

Efficient Method for Computing X16R Algorithm Hashes Using the VCU1525 FPGA

**Eleftherios Amperiadis
City College of New York
160 Convent Ave, New York, NY 10031
lefty.amperiadis@outlook.com**

Table of Contents

Executive Summary – Page 3

Objective – Page 4

Background – Page 4

Proposed Technical Approach – Page 5

Budget – Page 7

Milestones – Page 8

Current Progress – Page 8

Deliverable – Page 9

Executive Summary

With the recent advent of cryptocurrencies, individuals and large corporations alike are exploring the best means to involve themselves while simultaneously mitigating financial risk. Like standard currency, it is possible to purchase cryptocurrency on an exchange. The issue with this approach is that the cryptocurrency market is highly volatile. Like any other market, it is incredibly difficult to predict price action. The second method for acquiring cryptocurrencies is through mining. The term mining is misleading in this context, a more accurate word is auditing. A miner of a cryptocurrency is lending that crypto's underlying network computational power to verify all transactions that take place. In doing so, the network rewards the miner by paying him with cryptocurrency that is proportional in magnitude with the amount of computational power he has.

Mining can be achieved on most modern hardware. In the early stages of a cryptocurrencies life cycle, a Central Processing Unit(CPU) from a desktop is usually enough to confirm the transactions on the network. As the network grows and more people begin to use the cryptocurrency, dedicated Graphical Processing Units(GPU) are often used. The goal of this project is to develop mining software for the X16R algorithm that will run on Field Programmable Gate Arrays(FPGAs). The X16R algorithm is associated with a cryptocurrency called Ravencoin. As of the time of writing this proposal, it is the most profitable cryptocurrency to mine.

FPGAs are an ideal candidate in terms of hardware. This is because of the nature of the X16R algorithm. It is a combination of 16 different hashing algorithms that are randomly cycled. This is to prevent the use of Application Specific Integrated Circuits(ASICs) in a bid to keep the network decentralized. All this will be explained in more detail in upcoming sections. An ASIC, once built, cannot be changed. It is rigid in its capabilities. An FPGA, by contrast, can be changed as the programmer sees fit. As a result, an entire FPGA's hardware can be dedicated to hashing one of the 16 algorithms, and when a new one is selected randomly, the FPGA can be re-flashed to solve this newly selected hash algorithm as well. This grants the FPGA an immense advantage over GPUs and ASICs alike.

The ability to hash more quickly and more efficiently leads to a much higher validity rate on the Ravencoin network. This translates to higher rewards from the network, and enormous profits.

Objective

The objective of this proposal is to design software for the model VCU1525 FPGA from Xilinx that is highly efficient at solving all 16 hashing algorithms of the X16R algorithm and be able to switch between them as quickly as physically possible. The result of this objective is the acquisition of significant profitability in mining Ravencoin.

A secondary objective is to design software for all modern Nvidia GPUs, whose main objective is to solve all 16 hashing algorithms of X16R as efficiently as possible. The switching for the GPU platform is a trivial matter compared to the FPGA task.

Background

An FPGA is an integrated circuit that is designed to be configured by an individual or company after manufacturing. This is in stark contrast to CPUs, GPUs, and ASICs which come with all the circuits they will ever have from the manufacturer. CPUs and GPUs are more versatile than ASICs. An ASIC is designed to perform a few computations exceptionally well, while CPUs and GPUs are designed to perform many tasks well. Between a CPU and a GPU, the CPU performs far more generalized computations. A good way to think about the differences between all these integrated circuits in an anthropomorphic fashion is as follows: The CPU is all brains, the GPU is somewhere in between brains and brawn, and the ASIC is all brawn at a single task. An FPGA is unique in that it can be all of the above. It is fully programmable, and thus can be changed on the fly to compute the current workload in the most efficient means possible. So, why aren't FPGAs far more ubiquitous? The short answer is that they are difficult to make and more expensive than the other three categories discussed.

An FPGA is an ideal device to target the X16R algorithm precisely because it is a jack of all trades. The 16 different hashing algorithms will all have their own custom FPGA implementations, allowing for the performance of an ASIC miner.

Another goal of this project is to create a program that is also able to run on Nvidia GPUs through the CUDA parallel computing platform. Nvidia is one of the largest manufacturers and producers of Graphical Processing Units, used in both gaming and parallel computing tasks. They have created an entire platform dedicated to using their GPUs to their fullest potential. Later, in the proposed technical approach, a section will be devoted to how the CUDA API can be used to make a very efficient form of an X16R cryptocurrency miner. This version of the program will be especially useful to individuals who do not have thousands of dollars to spend on custom FPGA hardware.

Proposed Technical Approach

X16R uses the following 16 hashing algorithms: blake, bmw, groestl, jh, keccak, skein, luffa, cubehash, shavite, simd, echo, hamsi, fugue, shabal, whirlpool, sha512. The algorithms operate in chain fashion with the ordering dependent on the last 8 bytes of the hash of the previous block. The correspondence between each hex number and hashing algorithm is as follows:

| | |
|--------------|---------------|
| 0 = blake | 8 = shavite |
| 1 = bmw | 9 = simd |
| 2 = groestl | A = echo |
| 3 = jh | B = hamsi |
| 4 = keccak | C = fugue |
| 5 = skein | D = shabai |
| 6 = luffa | E = whirlpool |
| 7 = cubehash | F = sha512 |

So, if the hash of the previous block is:

0000000000000000000000007e8a29f052ac2870045ae3970270f97**da00919b8e86287**

The final 8 bytes:

0x7da00919b8e86287

Each hex digit (nibble) determines which algorithm to use next.

cubehash -> shabal -> echo -> blake -> blake -> simd -> bmw -> simd -> hamsi ->
shavite -> whirlpool -> shavite -> luffa -> groestl -> shavite -> cubehash

The first step in creating this miner is the create 16 independent programs that can solve hashes for the above hashing algorithms. When this is accomplished, the next step in the project is to create an overlaying software that handles the chaining order. For example, if the above sequence is generated from the previous blocks hash solution, the overlaying software should prime the FPGA memory with the independent programs mentioned above in correct sequence. If the program for a algorithm is








already in the FPGAs memory, it can be “flashed” to the FPGA far more quickly. In this context, flash means to change the FPGAs circuitry to reflect the new hashing algorithm. After this overlaying software is completed, the miner must be interfaced with the Ravencoin network for testing.

The three stages of development are therefore as follows:

- 1) Create 16 independent VHDL programs for the above 16 hashing algorithms.
- 2) Create a software that will operate on a computer’s operating system that determines what VHDL program should be running on the FPGA based on the chaining order of X16R.
- 3) Interface with the Ravencoin network for testing.
- 4) Refine all steps to increase overall hashing efficiency.

Another important concept is to understand exactly what the solution looks like for a given block. In the case of Ravencoin, like Bitcoin, the solution is a predetermined number of consecutive 0s in the beginning of the hashing function output. For example, if the difficulty corresponds to 3 leading zeroes, any string that hashes to this condition is a solution for that block. Once a miner finds the solution, that miner broadcasts the solution to the network for authentication. If the other miners agree that this miner’s solution is correct, the block reward goes to the first miner to compute the correct string. As the difficulty goes up, the number of leading zeroes goes up to. This ends up becoming a difficult computational problem. The more efficiently the miner can compute and check hashes for a algorithm, the higher that miners hash rate. A higher hash rate leads to a greater chance of discovering the solution at any given difficulty. Mining is the tradeoff between computational power, and electrical expenditure. If a miner can compute many hashes without burning a lot of electricity in the process, chances are it will be a profitable venture. There is also a third factor, which is the coin’s price. If the coin’s price is too low, it may never be worth it to mine it regardless of how efficient and powerful the miners equipment and software is.

A popular site to check these numbers is called, whattomine.com. This website takes a hash rate, power consumption and electricity cost in Kw/H as input and returns the most profitable coins to mine in that instance.

| Name(Tag) Algorithm | Block Reward Last Block | Difficulty NetHash | Est. Rewards Est. Rewards 24h | Exchange Rate | Market Cap Volume | Rev. BTC Rev. 24h | Rev. \$ Profit | Current 24h 3 days 7 days |
|--|---|---|----------------------------------|-------------------------------------|---|----------------------|-------------------------|-----------------------------------|
|  EthereumClassic(ETC) Ethash | BT: 13.92s BR: 3.88 ① LB: 7,041,182 | 148,233,064M 10.65 Th/s 9.3% | 1.5750 1.7219 | 0.00124700 (Binance) 3.9% | \$555,641,365 832.59 BTC | 0.00196 0.00215 | \$8.99 \$2.16 | 99% 106% 109% 109% |
|  BitcoinDiamond(BCD) BCD | BT: 9m 59s BR: 125.00 LB: 545,007 | 104,986.721 752.78 Gh/s 13.1% | 7.8287 8.8557 | 0.00024200 (Binance) 0.1% | \$155,716,629 55.30 BTC | 0.00189 0.00214 | \$8.97 \$1.91 | 95% 106% 93% 92% |
|  Ethereum(ETH) Ethash | BT: 14.25s BR: 2.91 ① LB: 6,815,385 | 2,466,150,003M 173.08 Th/s 1.7% | 0.0710 0.0722 | 0.02798700 (Binance) -0.3% | \$12,129,776,252 4,141.63 BTC | 0.00199 0.00202 | \$8.46 \$1.63 | 100% 100% 100% 100% |
|  Bitcoin Interest(BCI) ProgPow | BT: 10m 18s BR: 10.76 LB: 14,185 | 147,511,425.724 15.64 Gh/s -6.8% | 23.4955 21.9301 | 0.00009200 (HitBTC) -0.7% | \$7,056,051 0.19 BTC | 0.00216 0.00202 | \$8.44 \$1.32 | 109% 100% 92% 96% |
|  Nicehash-Ethash Ethash | BT: - BR: - LB: - | - 7.86 Th/s 3.0% | 0.00198 0.00201 | 2.83940000 (Nicehash) -1.8% | - 22.00 BTC | 0.00198 0.00201 | \$8.43 \$1.60 | 100% 100% 100% 102% |
|  Ravencoin(RVN) X16R | BT: 59s BR: 5,000.00 LB: 474,632 | 67,937.957 4.95 Th/s -1.1% | 429.3228 424.7995 | 0.00000460 (Binance) -2.1% | \$45,683,938 347.61 BTC | 0.00197 0.00195 | \$8.18 \$1.19 | 99% 97% 98% 99% |
|  Pirl(PiRL) Ethash | BT: 11s BR: 6.00 LB: 2,530,844 | 1,970,422M 179.13 Gh/s 0.6% | 182.5330 183.6743 | 0.00001047 (Cryptopia) ? real | \$1,154,224 0.78 BTC | 0.00191 0.00192 | \$8.05 \$1.22 | 96% 95% 90% 84% |

The cryptocurrency market as of December 1st 2018 is at an all time low for the year. As a result, the profitability of mining for instantaneous profits is quite low. The above results shows the most profitable coins to mine with 12 GTX 1070 and 12 GTX 1060 Nvidia GPUs.

Budget

The FPGA that is ideal for this is the VCU1525 from Xilinx. It has a PCIe express interface, which allows it to be slotted with any modern motherboard. Furthermore, running some preliminary numbers yields 300 MH/s at 300 watts. In this context, MH/s is mega-hashes per second. This translates to \$22.20 per day of Ravencoin at current price and difficulty. At the MSRP cost of \$3600, this leads to a return of investment in 160 days.

An academic version of this algorithm can be demonstrated on most FPGAs. The biggest bottleneck on cheaper FPGAs is the overall bitstream size. It is possible that a complete implementation of a particular hashing algorithm will not fit on a cheaper FPGA without reducing performance. Regardless of this fact, models such as the Cyclone 4 can be used for testing purposes. The performance to cost ratio is likely not to be favorable in this test model. However, it acts as a good test bed for future bitstreams on more sophisticated hardware.

Another version of the program will be available for Nvidia GPUs as well. The initial cost of a GPU is very low when compared to an FPGA. A GTX 1060 can be acquired for 150 USD as of December 1st 2018. This can be a test bench for the academic version of the project, as well as a profitable GPU to mine with.

Milestones

Mid January: Implement all 16 hashing algorithms in C to demonstrate the ability to program said algorithms. Port the code to CUDA 9.2 API for Nvidia GPUs.

Mid February: Translate the 16 hashing algorithms to VHDL, and test all of them on an FPGA individually.

Mid March: Create an overlaying software that can dynamically reconfigure the FPGAs structure based on the correct chaining order as described in the Technical Approach section above.

Ongoing: Refine the algorithms to increase performance and increase profitability.

Current Progress

There is a demonstration C++ program that is essentially a Bitcoin miner. This program makes use of the OpenSSL library and generates an SHA-512 hash from a random input. The program accepts one argument, which is a simplified version of the difficulty. The difficulty is an integer between 1 and 64 that corresponds to the number of consecutive 0s in the hash output that is needed to solve that particular block. If the user enters a number like 4, the hash output must have 4 leading 0s.

SHA512 digest:

```
00005675b3d66158f8215122c2c62c4768faed3fd8ed4085f85beeb95512719153f04bfaf5000fab4343a5a6ccbacc9da57b3f7facff6b2a1fd9ec0485a70e
```

The string that generated the above hash will solve this block. Ravencoin has an additional 15 hashing algorithms that are randomly selected, as reviewed in the Technical Approach section. To complete the first milestone, all 16 must be implemented in C or C++. The estimated hash rate of the program is anywhere between 100,000 H/s to 160,000 H/s running on a Intel 6200U CPU. In the coming weeks, optimizations will be explored in C to increase this number. The utilization of a GPU can greatly increase the ability of the algorithm to hash more efficiently. As of right now, the highest solvable difficulty in a decent amount of time is 5 leading zeroes.

A VHDL version of the SHA-512 algorithm has also been tested. The intention of this project is to offload all the hash generation to the FPGA and check the result on the associated computer. Ideally the FPGA can do better than 150,000 H/s in SHA-512.


```
methos@deus:~/Documents$ ./sample1 4
Passed
String: ShEiF2sH00000?V
SHA512 digest: 0000d8c21b58767ae783ca2507ed04c69b125d36d0dd390ae3e63a55a9c1dcfb2
66f42aa97742efadff50254e0f28175862565c6fb7eb9d28e2e66f722006f39
Attempts: 72806
Hash Rate: 110302.093749 H/s
```

Deliverable

The result of this experiment will be a bitstream that is able to appropriately mine all 16 hashing algorithms of X16R, and an overlaying software that is able to intelligently reconfigure the FPGA's bitstream for the appropriate hashing algorithm.

There will also be a GPU variant of the above algorithm. Nvidia CUDA library 9.2 will be used to compute the cryptographic hash functions and attempt to solve the requisite problem. The advantage of this is that anyone who can afford an Nvidia GPU is able to test the program and mine Ravencoin with it.