CryptoHash

David Cheung, William Zhao, Lulu Shyr

Use Case

Specialized hardware only works for a specific cryptocurrency.

Cryptocurrencies values fluctuate wildly!

Specialized hardware is expensive!

Support at least two proof-of-work coins, Bitcoin and Ethereum.

Others include Litecoin, Dogecoin.

Our default choosing algorithm will choose the optimal spread of coins.

Quantitative Use-Case Requirements

Hash rate has to be competitive, we want to have at least 90% of the hashing power that market GPUs boast, but with a lower hardware cost.

Choosing mechanism needs to pull from current data, trained from price data and trends from the past 3 months.

Communication overhead minimal, configure the new settings within 10 seconds.

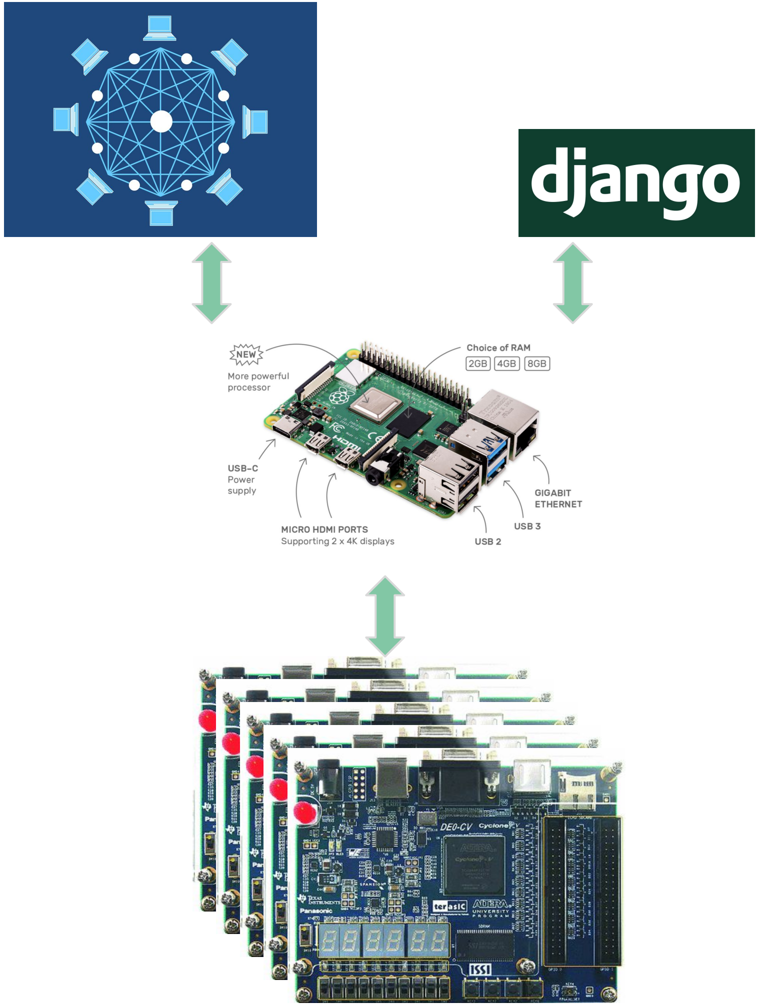
Display metrics to user such as hashrate, power consumption, current prices.

Solution Approach:

For our FPGAs, we wanted low-cost boards that still have good performance.

We chose to use a Raspberry Pi 4 Model B for its connections and performance.

Django is a free and open source web framework that works well with our Raspberry Pi.



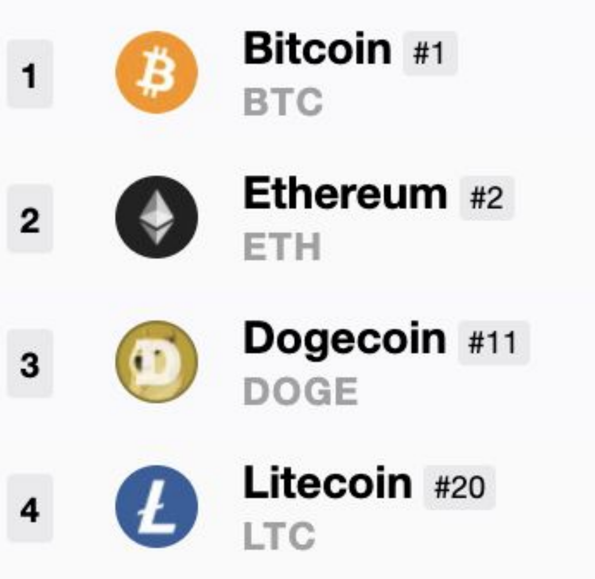
Solution Approach:

BINANCE

Free Binance API for retrieving cryptocurrency statistics

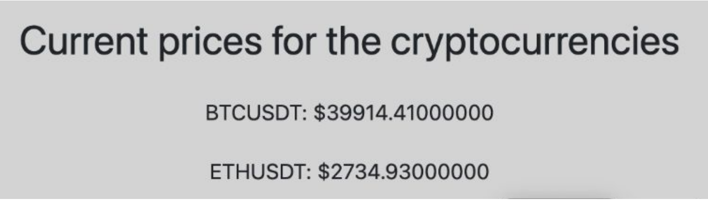
Bitcoin and Ethereum are the most popular proof of work coins

Scalability: Use network buses for the GPIO pins that connect the FPGAs and the Raspberry Pi



1	Bitcoin #1	BTC
2	Ethereum #2	ETH
3	Dogecoin #11	DOGE
4	Litecoin #20	LTC

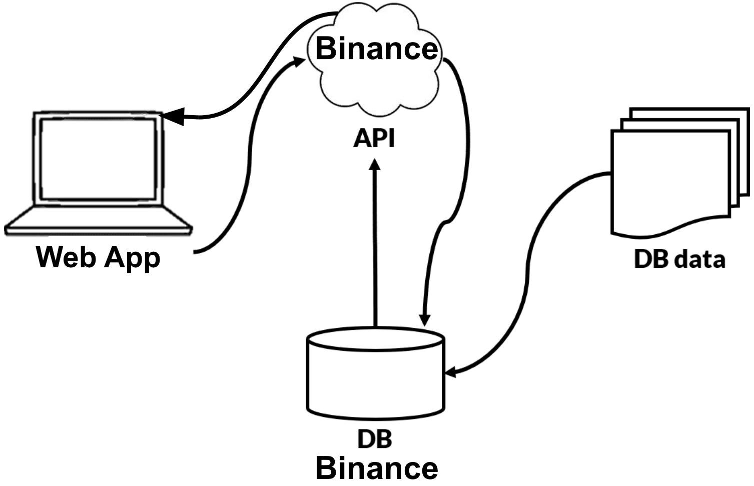
System Specification

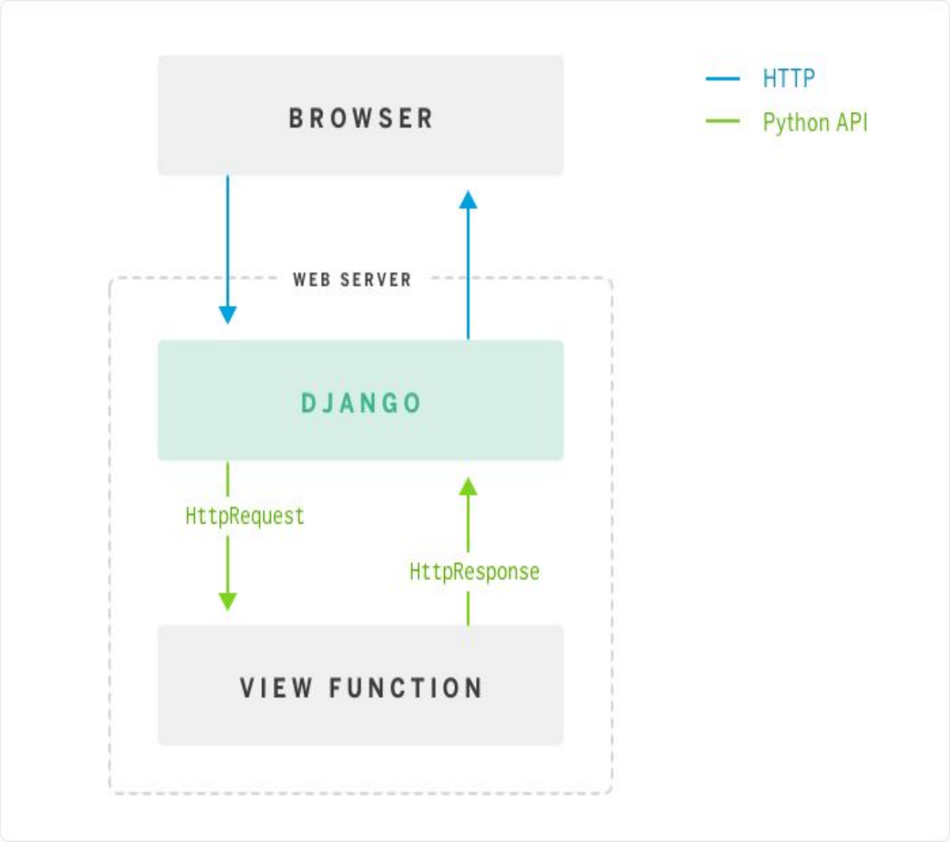


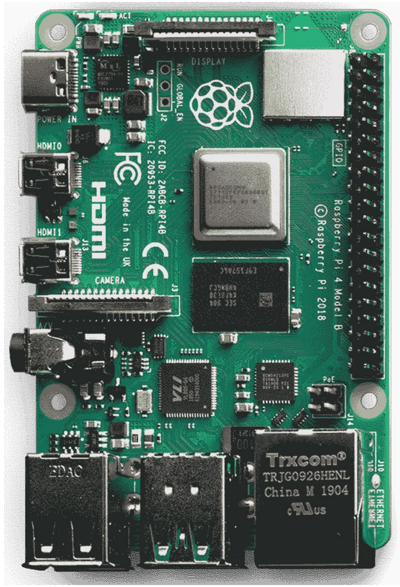
Current prices for the cryptocurrencies

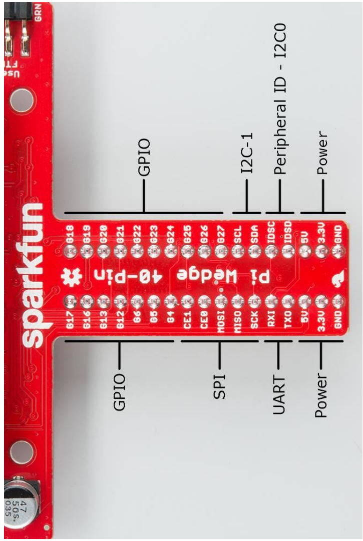
BTCUSDT: $39914.41000000

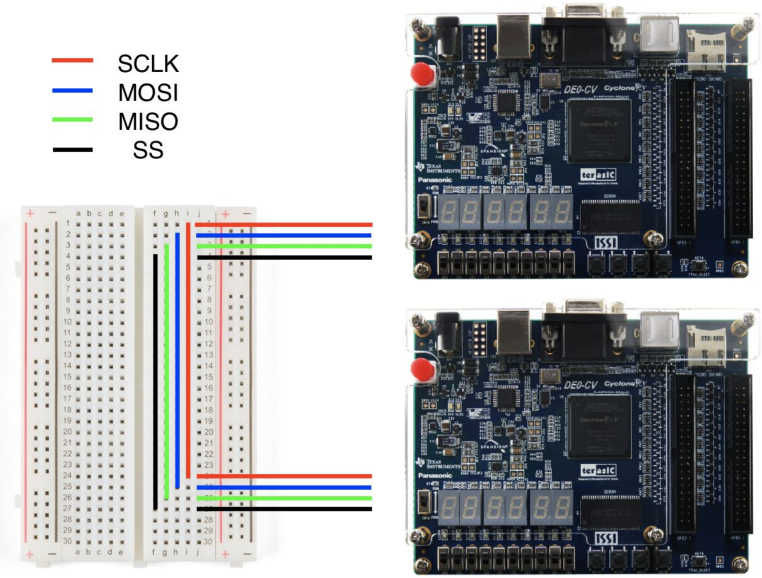
ETHUSDT: $2734.93000000



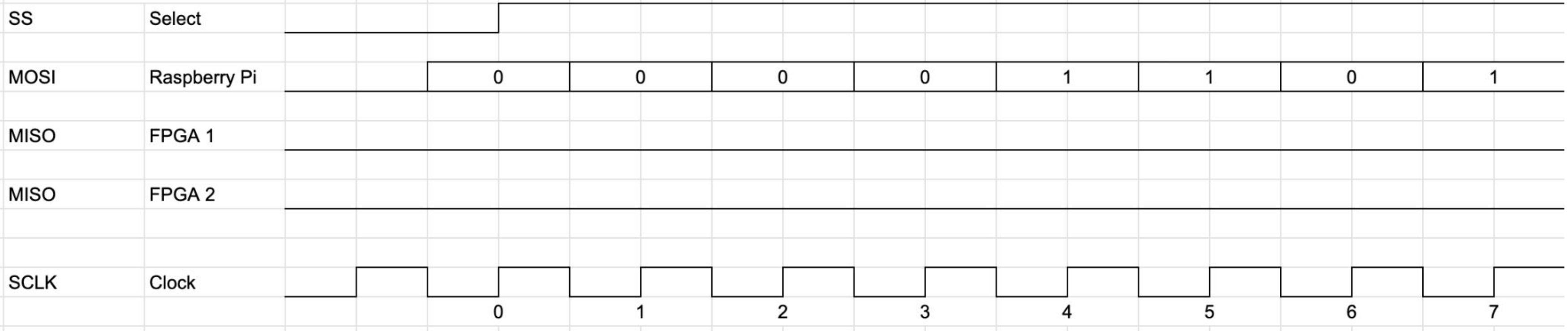




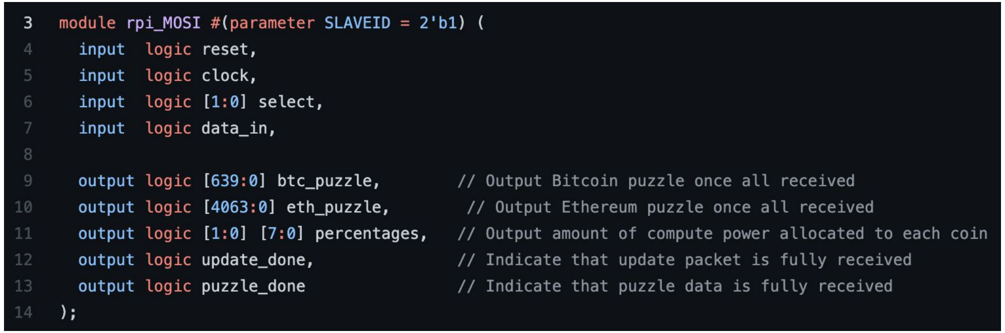

GPIO
I2C-1
Peripheral ID - I2C0
Power
G18 G19 G20 G21 G22 G23 G24 G25 G26 G27 SCL SDA IDSC IDSD 5V 3.3V GND
sparkfun
Pi Wedge 40-Pin
G17 G16 G13 G12 G6 G5 G4 CE1 CE0 MOSI MISO SCK RXI TXO 5V 3.3V GND
GPIO
SPI
UART
Power



Two FPGA Communciation



Implementation Plan



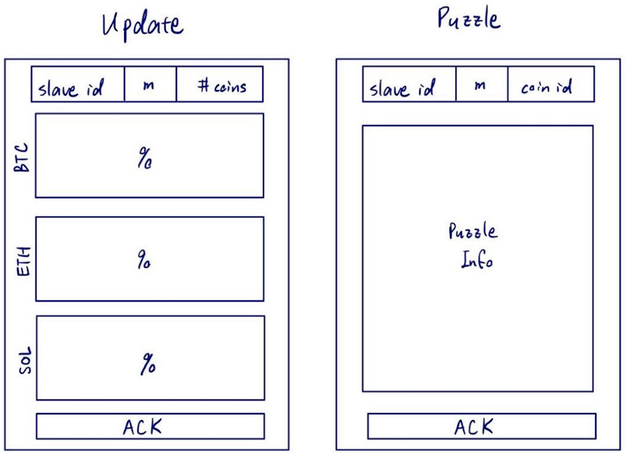
```
module rpi_MOSI #(parameter SLAVEID = 2'b1) (
  input  logic reset,
  input  logic clock,
  input  logic [1:0] select,
  input  logic data_in,

  output logic [639:0] btc_puzzle,         // Output Bitcoin puzzle once all received
  output logic [4063:0] eth_puzzle,         // Output Ethereum puzzle once all received
  output logic [1:0] [7:0] percentages,    // Output amount of compute power allocated to each coin
  output logic update_done,                // Indicate that update packet is fully received
  output logic puzzle_done                 // Indicate that puzzle data is fully received
);
```

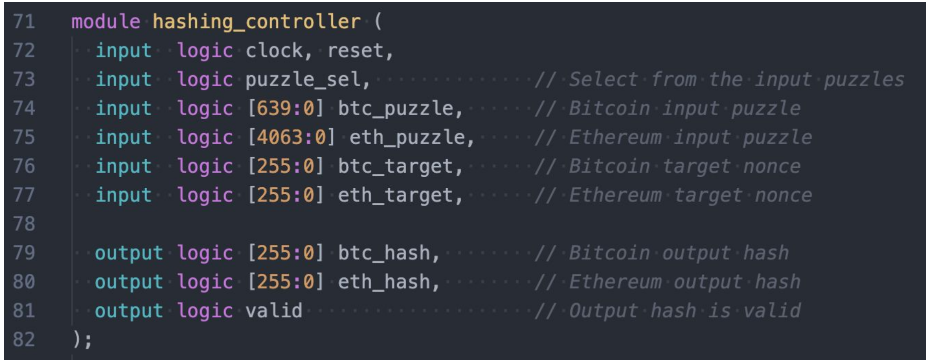
Raspberry Pi

- Defined communication protocol called SPI
- Using Python RPI.GPIO library
- Along with the library, developing own software to handle sending/receiving bytes

Other setups exist using a Pi Wedge but we are creating our own.



Implementation Plan



```
module hashing_controller (
  input  logic clock, reset,
  input  logic puzzle_sel,              // Select from the input puzzles
  input  logic [639:0] btc_puzzle,      // Bitcoin input puzzle
  input  logic [4063:0] eth_puzzle,     // Ethereum input puzzle
  input  logic [255:0] btc_target,      // Bitcoin target nonce
  input  logic [255:0] eth_target,      // Ethereum target nonce

  output logic [255:0] btc_hash,        // Bitcoin output hash
  output logic [255:0] eth_hash,        // Ethereum output hash
  output logic valid                    // Output hash is valid
);
```

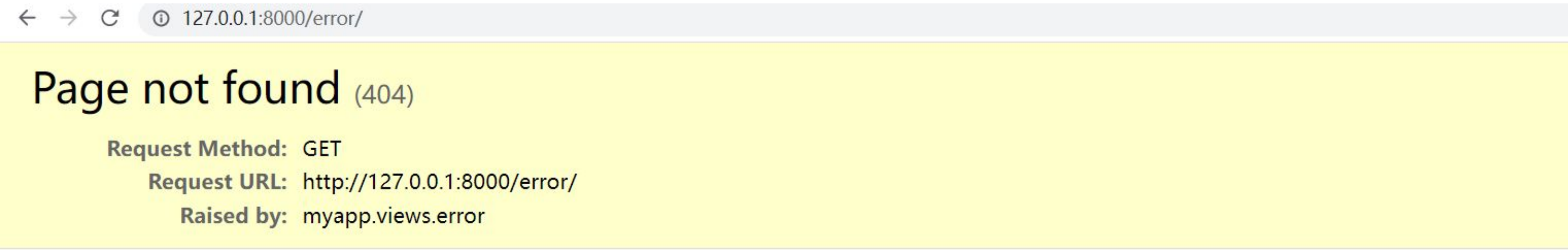
Mining Elements

- Using Binance API to pull pricing data
- Quantitative analysis of pricing data will be self-developed
- Connect to a mining pool with a stratum+tcp url
- Custom FPGA mining controller to provide inputs to the hashing module
- Depending on the coin, we will find how to implement the hash functions and integrate them

Test, Verification and Validation

Web app

- Test the user interface
- Test the functionality of web app
- Test the data transmission between Raspberry Pi and Web app
- FPGA output a done signal to check the 10s requirement for switching configuration



127.0.0.1:8000/error/

Page not found (404)

Request Method: GET
Request URL: http://127.0.0.1:8000/error/
Raised by: myapp.views.error

Test, Verification and Validation

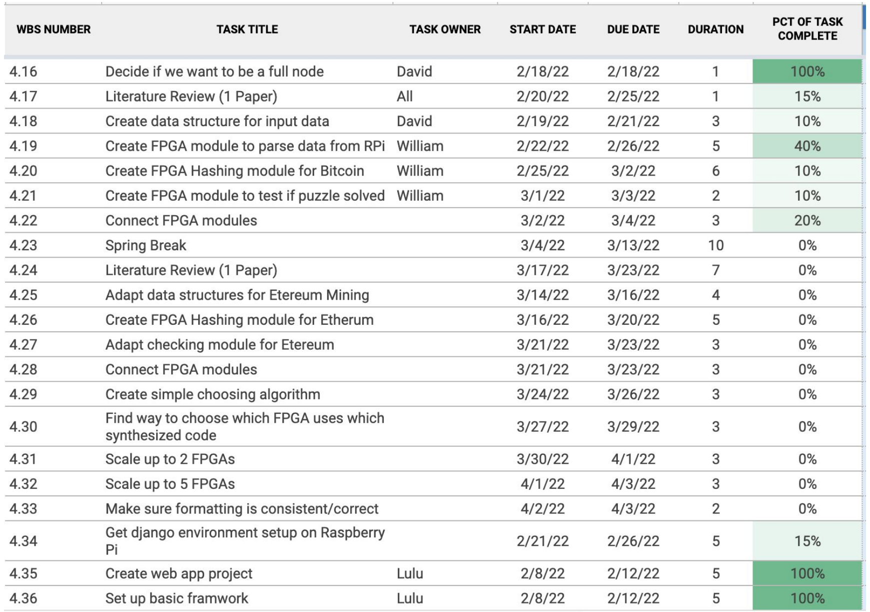
Raspberry Pi

- Test Raspberry Pi communications locally before deployment
- Simulate back and forth communication between host and device

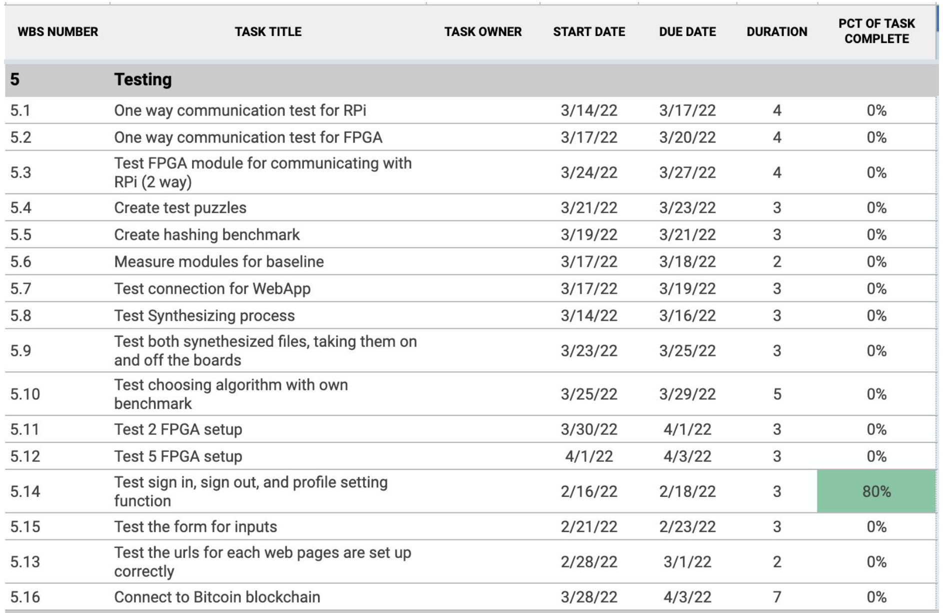
FPGA

- Use combination of SystemVerilog testbenches and VCS to simulate design
- Synthesize onto FPGA to test communication and hashing
- Simulate input through the board’s switches and buttons

Project Management



WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
4.16	Decide if we want to be a full node	David	2/18/22	2/18/22	1	100%
4.17	Literature Review (1 Paper)	All	2/20/22	2/25/22	1	15%
4.18	Create data structure for input data	David	2/19/22	2/21/22	3	10%
4.19	Create FPGA module to parse data from RPi	William	2/22/22	2/26/22	5	40%
4.20	Create FPGA Hashing module for Bitcoin	William	2/25/22	3/2/22	6	10%
4.21	Create FPGA module to test if puzzle solved	William	3/1/22	3/3/22	2	10%
4.22	Connect FPGA modules		3/2/22	3/4/22	3	20%
4.23	Spring Break		3/4/22	3/13/22	10	0%
4.24	Literature Review (1 Paper)		3/17/22	3/23/22	7	0%
4.25	Adapt data structures for Etereum Mining		3/14/22	3/16/22	4	0%
4.26	Create FPGA Hashing module for Etherum		3/16/22	3/20/22	5	0%
4.27	Adapt checking module for Etereum		3/21/22	3/23/22	3	0%
4.28	Connect FPGA modules		3/21/22	3/23/22	3	0%
4.29	Create simple choosing algorithm		3/24/22	3/26/22	3	0%
4.30	Find way to choose which FPGA uses which synthesized code		3/27/22	3/29/22	3	0%
4.31	Scale up to 2 FPGAs		3/30/22	4/1/22	3	0%
4.32	Scale up to 5 FPGAs		4/1/22	4/3/22	3	0%
4.33	Make sure formatting is consistent/correct		4/2/22	4/3/22	2	0%
4.34	Get django environment setup on Raspberry Pi		2/21/22	2/26/22	5	15%
4.35	Create web app project	Lulu	2/8/22	2/12/22	5	100%
4.36	Set up basic framwork	Lulu	2/8/22	2/12/22	5	100%



WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
5	**Testing**					
5.1	One way communication test for RPi		3/14/22	3/17/22	4	0%
5.2	One way communication test for FPGA		3/17/22	3/20/22	4	0%
5.3	Test FPGA module for communicating with RPi (2 way)		3/24/22	3/27/22	4	0%
5.4	Create test puzzles		3/21/22	3/23/22	3	0%
5.5	Create hashing benchmark		3/19/22	3/21/22	3	0%
5.6	Measure modules for baseline		3/17/22	3/18/22	2	0%
5.7	Test connection for WebApp		3/17/22	3/19/22	3	0%
5.8	Test Synthesizing process		3/14/22	3/16/22	3	0%
5.9	Test both synethesized files, taking them on and off the boards		3/23/22	3/25/22	3	0%
5.10	Test choosing algorithm with own benchmark		3/25/22	3/29/22	5	0%
5.11	Test 2 FPGA setup		3/30/22	4/1/22	3	0%
5.12	Test 5 FPGA setup		4/1/22	4/3/22	3	0%
5.14	Test sign in, sign out, and profile setting function		2/16/22	2/18/22	3	80%
5.15	Test the form for inputs		2/21/22	2/23/22	3	0%
5.13	Test the urls for each web pages are set up correctly		2/28/22	3/1/22	2	0%
5.16	Connect to Bitcoin blockchain		3/28/22	4/3/22	7	0%

Project Management

Blockchain (David):

Create Raspberry Pi software to communicate with Bitcoin blockchain (input)

Create Raspberry Pi software to communicate with Bitcoin blockchain (output)

Mining (William):

Create FPGA module to parse data from RPi

Create FPGA Hashing module for Bitcoin

Create FPGA module to test if puzzle solved

Web App (Lulu):

Get django web app setup on Raspberry Pi

Communicate with the Binance API to get up to date cryptocurrency data