CryptoHash

David Cheung, William Zhao, Lulu Shyr

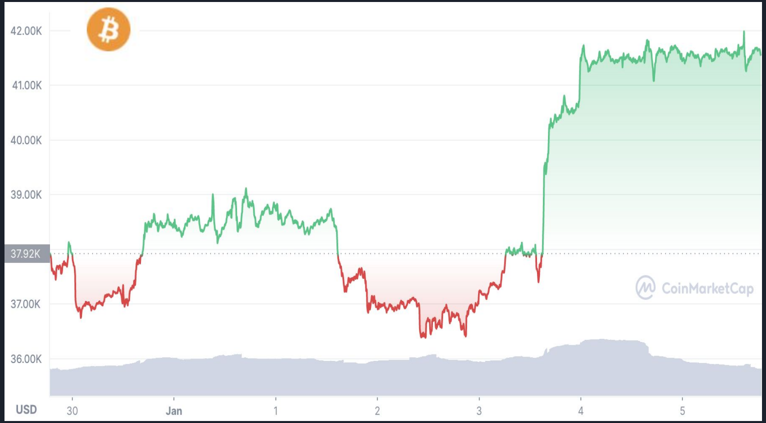
Use Case

Specialized hardware only works for a specific cryptocurrency.

Cryptocurrencies values fluctuate wildly!

Specialized hardware is expensive!

Now, what if they had some spare cheaper FPGAs lying around…

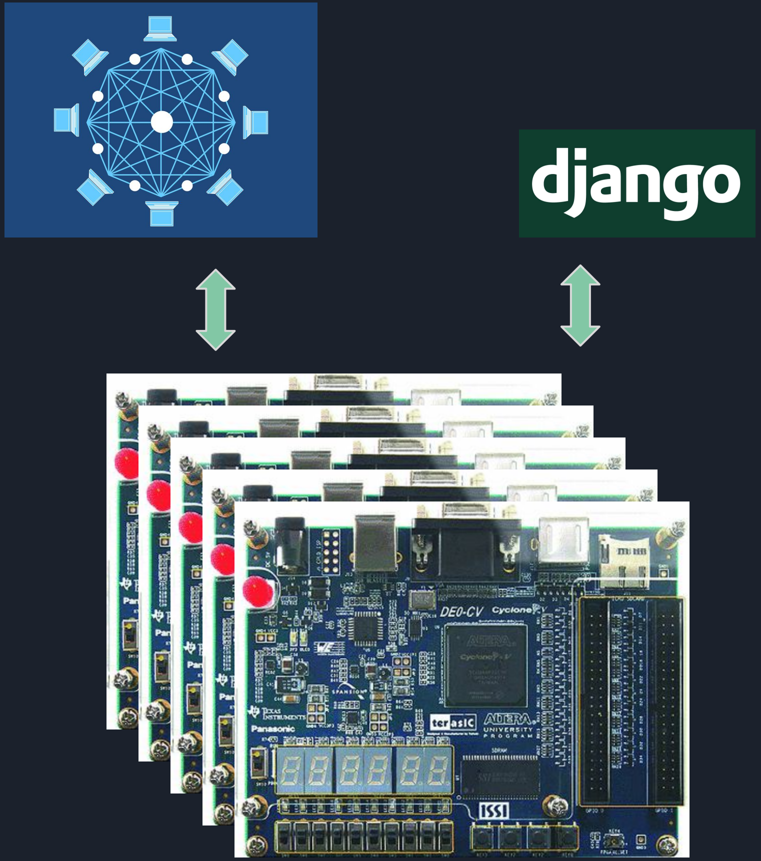


Use Case

Our Solution?

A cryptocurrency miner that can choose which currency to mine, using some DE0-CV (Cyclone V) boards.

The user will also have a webapp where they can control how much they want to mine of a certain cryptocurrency or allow our miner to choose for them.



Our project will span the Hardware Systems and Software Systems areas

Use Case Requirements

Requirements for the FPGA miner:

Support Bitcoin and Ethereum Mining

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 price

Choosing mechanism needs to pull from current data, trained from previous history

Expected returns has to be greater than energy cost, our goal is 5% returns

Use Case Requirements

Requirements for the WebApp:

Communication overhead minimal, we want to configure the new settings within 10 seconds

Defaults to program chosen spread for optimal spread, unless user changes settings, where it will default to user settings

Needs to display metrics to user such as hashrate, power consumption, current prices

Technical Challenges

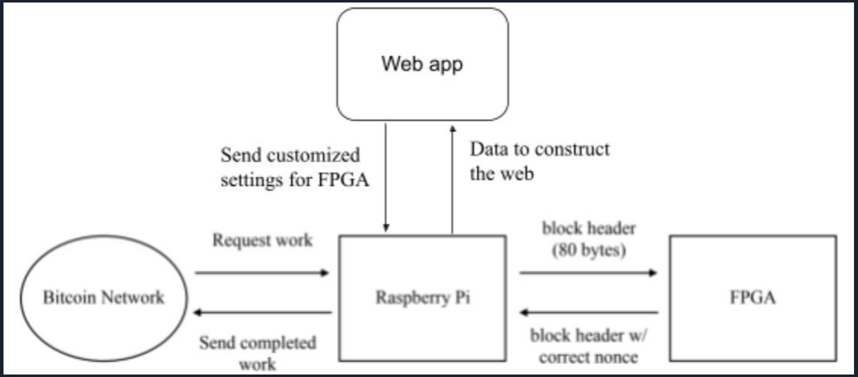
WebApp Challenges

- System to WebApp communication must not degrade mining performance
- User customization limited to a certain degree of specificity

FPGA Challenges

- FPGAs communicate with the system using a protocol that supports Master/Slave communication.
- Need a means of gathering power consumption data and the returns generated so far by mining.

Technical Challenges



Raspberry Pi Challenges

Manage communications between three entities:

Web Application, Cryptocurrency Network, and FPGA

Choosing algorithm needs to scan current price and make a decision on when to switch mining.

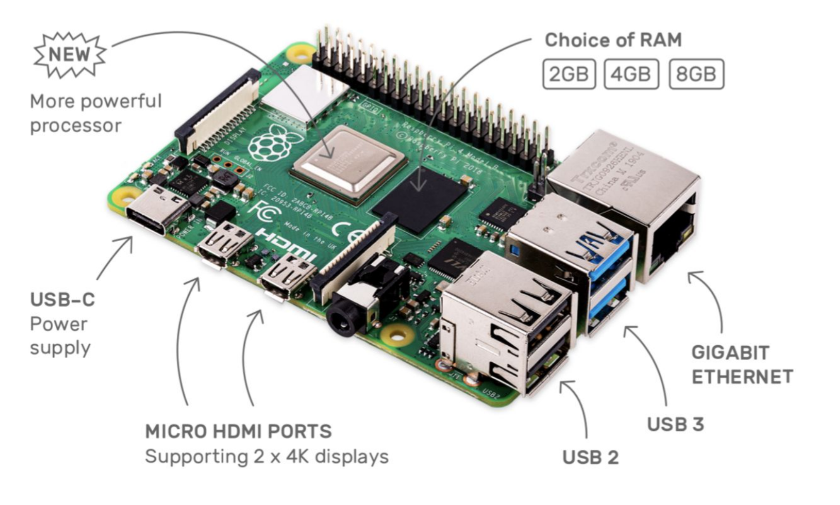
Monitor returns so far to ensure costs are being maintained

Solution Approach

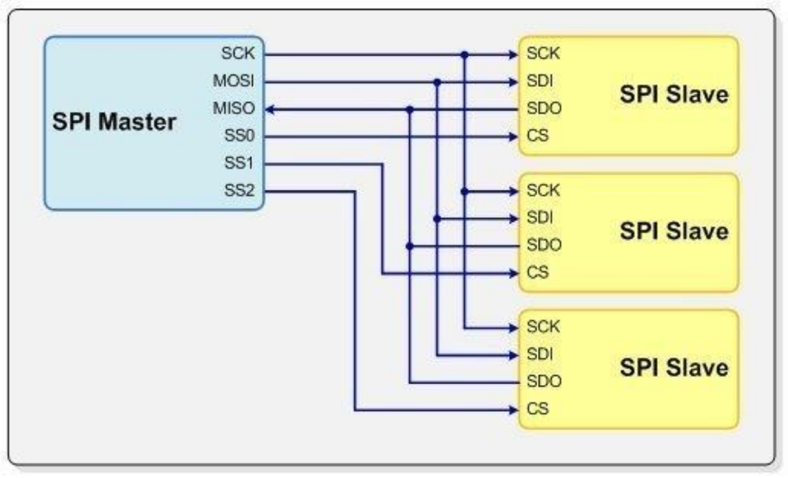
Django is a free and open source web framework we are using to develop our web application. It contains everything we need in one location and can be installed onto a Raspberry Pi.

For our FPGA, we wanted a low-cost board that still had good performance. The DE0-CV has half the logic elements of the DE2-115 at a third of the cost.

We chose to use a Raspberry Pi 4 Model B for our single board computer. Other RPI models did not have wired network connections nor the GPIO pins we wanted.



Solution Approach



We chose SPI as our communication protocol because it was capable of master/slave communication.

I2C is a half-duplex method which would limit the amount of bandwidth available to the system.

We are using Binance as our API for retrieving cryptocurrency prices. They offer a free service limited to 1,200 requests per minute. This is easily sufficient as our choosing algorithm would not run that frequently.

Testing, Validation, Metrics

To test performance, we will use the hashing rate, defined as the “number of hashes per second.” This can be obtained by using a counter to keep track of the number of hashes the FPGA has performed.

To test power consumption, we will use electricity usage monitor which monitor electricity usage when plugged into a wall socket. The power outlet of our FPGA will be attached to such a device which will give us a power consumption reading. Then we can use this data to calculate the electricity bill.

We will use the cost per FPGA in order to calculate the total cost of running the miners and combine it with the revenue generated from the miners to ensure a return of at least 5%.

Testing, Validation, Metrics

WebApp Requirement Testing:

To be within 10 seconds of configuration switching, the FPGA will output a signal once it finishes switching. We will look at the time from user submitting a new setting to the time this signal appears.

Ensure that the current day’s default configuration is truly optimal by comparing prices at the start of day.

Hashrate will be the average number of hashes across the last hour. Current prices will be retrieved every 30 seconds through the Binance API.

Tasks and Division of Labor

FPGA (David):

Create FPGA module for communicating with RPi (input)
Create FPGA module for communicating with RPi (output)
Create FPGA modules for mining bitcoin
Create FPGA modules for mining Ethereum
Create simple choosing algorithm

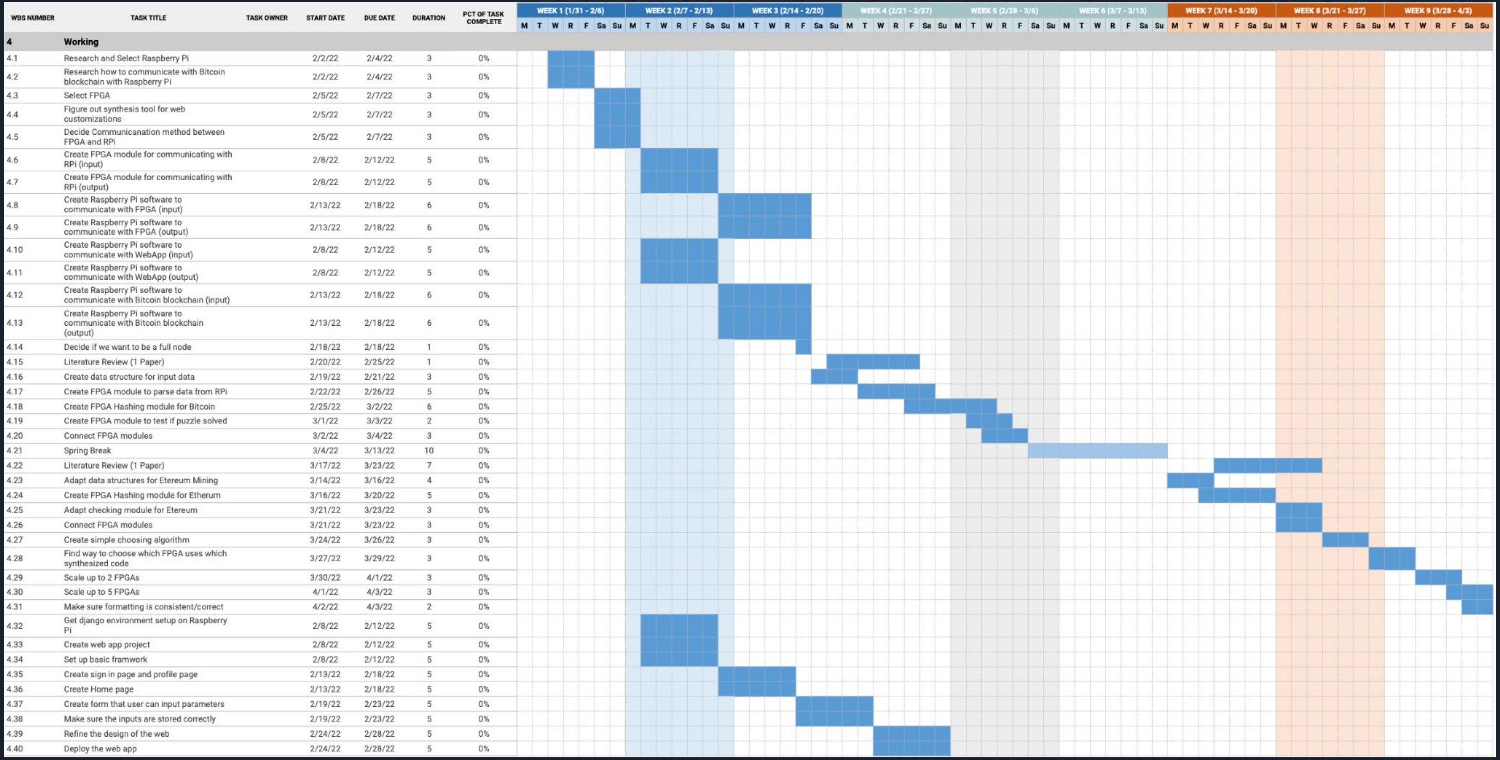
Intercommunication (William):

Create Raspberry Pi software to communicate with FPGA (input)
Create Raspberry Pi software to communicate with FPGA (output)
Create Raspberry Pi software to communicate with Web App (input)
Create Raspberry Pi software to communicate with Web App (output)
Create Raspberry Pi software to communicate with Bitcoin blockchain (input)
Create Raspberry Pi software to communicate with Bitcoin blockchain (output)

Web App (Lulu):

Create Web App
Send data to Raspberry Pi to communicate with FPGA
Display outputs from FPGA

Schedule



WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
4	**Working**					
4.1	Research and Select Raspberry Pi		2/2/22	2/4/22	3	0%
4.2	Research how to communicate with Bitcoin blockchain with Raspberry Pi		2/2/22	2/4/22	3	0%
4.3	Select FPGA		2/5/22	2/7/22	3	0%
4.4	Figure out synthesis tool for web customizations		2/5/22	2/7/22	3	0%
4.5	Decide Communicanation method between FPGA and RPi		2/5/22	2/7/22	3	0%
4.6	Create FPGA module for communicating with RPi (input)		2/8/22	2/12/22	5	0%
4.7	Create FPGA module for communicating with RPi (output)		2/8/22	2/12/22	5	0%
4.8	Create Raspberry Pi software to communicate with FPGA (input)		2/13/22	2/18/22	6	0%
4.9	Create Raspberry Pi software to communicate with FPGA (output)		2/13/22	2/18/22	6	0%
4.10	Create Raspberry Pi software to communicate with WebApp (input)		2/8/22	2/12/22	5	0%
4.11	Create Raspberry Pi software to communicate with WebApp (output)		2/8/22	2/12/22	5	0%
4.12	Create Raspberry Pi software to communicate with Bitcoin blockchain (input)		2/13/22	2/18/22	6	0%
4.13	Create Raspberry Pi software to communicate with Bitcoin blockchain (output)		2/13/22	2/18/22	6	0%
4.14	Decide if we want to be a full node		2/18/22	2/18/22	1	0%
4.15	Literature Review (1 Paper)		2/20/22	2/25/22	1	0%
4.16	Create data structure for input data		2/19/22	2/21/22	3	0%
4.17	Create FPGA module to parse data from RPi		2/22/22	2/26/22	5	0%
4.18	Create FPGA Hashing module for Bitcoin		2/25/22	3/2/22	6	0%
4.19	Create FPGA module to test if puzzle solved		3/1/22	3/3/22	2	0%
4.20	Connect FPGA modules		3/2/22	3/4/22	3	0%
4.21	Spring Break		3/4/22	3/13/22	10	0%
4.22	Literature Review (1 Paper)		3/17/22	3/23/22	7	0%
4.23	Adapt data structures for Etereum Mining		3/14/22	3/16/22	4	0%
4.24	Create FPGA Hashing module for Etherum		3/16/22	3/20/22	5	0%
4.25	Adapt checking module for Etereum		3/21/22	3/23/22	3	0%
4.26	Connect FPGA modules		3/21/22	3/23/22	3	0%
4.27	Create simple choosing algorithm		3/24/22	3/26/22	3	0%
4.28	Find way to choose which FPGA uses which synthesized code		3/27/22	3/29/22	3	0%
4.29	Scale up to 2 FPGAs		3/30/22	4/1/22	3	0%
4.30	Scale up to 5 FPGAs		4/1/22	4/3/22	3	0%
4.31	Make sure formatting is consistent/correct		4/2/22	4/3/22	2	0%
4.32	Get django environment setup on Raspberry Pi		2/8/22	2/12/22	5	0%
4.33	Create web app project		2/8/22	2/12/22	5	0%
4.34	Set up basic framwork		2/8/22	2/12/22	5	0%
4.35	Create sign in page and profile page		2/13/22	2/18/22	5	0%
4.36	Create Home page		2/13/22	2/18/22	5	0%
4.37	Create form that user can input parameters		2/19/22	2/23/22	5	0%
4.38	Make sure the inputs are stored correctly		2/19/22	2/23/22	5	0%
4.39	Refine the design of the web		2/24/22	2/28/22	5	0%
4.40	Deploy the web app		2/24/22	2/28/22	5	0%