b. Functionality design for each component
3. Code components
  - a. Verilog development of each component used in the high level design
4. Test components
  - a. Testbenches for each individual component to verify functionality
  - b. Potential component specific feedback stage
5. Bring components together into a system
  - a. Put components together in the larger wrapper to create a final system
  - b. Test integrated components within the system to verify they are communicating correctly.
6. Test the system
  - a. Create testing that will cover all critical areas and ensure functionality of the system
7. Feedback from Dr. Duwe
  - a. Ensure the system and components meet expectations.
  - b. Revise components, testing, or other aspects and repeat steps 2-6 (based on additions/revision)
8. Submit design to eFabless Open MPW shuttle

### 3.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

The main criteria for a successful project is our goal to have a successful candidate for the Open MPW Shuttle in order to enter the lottery selection process, where our design may be selected for fabrication. Getting to this point will demonstrate that we have gone through the full process that leads to asic fabrication.

This requires us to pass precheck and tapeout as defined by Efabless.

The precheck requirements are as follows.



- Open Source Github Repository
- Checks:
  - LICENSE & README
  - YAML file
  - Consistency Checks (Logical Data Propagation)
  - DRC (Design Rule Checking) & LVS (Layout vs. Schematic) checks on the user project
  - XOR check
  - Chip passes RTL (register transfer language) and GL (gate level) simulations
  - Gate-level netlist (textual description of components) & hardened user project wrapper successfully generated
- Local precheck
  - Fast and readily available method
- Efabless website precheck
  - The official submission precheck which is much slower and more detailed

In order to pass tapeout our design must be able to harden using the openlane software which simulates the full transistor layout of our design.

### 3.4 PROJECT TIMELINE/SCHEDULE



**Digital ASIC  
Fabrication**

## GANTT CHART

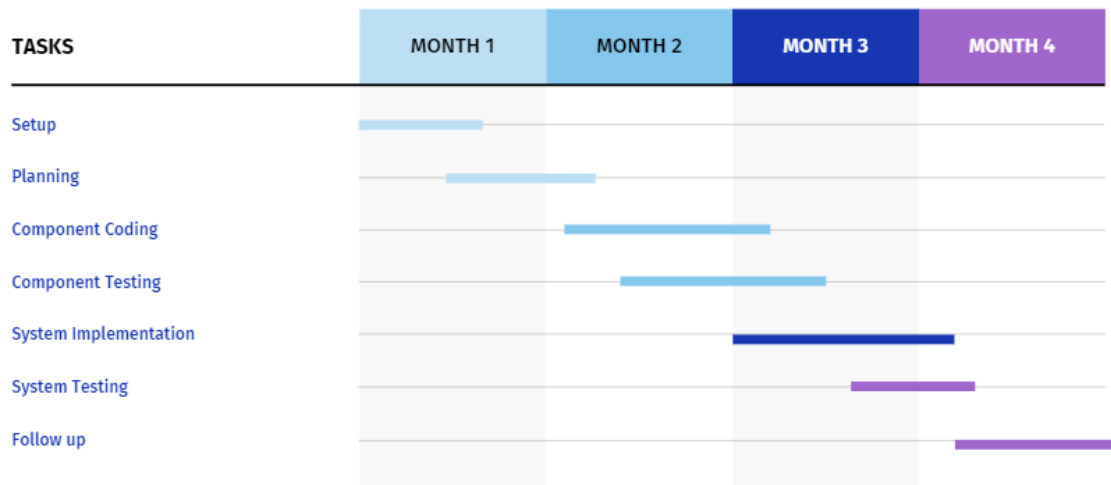


Figure 9. Project Timeline GANTT Chart

Week starting from 3/21/22	Task
1	High level design
2	Begin component design
3	Component design
4	Finish component design
5	Begin component coding
6	Component coding

7 (end of 491)	Component coding
8 (beginning of 492)	Component coding
9	Feedback from Dr. Duwe (additional component design)
10	Component coding (if needed) & testing
11	Component coding (if needed) & testing
12	Component coding (if needed) & testing
13	Component testing
14	Begin bringing components together into a system
15	Bringing components together into a system
16	Finish bringing components together into a system
17	Testing system
18	Testing system
19	Feedback from Dr. Duwe (provide any additional testing/implementation)
20	Testing system & additional implementation (if needed)
21	Testing system & additional implementation (if needed)

22	Testing system & additional implementation (if needed)
23	Finish entire system

### 3.5 RISKS AND RISK MANAGEMENT/MITIGATION

Agile projects can associate risks and risk mitigation with each sprint.

1. Setup of eFables dev environment - difficulty setting up and installing workstations .6

We have already come into considerable roadblocks when setting up our dev environments as the information for setting it up is scattered and inconsistent. There is no way to eliminate this task as it is critical to do anything for our project. Although there are development tools available from eFables these tools aren't great for development. There aren't any other alternatives for this task.

2. Plan what we will design - .3
3. Code components - .75

There is no way to eliminate this as we need to come up with code for the project. We are able to find some off the shelf solutions as there are free IP's available with coded components. There aren't really any other alternatives as all the components need to be coded in Verilog.

4. Test components - .5

There is no way to eliminate this task as we need to verify that our components are working correctly. There aren't any ways to get any off the shelf tests as our components will be of our own making. There isn't an alternative solution to this either. This is a critical part of hardware design that must be handled to ensure the fabricated hardware will work as expected since there is no way of modifying the chip afterwards.

5. Bring components together into a system - .4
6. Test the system - .2

### 3.6 PERSONNEL EFFORT REQUIREMENTS

Task	Total Estimated Person hours
------	------------------------------

Setup dev environments	10 per person
Plan our overall design and each component	20
Code Components	70
Test Components	70
Bring components together into system	70
Test System	70

Setting up a dev environment includes us familiarizing ourselves with the eFabless ASIC development process. Setting up a Github repository that makes use of an eFabless template project. Setting up a linux environment that has all the tools to run through all of the checks and tests required for the project.

Planning the design will include a high level schematic of what components we need to make. Schematics for how each component will be implemented with simpler logic modules.

Coding the components will entail designing Verilog files for every subcomponent.

Testing each of these subcomponents to ensure proper individual functionality is the initial testing components phase.

Bringing the components together into a system entails bringing all subcomponents together to test actual component functionality then hooking all components together to verify system functionality.

Finally testing incremental functionality of the system will ensure that once everything is put together our final product works as intended.

### 3.7 OTHER RESOURCE REQUIREMENTS

None, as this is a purely digital project. However, proceeding teams will need tools such as wave form analyzers and generators to be able to test the proceeding teams designs.

## 4 Design

### 4.1 DESIGN CONTEXT

#### 4.1.1 Broader Context

The communities that are being designed for are primarily students that will be using our project as an example, and for our clients/professors that will be seeing the validity of having this as future senior designs. It also will address the safety and welfare of the teams that will be coming after us as we will be making sure that their health and safety are being protected.

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

Area	Description	Examples	Considerations
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., solution is implemented in their communities)	Increasing/reducing exposure to pollutants and other harmful substances, increasing/reducing safety risks, increasing/reducing job opportunities	During the bring up process there is risk of injury if one of them components serious malfunctions. Reducing those risks is a priority.
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.	Development or operation of the solution would violate a profession's code of ethics, implementation of the solution would require an undesired change in community practices	This process we aim to develop will make it far more feasible for groups of students to develop and fabricate their own ASICs which will have long lasting benefits to the future of ASIC design.
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.	Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development cost creates risk for organization	It may lead to more efficient ASIC design in the future which could lead to many more ASIC projects entering the market and shaking the technological field.

#### 4.1.2 User Needs

Students need a way to understand how to create an ASIC and get it fabricated by eFabless, because as it stands now there isn't much cohesive documentation on how to do so.

Prof/client needs a way to demonstrate how to have students fabricate their own ASICs, because they want to understand the feasibility of having a senior design group every year making their own ASIC's.

#### 4.1.3 Prior Work/Solutions

Because most of our work for this project is focused around the complicated nature of fabricating an ASIC we plan on finding publicly available VHDL IP that we can use in our design. We plan to make custom parts in the design where time allows but most of the uniqueness of our design will come from how the design interacts with the chip harness. Our highest priority is having a functional ASIC design submission that can be selected with as much uniqueness to design as we can fit in our time frame.

Unique components we plan on designing for example is an accelerated adder design that includes a SHA-1 hash.

Advantages

- More space constrained to meet the requirements
- Application specific design for lower power consumption

Disadvantages

- Simple design model may make us less likely candidates in the selection process

#### 4.1.4 Technical Complexity

The design for a SHA-1 ASIC requires us to build all auxiliary components and correctly integrate our accelerated SHA-1 hasher into the caravel harness. The application of the algorithm is explained in the cited research above.

Finally the compressor will be composed of 4 types of computation circuits that are all performing various levels of bit addition in order to complete the SHA-1 hash.

The most complicated part of all of this however will be integrating the design within the eFables caravel harness in order to submit a design that can actually be fabricated into an ASIC. This requires multiple levels of testing and port alignment to make sure that the design is fully functional within the harness.

## 4.2 DESIGN EXPLORATION

### 4.2.1 Design Decisions

One key design decision that has been made is to not include a random number generator on the design as we have a limited amount of chip space and would add an unnecessary layer of complexity to our design. Another design decision that had to be made was the type of adder. We decided to go with the carry lookahead adder in our original design as this type of adder was shown through the synthesized Verilog to be the least expensive in terms of space. Hardening and precheck requirements were passed and feasibility/efficiency was determined to be met in our local environments. However, with the changes in the eFabless process, the area constraints were altered to our detriment. As such, we had to alter our design to use a SHA-1 IP instead which utilized a pre-synthesized carry lookahead adder instead of our custom implementation. This resulted in us changing the overall scope of the design implementation from a Bitcoin mining accelerator to a hardware hashing accelerator.

Another area of the design was the layout of the components on the chip. The specific placement of the components will be handled by eFabless tools; however, there are several options when it comes to those tools. Once we implemented the components of our design, we determined the best ones that fit into our design.

## 4.3 PROPOSED DESIGN

### 4.3.1 Design Visual and Description

Below is our high level design of the cryptographic hash function (SHA-1). This is based on the research paper of the original SHA-256 hasher. It consists of a SHA-1 hash unit used to compute the new hash of the block and output the valid 160-bit hash to the management SoC. Otherwise, 1 will be added to the nonce and the hash will be recalculated.



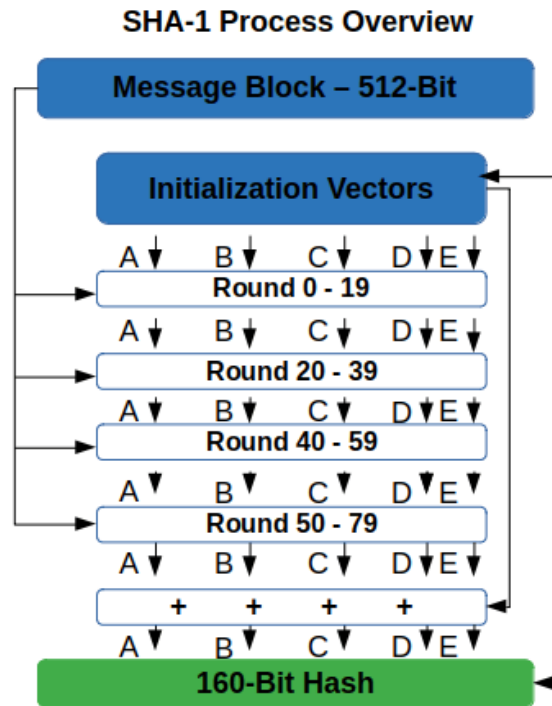


Figure 10. SHA-1 Process Overview. Source: [9]

To store the block header (512-bit input to the SHA-1 modules) we will have 2 types of registers. Ones that are 32-bit wide to store the version, timestamp, target, and nonce along with others that are 160-bit wide to store the hash of the previous block and Merkle root. These will be populated with data coming from the management SoC through the wishbone bus. A high level state machine diagram for the SHA-1 hardware accelerator is shown below with the SHA-1 computation state and target check being broken down in the diagram above.

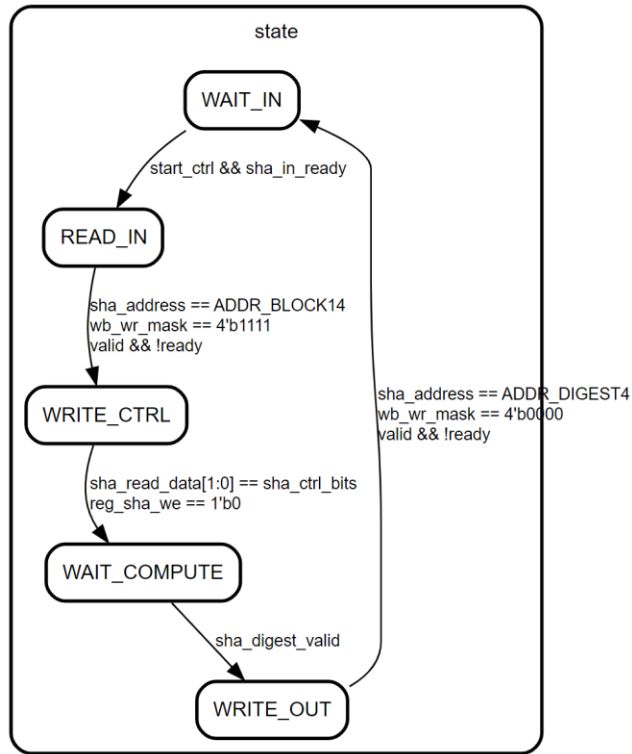


Figure 11. SHA-1 Computation State Diagram

Below is an example of how an arbitrary design (left) would be integrated into the caravel harness (middle), resulting in a complete design that can be fabricated and function as an ASIC (right).

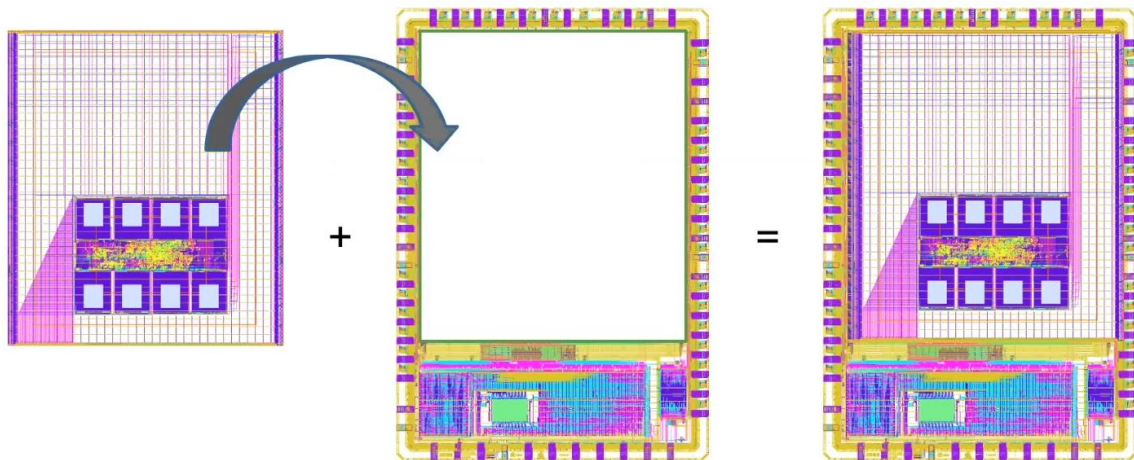


Figure 12. User Project and Caravel Harness Distinct. Source: [6]

### 4.3.2 Functionality

The design is intended to take in a previous hash input and calculate the next hash of a blockchain by running through various nonce values. The current design seems to satisfy all the functional requirements of the hashing process as demonstrated in this picture again.

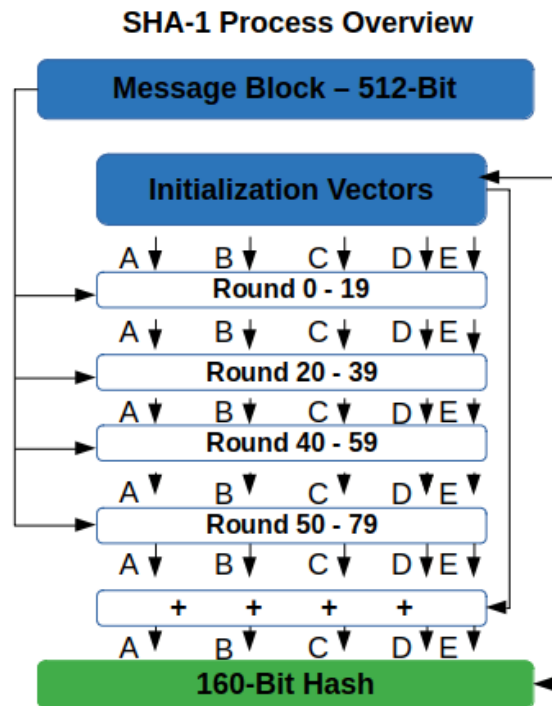


Figure 13. SHA-1 Process Overview. Source: [9]

## 4.4 TECHNOLOGY CONSIDERATIONS

There are a wide variety of hardware design tools available for digital AISC developers such as ModelSim, Vivado, Cadence, and more. These allow hardware designs to be simulated, synthesized, and run through checks such as DRC and LVS. With all this, the logic design can be translated to a physical layout that can be used to fabricate a chip. The Open MPW shuttle projects from eFabless rely on their own tools for submitting a valid chip design; thus, our team was constrained to learning, understanding, and using the provided Caravel User Project as a basis for our design. The logic and layout synthesis tools were OpenLane and the SkyWater 130 Process Design Kit was used for the standard cells in our design.

The largest tradeoff in using these tools was the learning curve and the correct setup of the local environments so everyone on the team could contribute to and test the design. Another challenge was communicating with the eFabless support teams in case of any errors. On the other hand, once the tools were correctly set up, the design process became much simpler and provided a streamline benchmark that ensured our project was acceptable for submission. Since the choice in hardware design tools were constrained, we did not have the option to consider alternatives and committed to understanding the provided software and hardware.

#### 4.5 DESIGN ANALYSIS

The SHA-1 hardware accelerator ASIC design has completed the testing and simulation stages. It is currently in the verification phase on eFabless. These testing and simulation stages include logic and gate level simulations as well as connecting logic analyzer testing points in the design to debug and bring up the chip post fabrication. To ensure a round trip of the entire design process can be completed, we ran individual subcomponents of the main design through the local environment prechecks. This confirmed that a custom design can be created and translated to a physical layout within the user project area of the chip. Therefore, we are confident that our final hardware accelerator ASIC design will be successfully synthesized and pass the verification phase on eFabless.

There are many optimizations and design choices we considered while implementing the chip. These included multiple SHA-1 hashing cores capable of simultaneously working together to compute more hashes that may be valid for the specified block. This would speed up the mining process but was found to increase the verification complexity, add more signals to bring out to the logic analyzer, and most importantly, take up too much physical space in the user project area. With the unforeseen changes in the eFabless manufacturing process, the goal shifted to create a simpler version of a single hashing core with lots of debug options to ensure it works correctly. We performed as many pre-silicon tests as possible to ensure the fabricated chip will behave as expected and will have the ability to determine what, if anything, goes wrong through proper debug functionality. If selected by eFabless, more complex iterations of our design could be created and sent for fabrication by future senior design teams.

#### 4.6 DESIGN PLAN

Our design plan began with finding a suitable IP for the SHA-1 that makes up the core of our design. Once fully tested and verified that the IP functions correctly we integrated it into the user project area and began the process of changing adders within the hasher to be able to demonstrate different adder efficiencies. As mentioned previously, with the changes to the eFabless manufacturing process within the Skywater foundry fabrication node, we were unable to follow through with this original design plan direction. We altered the design to utilize the pre-synthesized adders within the SHA-1 IP. Each component will be tested separately from the overall design to ensure that they are functioning correctly before being integrated into the user area. Once a new component is integrated into the user area the entire system will be tested again to verify that the new component has been integrated correctly. Finally once every component has

been tested and integrated into the user area the entire system will be tested to ensure that everything is working correctly within the Caravel Harness.

Once our design for the SHA-1 hashing ASIC is integrated into the Caravel Harness, it will be reset and held in a waiting for input stage until a logic analyzer signal is asserted high. This will occur when the firmware sets bit 0 of the logic analyzer to 1 and the reading of the block header will follow. This state requires bit 0 of the logic analyzer to stay high and will read 32-bit data fragments from the wishbone bus, storing the input into the corresponding register for the block header. A handshake will be performed to ensure the data was stored successfully and the next block of data is ready to be stored. Once the entire block header is stored, bit 0 of the logic analyzer must be deasserted and the SHA-1 IP core will begin computing the output hash or digest. An output valid bit will be set and the digest will be compared with the target. If the digest is less than the target, the valid output hash will be sent over the wishbone bus to the management SoC and stored in a variable. This will be compared to the expected output hash to ensure correctness. While this computation occurs in the user project area, various stages of the computation will be probed using the logic analyzer. These include the nonce to ensure it is incrementing correctly, the results of various adders within the SHA-1 module design, and the digest output. This ensures our design is functioning correctly and can debug post fabrication.

#### Module Constraints:

- I. N-bit registers
  - A. Able to read stored values as output
  - B. Able to take as input n bit values and write new values to register to store
  - C. Able to toggle write ability on and off with single bit control
  - D. Able to reset registers to stored values being set to NULL
- II. Nonce Incrementer
  - A. Takes a single bit input each cycle to determine whether or not it will increase the nonce value. Acts similar to a write enable control input.
  - B. Initialized value of 0 and each cycle it receives a 1 input it will increase the stored value by exactly 1.
  - C. Outputs the stored nonce value
  - D. Able to be reset with a reset control
- III. SHA-1 IP
  - A. Outputs a 160-bit hash value of the input where it has run through the SHA-1 hash twice. The SHA-1 definition we use is based on the National Institute of Standards

and Technology definition and citations here  
[https://csrc.nist.gov/glossary/term/sha\\_1](https://csrc.nist.gov/glossary/term/sha_1).

- B. Takes a 512-bit input.

## 5 Testing

The pre silicon bring-up of the ASIC will rely heavily on testing to ensure the design is functional. This is a portion of the requirements for our project, so we must ensure the test plan is thorough. Otherwise, the fabricated chip may behave unexpectedly without knowing the root cause. A successful bring-up process will involve lots of testing. Specifically, performing simulation tests focusing on edge-cases and analyzing C test benches and waveforms at the RTL and gate level to ensure correct functionality. A more detailed analysis of our test plan is discussed through this document.

### 5.1 TESTING PROCESS BACKGROUND

Tools used for testing our design:

- Built in logic analyzer and wishbone communication of the Caravel harness allows us to develop test cases for each unit in C code and testbenches in Verilog. Using this we can attach virtual probes to each unit and run tests on them.
  - Give input and specify expected output
- Waveform simulation output files at the RTL and gate level. These will be used to confirm functionality of each module before integrating into the design.
- OpenLane is software that will be used to test that all components fit within the limited space of the design, check that our resource allocation is within specifications of the harness, as well as show how they expect to be mapped out
- Verilator/ModelSim is also being used to test the IP for a SHA-1 hasher to determine its correctness and security.

## 5.2 INTERFACE TESTING

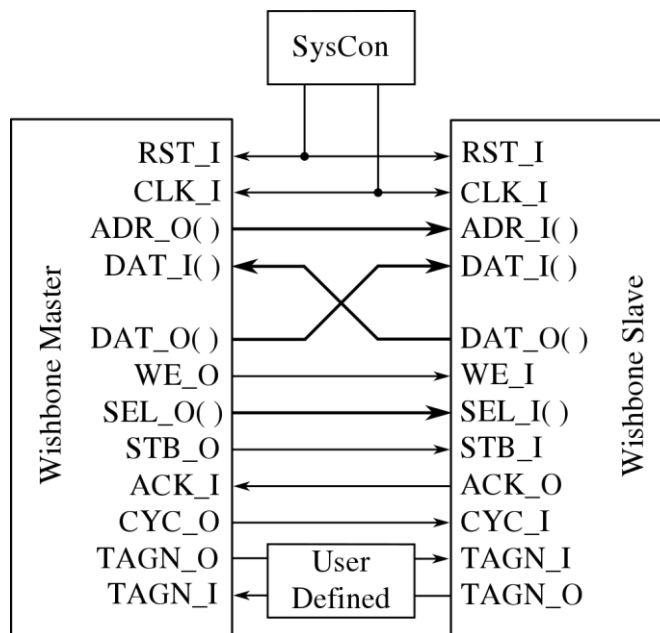


Figure 14. Wishbone Model. Source: [7]

DUT (Designs Under Test) will be specified within the testbenches written in Verilog and C on the “Wishbone Master” side (i.e., the firmware). Along with the wishbone, another interface which assists in the testing process is the Logic Analyzer. This helps to drive inputs and receive outputs from the master and slave respectively. Some important ports to consider above are the CLK\_I (the synchronous clock), DAT\_I/O (data input/output), and ACK\_I/O (acknowledge port). The clock allows the testbench (on the firmware side) to supply our project with a simulated clock, data input/output allows data communication between both the firmware and the user project area, and the acknowledge port serves as a handshake between both sides to ensure proper data communication.

## 5.3 INTEGRATION TESTING

The most critical integration path will be the SHA-1 hashing unit as it is made of many different subcomponents of adders. It is also the main functionality of the design and if it is not working then the design itself is somewhat meaningless. Another critical path will be the comparator as it will be comparing the hashed NONCE to the hashed target value. It is critical as even if we are hashing correctly if the comparator isn't working the design will also fail. These integration pathways will be tested using test cases in c that use the wishbone and logic analyzer on the management SOC of the Caravel harness to determine the values that are being generated by both integration paths. Before they are integrated together, each of the integration paths will be tested on their own to validate their individual functionality. The test benches themselves also generate

waveforms that allow us to go through and verify/debug values of different components of our design.

## 5.4 SYSTEM TESTING

System level testing will run test cases structured to look similar to expected use cases. For example, we will specify a target and a previous hash as inputs to our adder and run them through an online hashing tool to get an expected output for our miner. In our test cases we will check the outputs of each cycle of hashing. If this is functioning as expected for a sufficient number of test cases then we feel that our design will be functional. This is a critical part of our design and ties into the requirements since it revolves around the successful bring-up of an SoC.

Tools:

- Built in logic analyzer and wishbone communication of the Caravel harness allows us to develop test cases for each unit in C code. Using this we can attach virtual probes to each unit and run tests on them.
  - Give input and specify expected output

## 5.5 ACCEPTANCE TESTING

We will be demonstrating the non-functionality of our design through OpenLane. OpenLane is software used by Efabless to determine if the design is within the specifications of the Caravel harness. This test is also required to submit the design to eFabless. We will be verifying the functionality of our design through test cases and waveforms. The test cases will be in C and be using the wishbone and logic analyzer on the harness to check correct values. We will also be using waveforms that are generated during the test benches to manually verify that the correct values are being achieved. We also will be running through the test benches and waveforms with our advisor/client during the rest of the design process. The waveforms and test benches must also be submitted to Efabless to verify the functionality of our design for them to fabricate it.

## 5.6 SECURITY TESTING

The only security concern that is present in our project is with the imported IP that we are using for the SHA-1. The security risk associated with using an imported IP in the design is if there are any additional components of it that would give a third party access to it somehow, or if it would cause damage to the design when powered up. We are addressing this by going through each of the design files to determine if they are strictly necessary to what we will be using it for as well as testing the IP thoroughly before integrating it into our system. The license on the IP allows us to make changes to it so if we do find some unnecessary or malicious addition we are able to remove them. We also will be doing a basic search of the author of the design to determine if they are a



trustworthy source.

## 5.7 RESULTS

The results of our testing will come in two forms: one should be in a text reply that all of our C test cases have passed as expected, and the second will be an output of a waveform file for each component being tested that will show how every variable changes throughout the test cases. We can design our test cases based on the requirements we need to meet so that we can ensure compliance and demonstrate sufficient functionality. For example in an adder we can show test cases doing basic addition and subtraction as well as edge cases such as where we may expect overflow.

Example waveform of testing the custom adder:

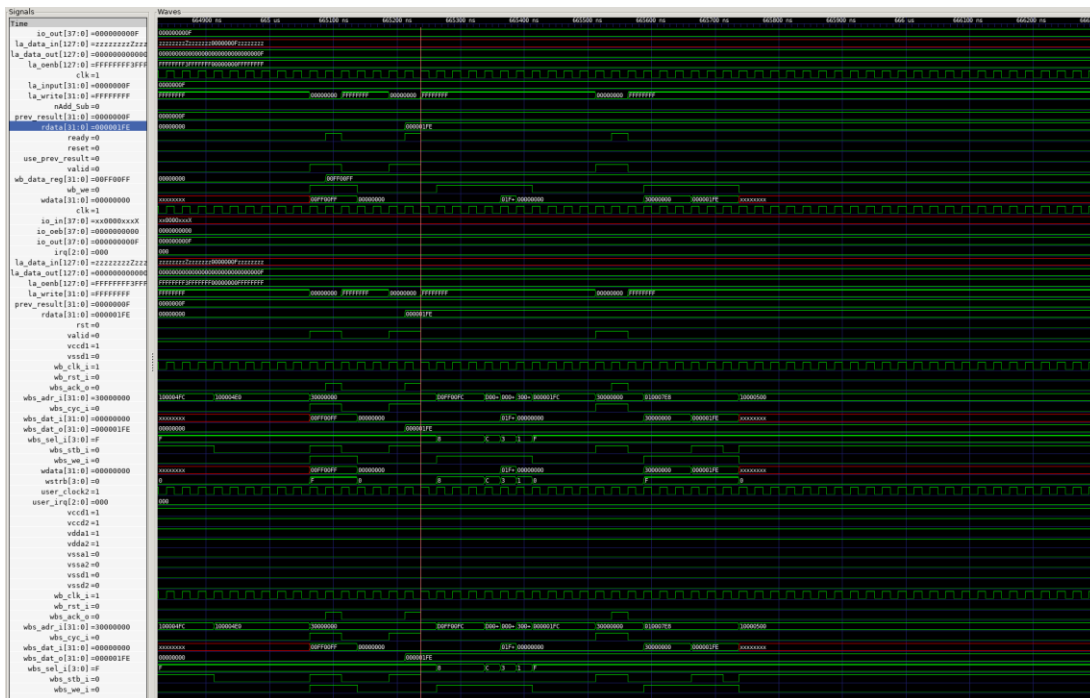


Figure 15. Custom Adder Waveform

## 6 Implementation

### 6.1 ORIGINAL DESIGN

With the original Bitcoin mining ASIC design, the plan was to further test the functionality of the IP to ensure it functioned as intended and did not present security issues. Then, we had to

integrate it into our project wrapper and harden it to ensure it is within the space constraints of the project specifications. The Verilog code and RTL simulations are complete for the finite state machine (FSM) and the SHA-256 module. Unfortunately, the hardening process could not be completed in time for the MPW-7 submission deadline on September 12th due to space and layout constraints of the user project area. Thus, an alternate smaller design quickly needed to be created to meet the requirements to be able to reach tapeout. Keeping our project similar to its intended goal while shrinking the size of the physical design considerably resulted in a SHA1 hardware accelerator module implementation which will be discussed in the next section.

## 6.2 FINAL DESIGN

The SHA1 hardware accelerated hasher is the final design that could go through the entire MPW submission process including hardening, passing RTL simulations, gate-level (GL) simulations, and passing the final precheck. To implement this, a SHA-1 hashing module IP was used and integrated into a custom FSM allowing for communication through the logic analyzer (LA) ports and the Wishbone bus. The LA bit mappings are shown in the Figure. Reference Figure 11 for the FSM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
la_sel	la_sel	la_sel	la_sel	la_sel	la_sel							sha_mode	sha_next	sha_init	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
rst	clk		auto_ctrl	start_ctrl	sha_we	sha_cs	sha256_addr	sha256_addr	sha256_addr	sha256_addr	sha256_addr	sha256_addr	sha256_addr	sha256_addr	sha256_addr
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
			idle	error	sha_we	sha_cs	sha_addr	sha_addr	sha_addr	sha_addr	sha_addr	sha_addr	sha_addr	sha_addr	sha_addr
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data	sha_read_data
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata	rdata

Figure 16. Custom Adder Waveform

With the Verilog design complete, testbenches had to be created in Verilog and C and RTL simulations had to be run to ensure correct functionality. After successful RTL simulations, the design had to be hardened which was a highly time consuming process. OpenLane had to be configured perfectly to successfully translate the high-level Verilog code to standard cells used in the physical layout of the user project area. The hardened user project area is shown below at 40µm and 2µm scales to illustrate the scale and density of the design.

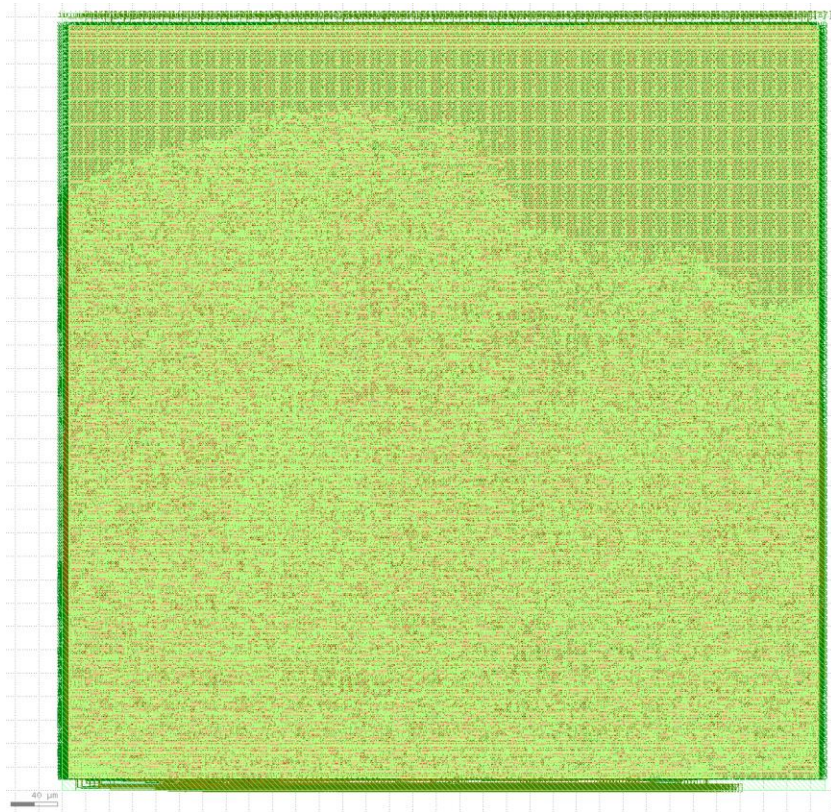


Figure 17. Hardened User Project Area (40  $\mu\text{m}$ )

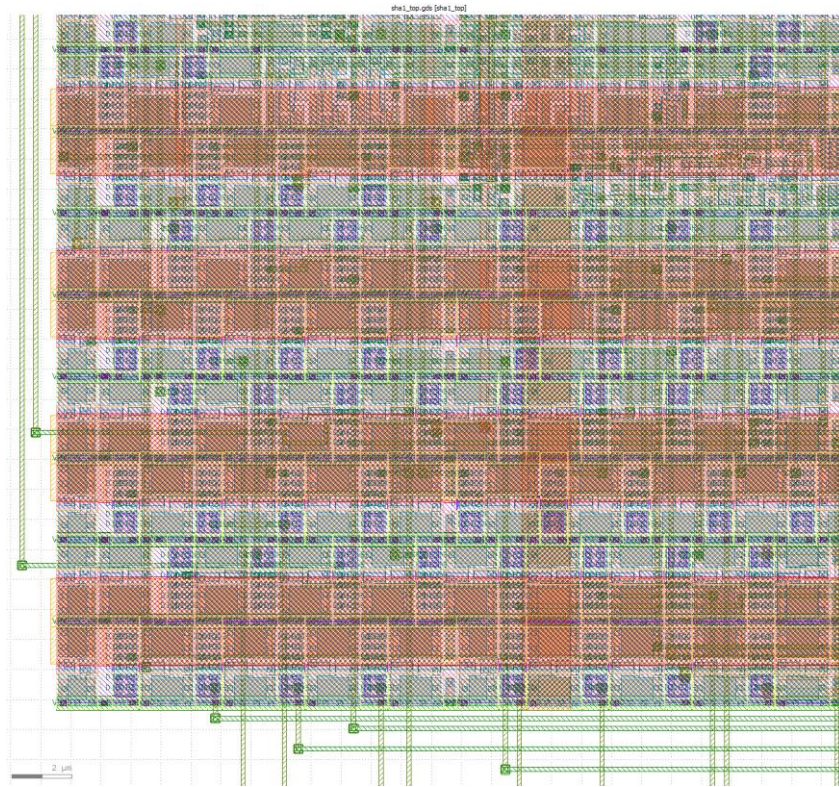


Figure 18. Hardened User Project Area (2  $\mu\text{m}$ )

The hardened user project area was placed inside the user project wrapper which connected the power lines, management SoC, and any I/O to our custom design. There were many obstacles during hardening mainly due to sparse documentation and feedback from the tools used. To resolve this, we reached out to the eFabless Slack support team to ensure our design could harden and pass precheck. The user project wrapper layout is shown below as well as a snippet of precheck results.

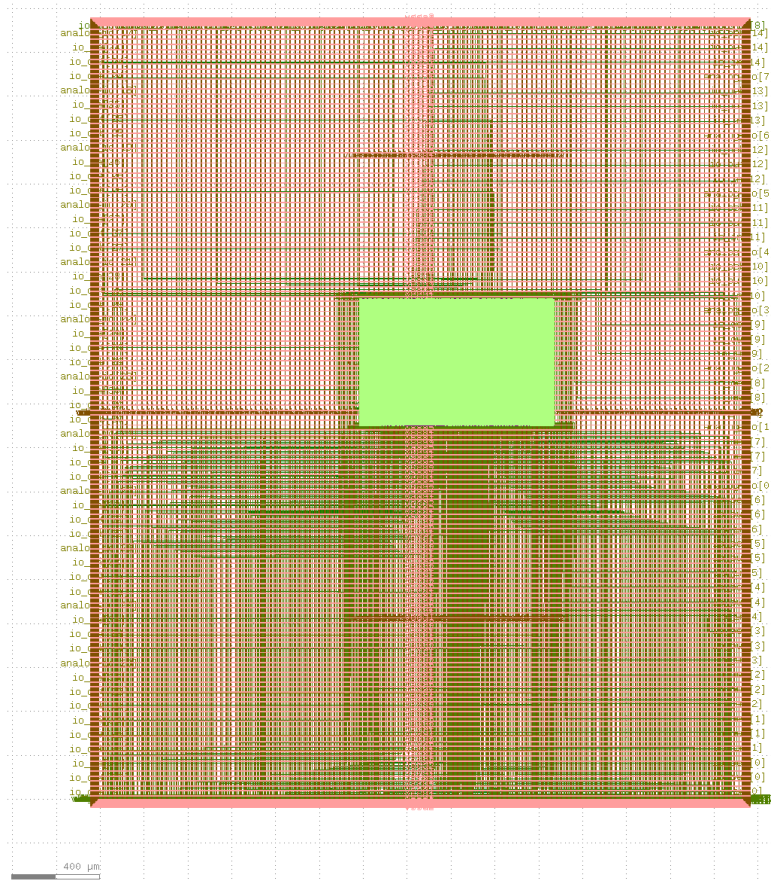


Figure 19. User Project Wrapper Layout

```

{{STEP UPDATE}} Executing Check 12 of 13: Klayout Pin Label Purposes Overlapping Drawing
No DRC Violations found
{{Klayout Pin Label Purposes Overlapping Drawing CHECK PASSED}} The GDS file, user_project_wrapper.gds, has no DRC violations.
{{STEP UPDATE}} Executing Check 13 of 13: Klayout ZeroArea
No DRC Violations found
{{Klayout ZeroArea CHECK PASSED}} The GDS file, user_project_wrapper.gds, has no DRC violations.
{{FINISH}} Executing Finished, the full log 'precheck.log' can be found in '/home/somasz/Documents/github/bitcoin_asic/precheck_results/02_DEC_2022___05_55_50/logs'
{{SUCCESS}} All Checks Passed !!!

```

Figure 20. Precheck Pass Results

The final part involved ensuring that GL simulations functioned correctly. The main difference between RTL and GL simulations was that our design had to be manually reset using the LA ports prior to using it. A GL simulation waveform for the SHA1 hardware accelerated hasher is shown below.



Figure 21. sha\_top\_test2 GL simulation showing all correct outputs (digests) using “abc” and “abcbcdcedefdefgefghghighijhijkijklklmklmnlmnomnopq” as input.

All this allowed us to reach tapeout, submit the design files to eFabless by September 12th, and wait for our ASIC to be manufactured before physically testing it on an FPGA board.

## 7 Professionalism

This discussion is with respect to the paper titled “Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment”, *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

### 7.1 AREAS OF RESPONSIBILITY

Area of responsibility	Definition	NSPE Canon	IEEE
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence;  Avoid deceptive acts.	5.  to improve the understanding of technology; its appropriate application, and

			<p>potential consequences;</p> <p>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;</p> <p>7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;</p> <p>10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.</p>
--	--	--	--

Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.	
Communication Honesty	Report work truthfully, without deception, and are understandable to stakeholders.	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.	3. to be honest and realistic in stating claims or estimates based on available data;
Health, Safety, Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	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;
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	8. to treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;



			<p>9.</p> <p>to avoid injuring others, their property, reputation, or employment by false or malicious action;</p>
Sustainability	Protect the environment and natural resources locally and globally.		<p>1.</p> <p>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;</p>
Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.	<p>2.</p> <p>to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;</p> <p>4.</p> <p>to reject bribery in all its forms;</p>

Each of the 10 IEEE codes fit under one of the categories of the NSPE version except for the financial responsibility category where IEEE seems to have no code taking this into consideration. The IEEE code of ethics also seems to focus more on ensuring workers uphold high standards of integrity, responsibility, equality, respect, credit association, and work in a safe, non-discriminatory environment.

Each entry is a rule in the IEEE code of ethics placed where they would correspond to their NSPE counterparts.

**Work Competence:** Focuses on collaborating with others for gaining high quality experience and maintaining as well as continuously improving our technical knowledge. Respectfully reviewing, critiquing, and giving credit for the work of others is also essential for improving abilities as well as maintaining integrity and respect.

**Financial Responsibility:** None

**Communication Honesty:** Focuses on being truthful and practical when addressing individuals, team members, or clients. Additionally, when using data, it is important to ensure competent, concise, and effective communication.

**Health, Safety, and Wellbeing:** Making sure to minimize any potential risks to the end users or stakeholders, whether they be physical, emotional, or their general welfare. These risks can come during the development and later use of our product and we will need to take into consideration resources that they will be able to use to minimize those risks when we are not present for them.

**Property Ownership:** Treat all people fairly and equally. Avoid injuring others property or reputations.

**Sustainability:** Accept responsibility for how the product you are designing may affect the environment.

**Social Responsibility:** Avoid conflicts of interest and reject coercion in all of its forms.

**Work Competence:** IEEE code has four rules that correspond to work competence in a more focused way on the engineering integrity of products produced by an engineer at work.

**Financial Responsibility:** The IEEE code of ethics does not have anything that would go along with the NSPE area of Financial Responsibility, it instead focuses more on other areas such as Work Competence.

**Communication Honesty:** The wording for these two versions is different but the basic message is very much the same.

**Health, Safety, and Wellbeing:** Instead of focusing on minimizing health risk there is a greater focus on accepting responsibility for potentially harmful actions.

**Property Ownership:** IEEE has a lot more about respecting the differences amongst people than they do about property.

**Sustainability:** Sustainability is only seen at the very end of one of the codes of conduct, rather than as its own entire code. This code overall is actually more applicable to Health, Safety, and Wellbeing.

**Social Responsibility:** IEEE code has two rules that correspond to social responsibility in more focused forms like rejecting bribery and disclosing perceived conflicts of interest.

## 7.2 PROJECT SPECIFIC PROFESSIONAL RESPONSIBILITY AREAS

**Work Competence:** This is an extremely relevant part responsibility for our project as we have to ensure that our work is of great quality to pass all the checks eFables has put in place for their OpenMPW shuttle. The checklist given by eFables has parts like the design passing precheck, a concise project repo with a great directory structure, and proper naming conventions. Our team is performing at a level of “High” in this area as we set up our local workflow structure and get started on our application for the ASIC.

**Financial Responsibility:** This applies to our project professionability context in that we may be given tools that our client has paid for to be better able to complete the project for them. However, we don’t have a financial responsibility in delivering the end product to our client.. Due to this, there isn’t much that applies for financial responsibility to the context of our project. Our team is performing highly for our financial responsibilities as we haven’t incurred any financial costs for our project.

**Communication Honesty:** This greatly applies to our project because we must all communicate effectively while ensuring there is minimal ambiguity regarding the tasks we have completed or will work on. Our team values honesty and ensuring that everyone is included as well as supported as the tasks are performed. Our team is performing highly in this area because we have frequent meetings, assign tasks evenly, and have proof of the work all members have completed. Furthermore, each member of the team understands the importance and value of this professional responsibility.

**Health, Safety, and Wellbeing:** This only applies to our project in specific portions. That aspect of the project is the bring up process. During the bring up, there will be a mild risk to the safety and wellbeing of the individuals doing the bring up as they will be a separate group. The team thus far has been performing at a high level for this area as we have been keeping in mind the information that will be required for the bring up. We have also made it a top priority to make the instructions and information about the processes as clear and digestible as possible.

**Property Ownership:** This area may be somewhat applicable to our team because there has not been any property or proprietary information we had to deal with, but our chip design may include intellectual property (IP) of others which we will have to take into account. Apart from licensing our own work and ensuring the design IP is accounted for, there is no need for handling confidential or sensitive information. Therefore, our team is performing medium because there will not be many applications for this responsibility area and can handle it well if needed.

**Sustainability:** This is not applicable to our team as there is no core sustainability component that our project is addressing. As eFabless is directly handling the fabrication component, and our chip is a very small part of the silicon that will be fabricated, there is a negligible impact on the environment that our project will have. Additionally, as the project is sponsored by Google, the SkyWater fabrication facility is focused heavily on making sure their practices are rooted in sustainable practices. To summarize, our project has no core sustainability component, and the sustainable components related to the fabrication are handled directly by eFabless; therefore, this is not applicable (N/A) to our team.

**Social Responsibility:** This area is somewhat applicable to our project as depending on how well we are able to document the process of building and getting a chip fabricated through Efabless it will be a great service to future groups doing this exact thing. In terms of our team's performance in this area we are performing at a high level. We have been able to compile an immense amount of resources for being able to create a user project within the context of the caravel harness and user project space. We have been condensing this information as we have examined it to give later groups an easier and more efficient way of digesting this information. We will be continuing this process throughout the project so as to maintain a high level of performance for this area.

### 7.3 MOST APPLICABLE PROFESSIONAL RESPONSIBILITY AREA

Communication Honesty is one area of professional responsibility that is extremely important for our project above all other areas. As a team of four, working closely with the numerous interconnected parts of this project, it is important that we communicate with one another about both research done and work implemented as it will likely directly impact another team member's area of specialization within the project. Our team has demonstrated this extensively over the past few weeks by having weekly client meetings where we discuss our direction as well as team meetings where we focus more in depth about our project's components. One of the biggest impacts recently that we found from this responsibility was the interconnection of understanding of the wishbone, caravel harness, workflow, and logic analyzer parts of the project. By continuing our commitment to this responsibility, we will be able to more efficiently and effectively implement each of our parts related to the project and meet the deadline given to us by eFabless.

## 8 Closing Material

### 8.1 DISCUSSION

The result our team hopes for is to have a submitted ASIC design be selected by eFabless for their Open MPW shuttle, manufactured, and then once we receive test its functionality to ensure that it works as intended.

In our specific design case we hope that our hardware accelerated SHA-1 ASIC is able to be placed onto an FPGA board and we are able to not only verify that the hash was done correctly, but also that it has a faster hashing time than non hardware implementations of SHA-1.

## 8.2 CHANGES SINCE CPRE 491

Our biggest change since Cpre 491 is that we had to completely redesign our ASIC submission because during the hardening process we found that our original design was not within the space constraints that Efabless provides for submission.

We shifted from a complicated bitcoin mining ASIC which had many parts that simulate the full bitcoin mining process to a much simpler accelerated adder that would meet the space constraints set by Efabless for shuttle submission.

We completed and tested this design and turned in our submission to Efabless. We have passed precheck and tapeout on the shuttle and are now waiting for review from the Efabless shuttle team to determine if we have been selected for fabrication or not.

## 8.3 CONCLUSION

The work we have done so far is researching the Open MPW submission process, previous submissions, provided workspace, and constraints. We have selected a design, researched about bitcoin mining, figured out what components would be needed to make the design functional and what their constraints are. We then had to drastically alter our design to meet the changed requirements for the MPW shuttle. We have devised a testing plan for the digital design before submitting the ASIC as well as a plan for once the ASIC is manufactured and sent back to us. We have designed a simple adder in the new workspace to demonstrate that we do have the ability to design and test components successfully within this new workspace. We have selected the IP we will use for our SHA-1 hashing, integrated it, implemented our new design, and submitted it.

The rest of the time spent on this course has been helping the group that is following us with their own ASIC design, and going through the additional testing of our ASIC that Efabless has specified. We will also be waiting to hear the final decision from Efabless, and making sure that the next team understands the bring up processes that we have developed. Over the course of this senior design project we have had to overcome many challenges pertaining to the Efabless workspace as well as changing requirements for the MPW 7 shuttle. We have been able to overcome these challenges as a team and managed to pioneer a method for student led groups to design and fabricate their own ASIC designs.

## 8.4 APPENDICES

### 8.4.1 Team Contract

**Team Name:** Digital ASIC Designers

**Team Members:**

- |                       |                    |
|-----------------------|--------------------|
| 1) Constantine Mantas | 2) Soma Szabo      |
| 3) Dawood Ghauri      | 4) Courtney Violet |

**Team Procedures**

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
  - a. Client Meeting: Friday in Durham with Duwe (1 pm; 1 hour).
  - b. Team Meeting: Weekend meeting (variable; 1 hour).
  
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
  - a. Discord
  - b. Snapchat
  - c. MS Teams
  - d. MS Planner
  
3. Decision-making policy (e.g., consensus, majority vote):
  - a. Majority vote with professor Duwe making a decision if there is a tie.
  
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
  - a. Constantine will be the main record keeper with everyone else adding to it as they feel necessary.
  - b. Times will be maintained by each individual in a shared excel spreadsheet which will also detail what each person has accomplished in the week.

## **Participation Expectations**

1. Expected individual attendance, punctuality, and participation at all team meetings:
  - a. Face to face meetings:
    - i. Be as punctual as possible, especially when meeting with professors or other people outside of our team.
    - ii. Try to arrive a few minutes early or as discussed.
  - b. Online meetings:
    - i. Similar to face-to-face but could have a few minutes (max 5) of slack.
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
  - a. Ensure they are completed or worked on before the deadline.
    - i. If someone is having issues, they should let the team know as soon as possible.
  - b. Preferably complete work to a high standard and follow guidelines, specifically when completing issues on GitHub and merging branches.
3. Expected level of communication with other team members:
  - a. Ensure effective communication about all project work and meetings.
  - b. As stated previously, if any issue arises, let team members know as soon as possible.
4. Expected level of commitment to team decisions and tasks:
  - a. Varies with task urgency. High urgency tasks will be expected to be completed within a week while others may require more time to explore solutions. Communicating progress weekly is essential to demonstrate commitment.

## **Leadership**

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

Role	Team Member
Team Organization	Constantine Mantas
Design Workflow	Dawood Ghauri
Individual Component Design	Soma Szabo
Testing	Courtney Violet

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

- a. Make use of human resources including asking team members and professor Duwe for assistance when stuck.
- b. Communicate roadblocks as quickly as possible so that the team can work together to find a solution.
- c. Use issues on GitHub and planned tasks so we know what each member is working on and no duplicate work is done.

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

- a. Assign weekly tasks that set expectations for each member with the usual expectation being that they be finished unless a roadblock occurs.
- b. Can verbally praise people for their work.

**Collaboration and Inclusion**

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

Skills/Experience/Perspectives	Team Member
VHDL coding experience, Github workflow, effective communicator, leadership experience in a hardware design environment (TA for 381), Embedded systems industry experience	Constantine Mantas



VHDL, Verilog, UNIX/Windows Server Management, Undergraduate Research Experience, Leadership Skills from TA (381).	Dawood Ghauri
VHDL, Verilog, scripting software, GitHub workflow, effective communicator, storage industry experience, PCB design	Soma Szabo
VHDL, verilog, GitHub workflow, Hardware design in 381, effective communicator. Experience in making WebDev applications, experience working with embedded side projects(Arduino)	Courtney Violett

2. Strategies for encouraging and support contributions and ideas from all team members:
  - a. Inquisition: Ask each other questions about what implementations and ideas they may have so all people get a good understanding of how things work.
  - b. Inclusion: Encourage members to speak up and take on tasks or discuss what they think are good action paths.
3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
  - a. Contact team members through any of the means of communication for most issues if it requires professor Duwe's attention make sure to @ him and team in Microsoft Teams.

### **Goal-Setting, Planning, and Execution**

1. Team goals for this semester:
  - a. Design and begin coding of a submittable and likely to be selected Open MPW chip design.
  - b. Learn more about chip design, firmware, and the process of fabricating an integrated circuit.
  - c. Work effectively as a team and resolve conflicts peacefully.
2. Strategies for planning and assigning individual and team work:

- a. Weekly progress evaluations and task assignments at meetings, and try to assign tasks relevant to people’s strengths with potentially multiple people working together on more difficult tasks.

3. Strategies for keeping on task:

- a. Task board with assignments and deadlines in Microsoft Teams group.
- b. Remind team members of any tasks if they seem behind schedule or drifting off topic.

**Consequences for Not Adhering to Team Contract**

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

- a. Reach out through a means of communication to ensure expectations are understood and infractions are minimized in the future.

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

- a. If they do not have a strong reason for the infraction, the team will reach out to professors like Duwe or Shannon to discuss consequences.

\*\*\*\*\*

*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) Constantine Mantas | DATE 2/12/2022 |
| 2) Soma Szabo         | DATE 2/12/2022 |
| 3) Dawood             | DATE 2/12/2022 |
| 4) Courtney Violet    | DATE 2/12/2022 |

## 8.4.2 Alternative Design

Our original ASIC design was for a SHA-256 Bitcoin mining accelerator. However, due to changes in the eFabless manufacturing process with respect to space constraints, we had to change our project. Our updated ASIC design is a SHA-1 hardware accelerator. More details can be found in sections 6.2, 4.5, and 4.6.

## 8.4.3 Github Link (Full Code Repository)

[https://github.com/WebKingdom/bitcoin\\_asic](https://github.com/WebKingdom/bitcoin_asic)

## 8.4.4 User Manual

Our user manual is focused around our primary objective of setting up future groups to be able to design their own asics using the Efabless Open MPW shuttle framework. This manual outlines the steps groups will need to take in order to put together a successful submission to the shuttle.

Use this guide alongside the most up to date README for user\_caravel\_project which is linked within the first bullet point

## Getting Setup

### Cloning & Caravel:

- Begin with cloning your caravel-project to your machine and then going to follow the caravel install instructions in README for user\_caravel\_project at.
- [caravel\\_user\\_project/index.rst at main · efabless/caravel\\_user\\_project \(github.com\)](#)

### WSL:

- Set up a linux vm on Windows:
  - <https://docs.microsoft.com/en-us/windows/wsl/install>
  - --<distro name> = ubuntu
  - Ensure Virtual Machine Platform and Windows Subsystems for Linux are enabled in Windows features

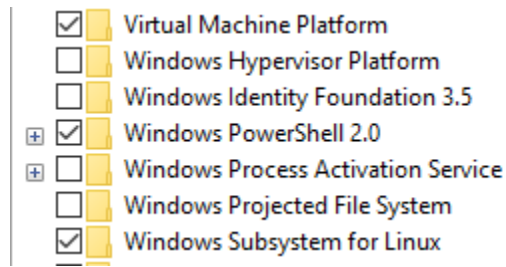


Figure 22. Required Windows Features

- To ensure WSL2 does not use all the RAM on your system, create or edit the .wslconfig file in your %USERPROFILE% directory (Can be found by typing %USERPROFILE% in file explorer but is usually in C:/Users/<username>/).
- Contents should be (can add more like processors = ...):

[wsl2]

memory = 8GB # Limits VM memory

- In WSL run these commands to setup basic linux toolkit or to keep tools updated:
  - sudo apt update
  - sudo apt upgrade

### MAP WSL:

- Open file explorer
- Right click This PC
- Click “Map network drive...”
- In the Folder: field paste “\\wsl\$\\Ubuntu”
- This will allow you to see your linux workspace from the windows file system for easy editing on VS Code

### Docker Setup:

- Setup Docker and integrate with WSL using this guide:
  - <https://docs.microsoft.com/en-us/windows/wsl/tutorials/wsl-containers>

Magic setup (optional, only if local environment does not function):

- Clone most recent magic repository into wsl
  - `git clone git://opencircuitdesign.com/magic`
- Install other dependencies by running these commands:
  - `sudo apt-get install m4`
  - `sudo apt-get install tcsh`
  - `sudo apt-get install csh`
  - `sudo apt-get install libx11-dev`
  - `sudo apt-get install tcl-dev tk-dev`
  - `sudo apt-get install mesa-common-dev libglu1-mesa-dev`
- Enter git repo for magic and run:
  - `./configure`
  - `sudo make`
  - `sudo make install`

### **Bash Script setup:**

- Go to /etc and add these exports to the bottom of the file `bash.bashrc` so that you don't have to on every boot up
  - `export DISPLAY=$(grep -m 1 nameserver /etc/resolv.conf | awk '{print $2}'):0.0`
  - `export DISPLAY=$(ip route|awk '/^default/{print $3}'):0.0`
  - `export OPENLANE_ROOT=~/<Tmp directory>/openlane`
  - `export PDK_ROOT=~/<Tmp directory>/pdk`

Once you have installed Ubuntu and have made your account and password run the command 'sudo apt install build-essential' to install all the essential commands that you will need to run ie(make and gcc)

When installing [caravel user project](#), make sure to use the latest mpw-5b release and root (that is, execute `sudo su -` prior to executing the instructions in that roundtrip document at step 4 and beyond). In addition, use the full caravel package (still pending verification) with the export `CARAVEL_LITE=0` command to prevent any make issues with openlane.

NOTE: mpw-5c has no issues following the quicktrip documentation (use that for future installs).

## Adding Custom Designs to the User Project Wrapper:

At this point the group needs to implement their own design(s) within the `user_project_wrapper` file in order to have a valid submission.

## Setting up an Existing Design in the User Project Wrapper:

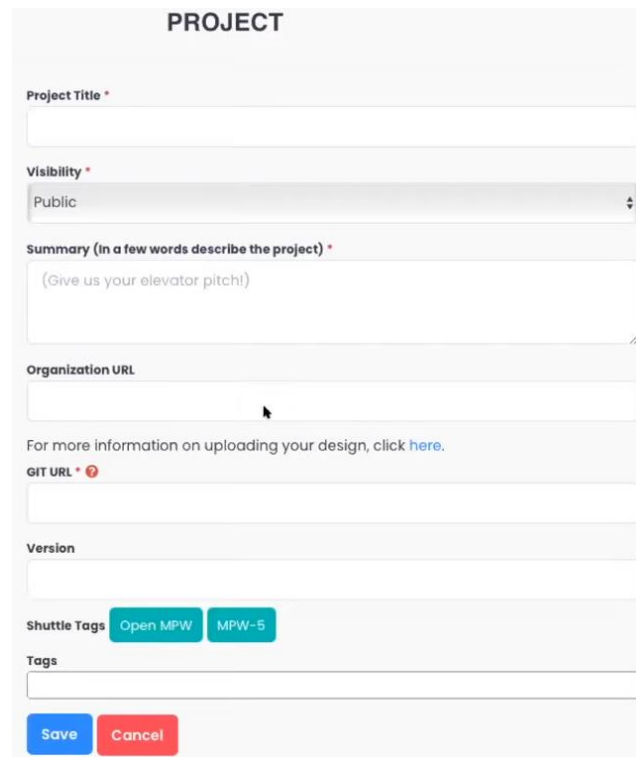
To harden and submit the SHA1 hardware accelerator design created by our team:

1. Ensure you have WSL set up on Windows mentioned in “Getting Setup” [8.4.4 User Manual](#) and can go through the example `user_project` hardening, simulation, and MPW precheck process. Refer to READMEs in: [efables/caravel user project: https://caravel-user-project.readthedocs.io \(github.com\)](https://caravel-user-project.readthedocs.io (github.com))
2. On Linux/Mac, you do not need WSL.
3. Clone the GitHub repository at: [WebKingdom/bitcoin\\_asic: A Bitcoin mining ASIC \(github.com\)](#)
4. If you have configured your local environment according to the READMEs in: [efables/caravel user project: https://caravel-user-project.readthedocs.io \(github.com\)](https://caravel-user-project.readthedocs.io (github.com)) and have Docker running, the following commands should harden the design:
  - a. `make sha1_top`
  - b. `make user_project_wrapper`
5. To get RTL and GL simulation results run:
  - a. `make verify-sha1_top_test1-rtl`
  - b. `make verify-sha1_top_test2-rtl`
  - c. `make verify-sha1_top_test1-gl`
  - d. `make verify-sha1_top_test2-gl`
6. To view simulation run:
  - a. `gtkwave verilog/dv/sha1_top_test2/RTL-sha1_top_test1.vcd`
  - b. `gtkwave verilog/dv/sha1_top_test2/GL-sha1_top_test1.vcd`
  - c. `gtkwave verilog/dv/sha1_top_test2/RTL-sha1_top_test2.vcd`
  - d. `gtkwave verilog/dv/sha1_top_test2/GL-sha1_top_test2.vcd`
7. To run precheck:
  - a. `make precheck`
  - b. `make run-precheck`

## Testing Locally:

Refer to the simulation and hardening sections of the Readme for the user\_project\_wrapper README. ([Link](#))

## Submitting to the eFabless Open MPW Shuttle:



The image shows a web form titled "PROJECT" for submitting a project to eFabless. The form contains several input fields and a dropdown menu. The fields are: "Project Title" (required), "Visibility" (dropdown menu set to "Public"), "Summary (In a few words describe the project)" (required, with a hint "(Give us your elevator pitch!)"), "Organization URL", "GIT URL" (required, with a help icon), "Version", "Shuttle Tags" (with two buttons: "Open MPW" and "MPW-5"), and "Tags". At the bottom of the form are "Save" and "Cancel" buttons.

Figure 23. Submission Page to eFabless. Source: [6]

- Create a project on the Efabless website, and fill out all required information.
- Execute a precheck verification on your project's workspace.
- Execute a tapeout verification on your project's workspace.
- Download the export compliance form and complete and submit via the request.
- Review and complete your MPW service agreement.
- Review deliverables of your MPW request and select 'Submitter Confirmed' when complete.

## 8.4.5 References

- [1] H. L. Pham, T. H. Tran, T. D. Phan, V. T. Duong Le, D. K. Lam and Y. Nakashima, "Double SHA-256 Hardware Architecture With Compact Message Expander for Bitcoin Mining," in IEEE Access, vol. 8, pp. 139634-139646, 2020, doi: 10.1109/ACCESS.2020.3012581.
- [2] M. Vilim, H. Duwe and R. Kumar, "Approximate bitcoin mining," 2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), 2016, pp. 1-6, doi: 10.1145/2897937.2897988.
- [3] J. Kaur and L. Sood, "Comparison Between Various Types of Adder Topologies," 2022 IJCST. Available: <http://www.ijcst.com/vol61/1/13-Jasbir-Kaur.pdf>. [Accessed: 26-Mar-2022].
- [4] H. L. Pham, T. H. Tran, T. D. Phan, V. T. Duong Le, D. K. Lam and Y. Nakashima, "Double SHA-256 Hardware Architecture With Compact Message Expander for Bitcoin Mining," in IEEE Access, vol. 8, pp. 139634-139646, 2020, doi: 10.1109/ACCESS.2020.3012581.
- [5] "Getting Started with Open MPW and chipIgnite" YouTube, uploaded by Efabless, 3rd of February 2022, <https://www.youtube.com/watch?v=vJqP7ZRoNrI>.
- [6] "Efabless.com," eFabless. [Online]. Available: [https://efabless.com/open\\_shuttle\\_program](https://efabless.com/open_shuttle_program).
- [7] "Caravel," Zero to ASIC Course. [Online]. Available: <https://www.zerotoasiccourse.com/tags/caravel/>.
- [8] "Caravel user project," Caravel User Project - CIIC Harness documentation. [Online]. Available: <https://caravel-user-project.readthedocs.io/en/latest/>. [Accessed: 05-Dec-2022].
- [9] "Hash algorithm comparison: MD5, SHA-1, SHA-2 & sha-3," Code Signing Store, 23-Mar-2022. [Online]. Available: <https://codesigningstore.com/hash-algorithm-comparison>. [Accessed: 05-Dec-2022].

[Double SHA-256 Hardware Architecture With Compact Message Expander for Bitcoin Mining | IEEE Journals & Magazine | IEEE Xplore](#)