Memory Dependant PoW Hashing Algorithm
Demonstrating Economic Infeasibility for Mining ASICs

AlterHash (aka XDag16R)

By Arsen Julhakyan

Altered Silicon, Inc.

Introduction

Feasibility Study:
An analysis and evaluation of a proposed project to determine if it (1) is technically feasible, (2) is feasible within the estimated cost, and (3) will be profitable.

ASIC (Application Specific Integrated Circuit) chips can be very powerful at hashing and very cheap to produce. This has been shown to create centralization dynamics for blockchain ecosystems. 51% attacks, only dreamed of when bitcoin first got its start, have been utilized to steal from exchanges and destroy upstart blockchains. Centralization of hashing power and mass manufacturing of high powered ASIC miners puts network consensus at risk. A recent article on Coindesk suggests that as much as 60% of bitcoin's network hashrate resides in China:
https://www.coindesk.com/highest-in-2-years-65-of-bitcoin-hash-power-is-in-china-report-finds

Let us be clear: in technology and hashing, ASICs are **always** technically feasible. In producing this paper we are not attempting to say that the first premise of feasibility is not true.

Economic Infeasibility Explained

This proposal seeks to explore an ASIC resistance concept we call Economic Infeasibility(EI). EI resistance asserts that, based on the number, type and size of the algorithms used in the PoW protocol, we can force an arbitrarily high risk of design and manufacturing difficulty as well as cost in an ASIC attempt. This puts the second premise of feasibility in question and creates uncertainty about the third: profitability. A primary element of EI is to force the use of a common component already well advanced and mass produced by a large non-crypto company, such as High Bandwidth Memory (HBM). EI doesn't try to produce algorithms that are too difficult to replicate in ASIC, it simply arranges and orders them in such a way that an ASIC would not necessarily outperform reprogrammable computing hardware such as CPU, GPU and FPGA well enough to justify the investment.

By creating a dependency on external memory, all attempts to create a miner will be forced to optimize performance by using the same basic components. For example, it would be a massive undertaking to try to design better HBM memory than Samsung, Micron and SK Hynix. So most mining hardware will implement their memory, and achieve similar performance with no outsized advantage relative to other hardware. Furthermore, mining hardware will likely take the form of combinations of CPU, GPU and

FPGA devices, which fundamentally builds a different type of network - one that can be utilized for programmable computing.

Hashing Hardware

Since FPGAs are typically the engineering precursors to ASICs, an algorithm that is highly compute intensive and feasible on FPGA confers a large advantage to ASIC approaches. However with the advent of integrated on-chip HBM (High Bandwidth Memory), CPUs, GPUs and FPGAs can all implement memory bound algorithms to provide relatively equivalent PoW hashing performance for a blockchain. CPU, FPGA and GPU are all capable of significant compute outside of just hashing, and in many cases can provide their compute without interrupting the mining process - thus presenting an offsetting value proposition to single-purpose ASIC cryptocurrency mining hardware.

The overall idea of this proposal is to find a way to lower the efficiency gap as much as possible between ASICs and commonly available hardware types like CPUs, GPUs, FPGAs to the level where even if ASICs join the network, the likelihood of a single company taking control of more than 50% of network hashrate is minimal. We are not aiming to level things up on manufacturing cost, as there are always going to be cost differences that we can't plan for in different chip/board combinations. We are also not aiming to give any particular hardware type significant advantage over another one. The idea is to find the best balance possible in terms of price / performance / efficiency and risk among mentioned hardware types.

As all hardware types are designed for different applications, there are fundamental architectural differences between them, hence finding a single algorithm that could make them equivalent is nearly impossible. Essentially, by forcing different hardware types to be bound by external memory ensures that at least on efficiency ASICs cannot have significant advantage, which consequently adds higher risk and price for an ASIC company to try to take control of the network.

The inspiration for this ethash-x16r mashup comes from the fact that the Ethereum network resisted ASICs for quite a few years, though ultimately ASICs were introduced there. Ethereum is still mineable by common GPUs, and there are multiple reasons for this phenomenon, which we will talk about in this paper, but the main one of them is the low efficiency gap between ASIC miners and GPUs. Ethash algorithm is considered a memory "hard" algorithm and our aim is to take all the good knowledge and lessons learned from it, combine it with the randomization idea behind X16R to cover possible flaws identified in ethash and conclude with a new PoW algorithm we call "AlterHash".

Current generation CPUs don't have access to high memory bandwidth except multi-slot DDR4, however, there is a very high chance that HBM memories with huge memory bandwidth and less power consumption are going to be common place with future CPUs, hence, this type of hardware should be able to join the network in the future. GPUs don't have issue with high memory bandwidth, as they are designed for such tasks. HBM based FPGAs can work alongside the latest GPUs on the proposed Alterhash algorithm. Even FPGAs which are limited to DDR4 memory can mine the proposed algorithm, albeit the economics may favor other algorithms.

Acknowledgement

I would like to take this opportunity and express my gratitude to **Traysi**, **WF**, **GPUHoarder**, my good friend **Eureka**, my business partner **Dave Carlson (buzzdave)** and the whole **Ravencoin community** for the valuable advice, healthy criticism and encouragement during the past 2 months. It was a great

learning process and I am very much thankful to everyone who helped to shape the algorithm as much as possible.

Alterhash - The Algorithm

This paper is organized in the following way:

1) First we will talk about current X16R/X16RV2 algorithms and their vulnerability to potential ASICs taking control of the network.
2) We will go in depth of how current Ethereum algorithm Ethash is functioning and what are potential "flaws" which should be fixed to increase the chances of ASICs not having much choice but to use common external memory.
3) We will propose solutions to the Ethash "flaws" and how a combination of X16R can solve those problems.

The Ravencoin Blockchain

X16R/X16RV2

X16RV2 and its predecessor X16R are essentially composed of 16 different primary hash functions. The length of the chain is 16 and at each location in this chain, any of the 16 hash functions can be randomly selected for each new block target. Hence we end up with an enormous amount of possible chains equal to 16 ^ 16.

Hash functions used in Ravencoin's X16R algorithm:

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

Table 1

In case of X16RV2, Tiger192 was added at the beginning of keccak, luffa and sha512 algorithms, which was done specifically to break RVN mining for a known set of ASICs on the network. X16RV2 was forked in by the Ravencoin community as a stop-gap to known ASICs which were having a negative impact on the network.

A Ravencoin ASIC chip design

The X16R algorithm adds complexity into potential ASIC implementations. In comparison to a simple X11 type of algorithm, where 11 hash functions are chained with fixed connections, an X16R ASIC implementation requires a smart sequencer to be able to connect hash cores in the right order

depending on the chain generated upon each new block. On top of that, repetition of the same function within the chain target either requires multiple cores of the same hash function to be present (worse case 16 x 16 = 256 cores on the same die) or otherwise the performance (hashrate) of a potential ASIC will be limited by the factor of the longest repetition in the chain. However, the statistical distribution of chain targets generated on the Ravencoin network shows that about 95%-96% of all possible chains contain a maximum of 4 repetitions of the same hash function. This means that a potential implementation of an X16R ASIC could be quite effective with a 64 core + complex sequencer architecture, and would be able to mine 95% of all chains of X16R algorithm.

To be able to run price / performance / power analysis of a potential ASIC, we have decided to take 28nm technology as a playground for our estimation. Considering the current market conditions and emission rate of Ravencoin, 28nm seems like the most realistic target for a company intending to create an ASIC for X16R/X16RV2 mining. We are presenting here the synthesis results of all 16 cores. For reference -> 1mm^2 == 1.000.000 um^2.

Algos	Technology	
	28nm	
	Area(um^2)	Power(mW)
Blake	348,666	23
Bmw	333,257	18
Cubehash	1,065,471	53
Echo	885,209	79
Fugue	576,546	68
Groestl	316,050	34
Hamsi	564,660	29
Jh	390,455	28
Keccak	243,298	13
Luffa	381,413	17
Sha512	364,428	19
Shabal	384,678	26
Shavite	423,684	31
Simd	1,123,478	60
Skein	950,512	41
Whirlpool	398,736	29
Average	546,908	35.5

Table 2

All cores are fully unrolled, which means each core can run at full speed of hash / clock cycle.
Clock frequency was constrained to 500Mhz.
We also synthesized a very dumb sequencer, which has a capability of connecting each core to any of the remaining 63 cores:

Technology
28nm

Area(um^2)	Power(mW)
781,986	171.176

Table 3

Clock frequency was again constrained to 500Mhz.

On average each core occupies roughly 0.55mm^2 area. 64 cores will occupy 64 x 0.55 = 35.2mm^2 area. The power numbers are tool estimations and are very likely to be wrong, so to compensate for a potential under-estimate, we will simply multiply final power number by 10x to make sure we safely over-estimate power consumption of this chip design.

We can assume that 64 cores + dumb sequencer occupy roughly 36mm^2 area. Obviously real numbers can be obtained only after full ASIC development cycle but final die area numbers would be roughly 30% more after Place & Route process, so putting that in place we get a chip die roughly 50mm^2 in size, which is able to generate 500Mh/s performance, as all cores are fully unrolled.

The estimated power would be ((0.035W x 64) + 0.172W) x 10 (our conservative over-estimate), equalling 25W per chip.

Ravencoin ASIC ROI analysis

The cost of a TSMC 300mm 28nm wafer is roughly $4,500 per wafer. Each wafer can produce roughly 1,000 chips of 50mm^2.
The cost of production of such a chip would be $4,500 / 1,000, or roughly $4.50 per die.
Let's assume with chip packaging and testing the cost settles at $10 per chip (also very conservative).
To summarize, we end up with a die consuming 25W at $10 of cost, producing 500MH/S.
Let's assume an Antminer S9 type of device can have 15 such chips (5 per hashing board). The overall hashrate of a miner like this would result in 7.5 GH/s at a cost of 15 x $10 + PCB + Case + PSU = roughly $300. Let's again be conservative and assume a total production cost of $500 per device. Each device will consume probably 500W, but let's say we made further mistakes and the actual power consumption settles at 1500W.
So in summary we have a machine producing 7.5GH at 1500W and having production cost of $500 per unit.
To hit half of Ravencoin network we would need to produce roughly 20,000 of these 500MH chips or roughly 1300 x 7.5GH devices.
The cost of producing such devices would be a mere $650,000. Overall power consumption would be roughly 2MW - a small crypto mine by today's standards.
So let's see how much time will be required to ROI such an investment.
The TSMC 28nm full maskset cost is roughly $2M. Tools + engineering say another $500K. IPs + device production cost say another $1M, so overall this is roughly a $3.5M investment.
At current market price of $0.025 per RVN coin, the Ravencoin network emits $180,000 USD per day.
When hitting 50% of network hashrate, very likely, a substantial number of GPUs will be forced off the network and we can infer that the ASIC project gets control of 65% of overall hashrate.
Gross profit per day = $180K USD x 0.65 x 0.95 = $111,000 USD
Let's assume the project operates at an average of $0.05 per KW/h electricity cost, which is high compared to most 2MW mining projects today.

Net profit per day = $111,000 – (2,000kW x 24 x $0.05) = $111,000 – $2,500 = $108,000 USD per day of net revenue in the form of Ravencoin.
$3.5M / $108.5K = roughly 33 days to ROI the investment, if the mined RVN could be all sold at once. And in some cases large block selling is possible, for example to a private investor or exchange operator seeking to build a trading book. However, as time goes on, the surplus RVN finds its way onto public trading exchanges, and this massive surplus adds pressure to prices. As prices fall, *all* miners begin to accelerate their selling in order to cover their power costs. This is how the presence of ASICs can create a downward spiral for coin prices and a “race to the bottom”, until the surplus coins are absorbed by the market and demand for coins begins to support prices again.

Ethash

Ethash is a “memory-hard” algorithm, which is supposed to be limited by external memory bandwidth, where every 30,000 blocks (epoch length) a new Directed Acyclic Graph (DAG) is generated.

How Ethash works - DAG generation

1) Epoch number = Block number / Epoch length.
2) for (int i = 0; i < epoch_number; ++i)
   epoch_seed = keccak256(epoch_seed);
3) Based on the seed, a pseudo random Multi-MByte Light Cache is being generated. Light clients store the cache.
4) From the cache, we can generate multi-GByte dataset, with the property that each item in the dataset depends on only a small number of items from the cache. Full clients and miners store the dataset. The DAG dataset grows linearly with time.
5) Mining involves grabbing random slices of the dataset and hashing them together. Verification can be done with low memory by using the cache to regenerate the specific pieces of the dataset that one seeds, so one only needs to store the cache.

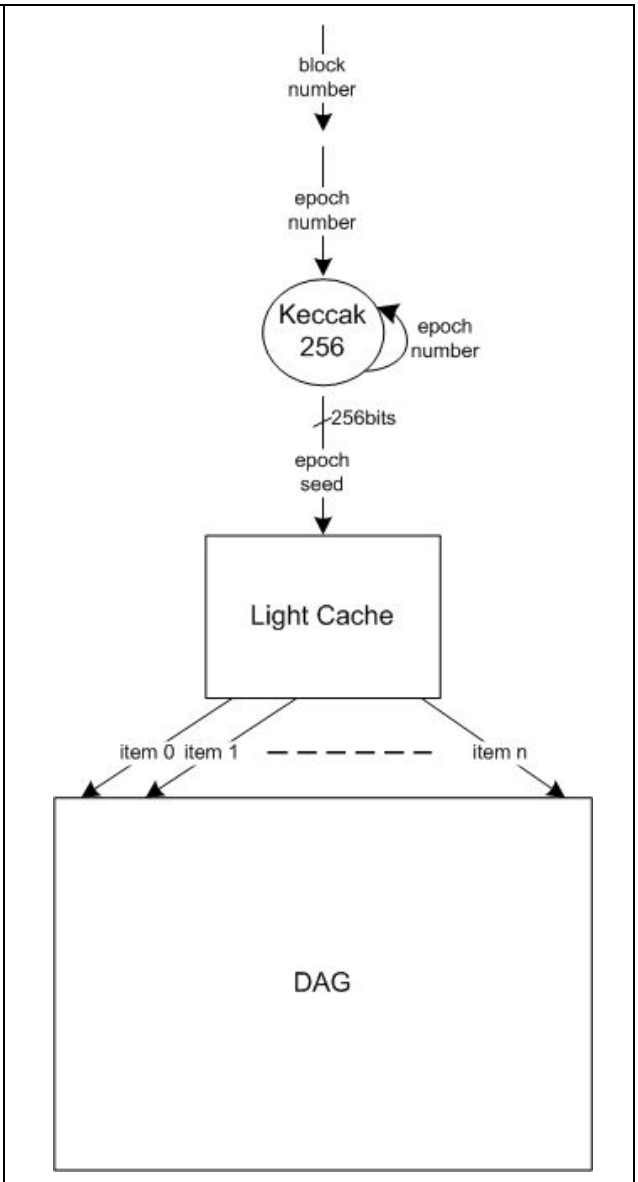

Figure 1 depicts the overall structure of how the final multi-GByte DAG is generated.

The Light cache size at genesis block was 16MB and grew by 128Kbyte per epoch, hence currently equals to 16MB + 37.5MB = 53.5MB. The DAG size at genesis block was 1GB and grew by 8MB per epoch, hence it currently equals 1GB + (8MB x 300) = 3.4GB. The increase of DAG size is done on purpose to make sure that technological improvements over time in practice cannot give an advantage to ASIC companies to put the DAG onto a single chip SRAM, hence ethash remains bottlenecked by external memory bandwidth.
Please refer to Ethash github repository for more information of algorithm functionality:
https://github.com/ethereum/wiki/wiki/Ethash

How Ethash works - DAG items

Each DAG item has got an **index**. Say first 512bits are at index 0. The next 512bits at index 1 and so on.

There are 3.4GByte / 512bits = more than 53M indices representing each item in today's Ethereum DAG.

To calculate a particular DAG item we pass through the following steps:

1) 32bits of cache[index%num_cache_items] is XOR-ed with index and then concatenated with the original 480bits of same cache[index % num_cache_items] item.
2) First **mix** is generated through Keccak512 hash function.
3) Next cache item index is generated by taking 32bits of mix and passing it through FNV1 function, which is the following:
   def fnv1(v1, v2):
   return ((v1 * 0x01000193) ^ v2) % 232**
   where
   v1 = index ^ round
   v2 = mix[round % num_words], where num_words = 512bits / 32bits = 16
4) **New mix** is generated by combining newly fetched cache[new_cache_idx] with previous mix through the same FNV1 function.
5) Steps from 3 to 4 are repeated 256 times, where **round** is integer number representing particular round.
6) **DAG[index]** item is generated by passing **final mix** through Keccak512 hash function.

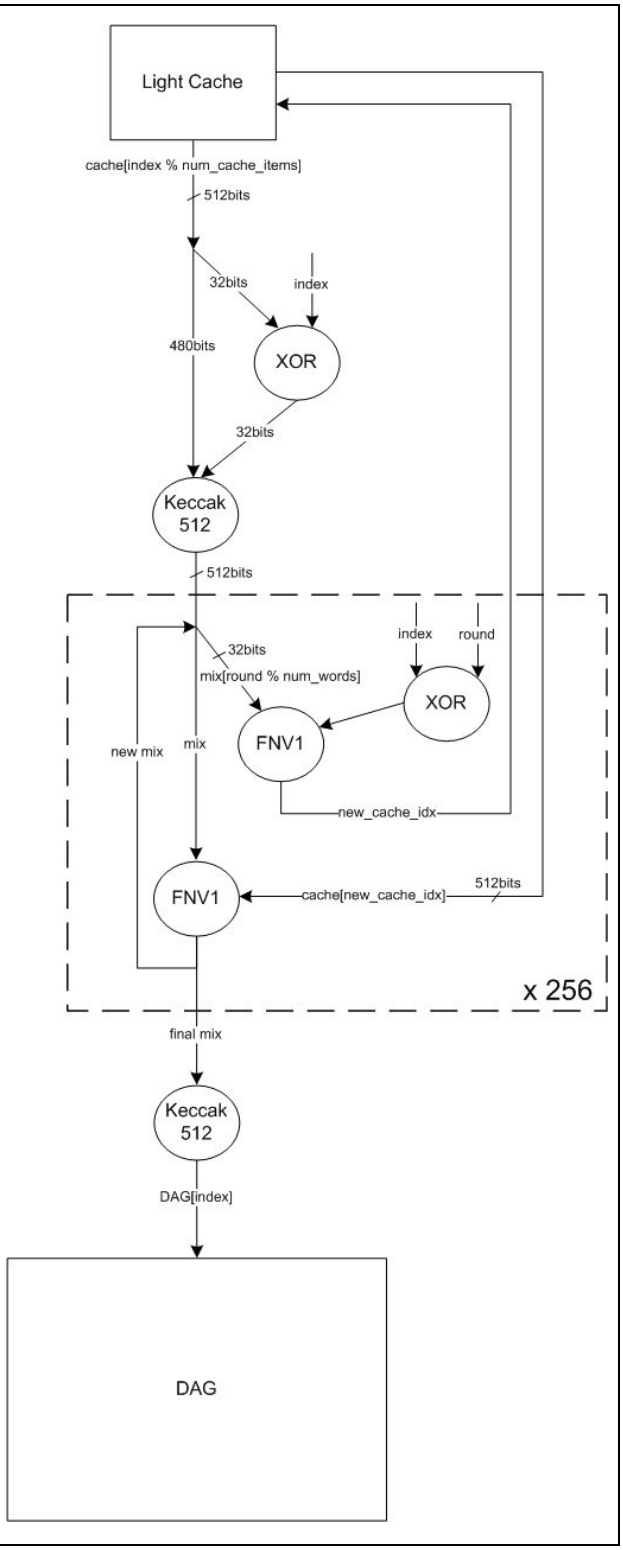

Figure 2 depicts detailed structure of how each DAG 512bit item is generated based on selected 512bit items from Light Cache. We are going to propose changes into this structure; hence proper understanding of this part of algorithm is required.

Ethash hashing

As the whole DAG calculation is based on block number, it can be fully calculated well ahead before block headers reach the particular block number. There are roughly 5.2 days between epoch changes.
Now let's have a look at how block header hash is being calculated in ethash algorithm:

1) 512bits **hash_seed** is the result of Keccak512 hash function over 256bits **header_hash** concatenated with 64bits **nonce.**
2) In this case starting **mix** is 1024bits long data composed of simply concatenating two identical hash_seeds.
3) New DAG index is calculated from combining mix[round % num_words] with **hash_seed[0] ^ round** through the same FNV1 function. Here num_words = 1024bits / 32bits = 32. hash_seed[0] is the 32 LSBs of hash_seed.
4) **New mix** is calculated from combining newly fetched DAG[new_DAG_idx] with previous mix through the same FNV1 function.
5) Steps 3 to 4 are repeated 64 times, where **round** is an integer number representing particular round.
6) **mix_hash** is 256bits number generated from running 3 x FNV1 functions over **final_mix.**
7) Keccak256 hash function over concatenated hash_seed with mix_hash generates **final_hash.**

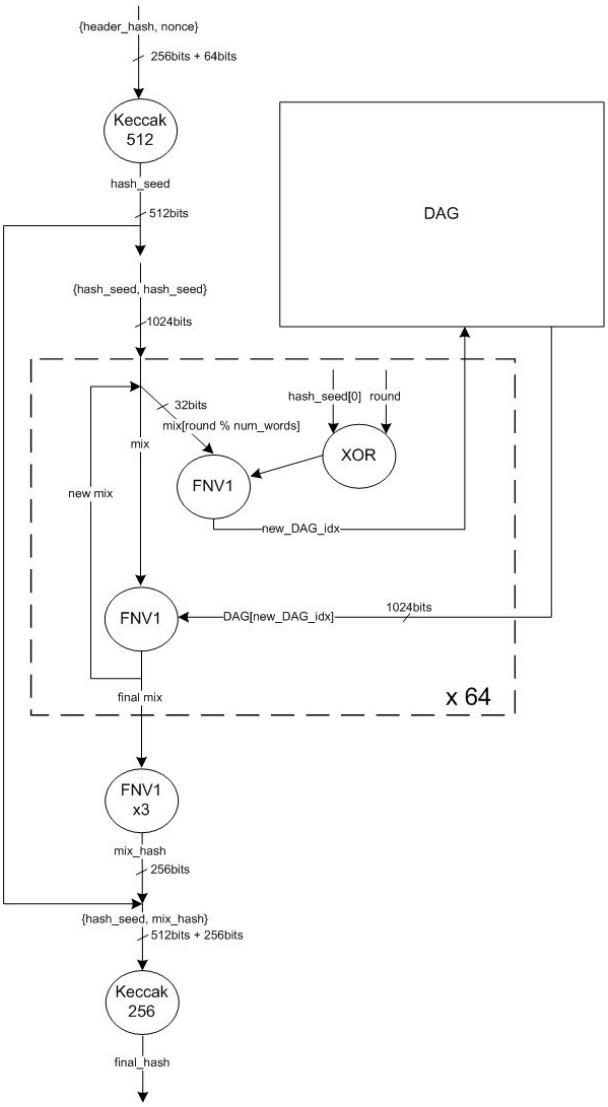

Figure 3 ETH Hashing

There are 3 potential ASIC "tricks"which can give an advantage to an ethereum ASIC to somehow overcome memory hard part of ethash algorithm.

1) The first one is so called "light evaluation method", where instead of requesting a DAG item from external memory, a potential ASIC can simply calculate each DAG item on the fly by just keeping light cache of around 50-60MByte on chip (Figure 2).
2) The second problem is a rumored issue with the FNV1 function, where instead of making a single memory request per nonce, an ASIC with relatively big on chip SRAM can keep millions of nonces' partial results, where each is 1024bit long, and combine multiple memory requests into one, as the FNV1 function creates memory collisions (multiple nonce calculations at various stages require same DAG item to move forward). Hence the speedup can be the average number of collisions found during the hash operation (Figure 3).

3) The third problem is the tactic of splitting the DAG over multiple chips and connecting them through high speed communication channels, hence avoiding using external memory. https://linzhi.io/ is a great example of such design and we will talk about it in detail.

Alterhash

Let's have a look at the issues mentioned above and see how this proposed algorithm is going to either totally solve or minimize them.

Light evaluation method

Let's assume one tries to make an ethash ASIC, which simply uses around 100Mbyte of on chip SRAM to keep the Light Cache on die and calculate each DAG item on the "fly". Say the idea is to make a chip producing something like 500Mh/s on Ethash running at 500Mhz. We will compare results with an NVIDIA 1080TI, which is making roughly 50Mh/s at Ethash algorithm.

To be able to produce 1024bits every cycle by light evaluation method, an ASIC must additionally contain:

1) Roughly 100MByte SRAM
2) 2 x 64 x 2 x (Keccak512 unrolled hash core) = 256 x Keccak512 unrolled hash cores.
3) 2 x 64 x 2 x 256 x 17 (very simple FNV1 core) = 1,114,112 x FNV1 cores.
   (There are two FNV1 functions depicted on Figure 2, however one of them works on 512bits, which in reality requires 512bits / 32bits = 16 FNV1 functions working on 32bits operands, hence number 17 in the equation above.)

This already looks scary requiring big on chip RAM and decent amount of compute resources. However, there is also a bottleneck created due to the enormous amount of SRAM bandwidth required to sustain 500Mh/s speed.
Essentially to be able to produce one hash per cycle, where each cycle is 500Mhz, an ASIC chip must access the distributed SRAM: 2 x 64 x 256 = 32,768 times per cycle. The current Ethash light-cache contains 53.5MB / 512bits = 876,544 possible locations. There is a small possibility of some requests going to the same location, so let's assume conservatively 32,768 requests end up being twice less -> 16,384.
The Xilinx VU9P 16nm multi-die FPGA contains 4,320 18bit wide BRAMs + 960 72bit wide URAMs. Essentially even such a big chip with very heavily distributed 4,320 x 18Kbit + 960 x 288Kbit = 44MB of on-chip RAM could serve a maximum of only 18 x 4,320 + 72 x 960 = 146,880 bits per cycle.
The requirement to achieve 500MH/s in the case of Ethash is 16,384 x 512bits = 8,388,608 bits per cycle, which is like 55 times more than even the relatively large VU9P can do.
It's pretty obvious from the calculations above that a single ASIC could never target something like 500Mh/s per chip. However, maybe it is possible to decrease the requirement by 10x and produce an ASIC generating 50Mh/s, instead of 500Mh/s. This should decrease compute and SRAM bandwidth requirements by 10x, however, still force an ASIC to keep 100Mbyte of on chip RAM.
Even though it would still have complications in terms of justifying cost and design effort, there is room for hope. To make sure there is no hope left, we are proposing the first modification to the original ethash algorithm:

As can be seen in the figure, there is a full X16R chain added after the first keccak512 as well as another full X16R chain after final mix is calculated, before the last keccak512. The chain order of the first X16R is decided based on the 64bit LSBs of the first keccak512 result, whereas the order of the second X16R chain is decided based on the 64bit LSBs of final_mix. As a result each DAG item calculated passes through two totally different chains.
In addition to double X16R chains, a 32bit FNV1 calculating a new cache address is run through the relatively small siphash hash algorithm: https://131002.net/siphash/siphash.pdf

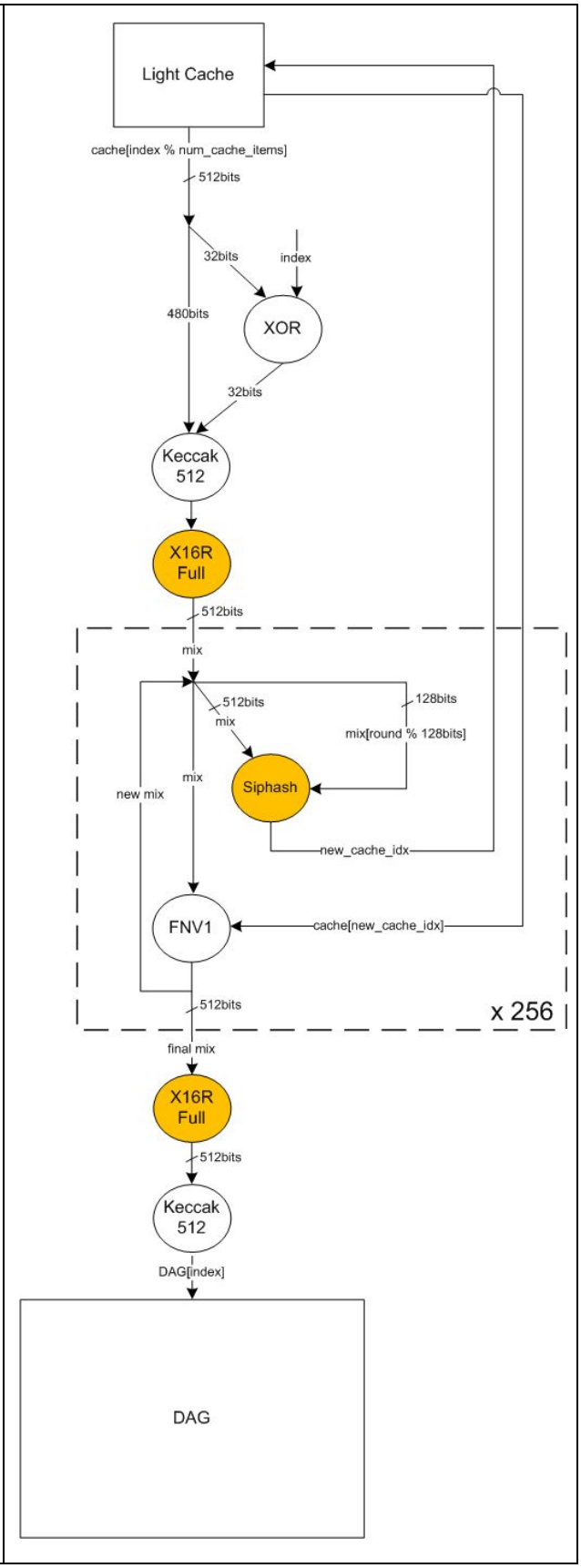

Figure 4 Proposed DAG item generation

Siphash takes mix as a 512bits message, another 128bits of the same message as a key depending on the round and produces a 64bit result, which serves as a new cache item address. The proposal is to use Siphash-2-4 version.

Addition of totally dynamic X16R chains into each DAG calculation, as well as the more complicated siphash function into the DAG item calculation process makes it so difficult and compute intensive that the Light Evaluation method becomes no longer practical on any realistic ASIC node, even probably including one of the latest: 7nm technology. However, our approach does not affect hashing functionality, and hence GPU capability is not affected, because it is assumed that the full DAG will be precalculated offline. Lets run numbers again for 50MH case:

1) Roughly 100MByte SRAM
2) 2 x 64 x 2 x (Keccak512 unrolled hash core) / 10 = 25.6 x Keccak512 unrolled hash cores.
3) 2 x 64 x 2 x 256 x 16 (very simple FNV1 core) / 10 = 104,857 x FNV1 cores.

We removed only single 32bit FNV1 function; the remaining 16 are still there for combining the mix.

4) 2 x 64 x 2 x X16R (16 random cores) / 10 = 410 various dynamic cores.
5) 2 x 64 x 2 x 256 (Siphash-2-4) / 10 = 6,550 Siphash cores.

The amount of compute and complexity is so large that any ASIC attempting to use the Light Evaluation method simply becomes impractical.

Memory access collisions

The tactic here is to combine multiple memory accesses into one and hence improve produced hashrate by a factor of k, where k is the average number of memory accesses combined during hashing operation. To be able to solve the problem, one needs to understand the problem. Hence, we first decided to run a series of simulations to see how many memory collisions the FNV1 function can produce during Ethash hashing operation.

As a starting point let's assume we have 128MB of on-chip RAM and we try to have roughly 1M midstates active at all times due to the fact that the Ethash midstate is a 1024bit mix. Essentially 1M x 1024bit equates to more than 100MB of on-chip memory space.

For our example, we are taking epoch number 295, where the DAG size is roughly 3.4GB and had 3.4Gbyte / 1024bit = 27M various locations, each 1024bits long.

1M nonces will produce 64M different addresses. Now, among 64M accesses, the best distribution is 64M / 27M = 2.37, which means the ideal algorithm cannot avoid but being hit by a speed up factor of at least 2.37. Obviously if you have possible 27M locations and you generate random 64M requests there is 100% going to be collision (multiple requests going to the same location). So the distribution produced by the ideal algorithm would be as near to 2.37 as possible.

Let's see what FNV1 is producing on randomly selected 256bit header_hash + 1M nonces starting from 0. As mentioned, in order to hash all 1M nonces, we will need to generate 64M different memory addresses:

Repetition Count	Memory Accesses	Repetition multiplied by memory accesses
14	2	28
13	20	260
12	129	1,548
11	698	7,678
10	3,260	32,600
9	13,984	125,856
8	55,288	442,304

7	191,756	1,342,292
6	578,356	3,470,136
5	1,506,349	7,531,745
4	3,258,950	13,035,800
3	5,652,212	16,956,636
2	7,347,364	14,694,728
Total	**18,608,368**	**57,641,611**

Table 4

Essentially **k=3: 5,652,212** means among 64M accesses there were 5,652,212 different address accesses which have been repeated 3 times. The rest follows the same logic. Now let's see how many accesses we have been able to optimize taking advantage of memory collisions.
Instead of 57,641,611 total memory accesses, we end up generating only 18,608,368 memory accesses. There is remaining 64M – 57.6M = 6.4M, which we were accessing only once. Hence the overall maximum theoretical possible speedup is 64 / (6.4 + 18.6) = 2.56x.

Those calculations show that the idea of FNV1 being broken does not seem to be very real, because 2.56x is not that far from the possible best scenario of 2.37x.

However, this result is still not a realistic scenario, because 64M accesses combined together would require 64 x 128MByte of SRAM, which is also not practical.

Full 1024 bit mix

In the case of ethash (see Figure 3), each next DAG item address is generated based on less than 32bits of the 1024bit mix at a time. Despite this, we don't think this is really an issue hiding some tricky optimizations. For example, at round 20 of 64 iterations in the generation of a next DAG item address, we would still need 32bits of all 20 previous mixes (bits 639 to 608 in 1024bit mix).
In any case, with Alterhash we try to eliminate any possibility of optimizations here by making sure that each DAG item at a time is generated based on a full 1024bit mix instead of only 32bits. The result can be seen in the Alterhash hashing logical diagram, depicted in Figure 5.

The hashing starts with a 608bits header + 32bits nonce as is the case with the current Ravencoin protocol. We add two X16R single algorithms, but in comparison to the DAG generation depicted on Figure 4, we have the following differences:

1) Each X16R is now a dynamically selected single algorithm instead of full 16 algorithm chain.
2) The algorithms selected at the start of the hashing process and at the end are based on 8bit LSBs of previous block hash, as is currently the case with Ravencoin network. Essentially bits 7:4 will select the first algorithm and bits 3:0 will select the second algorithm.
3) 1024bit final_mix is passing through the FNV1 function to become 512bit input for the final X16R Single function.

This is done on purpose to make sure the amount of compute is minimal and allows both GPU and HBM based FPGAs to be able to fairly equally mine Alterhash, whereas it forces potential ASICs to keep all 16 algos on a die.
On top of that, we are proposing the Siphash cryptographic algorithm instead of the 32bit FNV1 function to make sure each address is generated based on full 1024bit mix instead of only 32bits at a time. Also Siphash produced slightly better distribution of addresses than FNV1. Instead of 2.56x we end up seeing something like 2.51x.

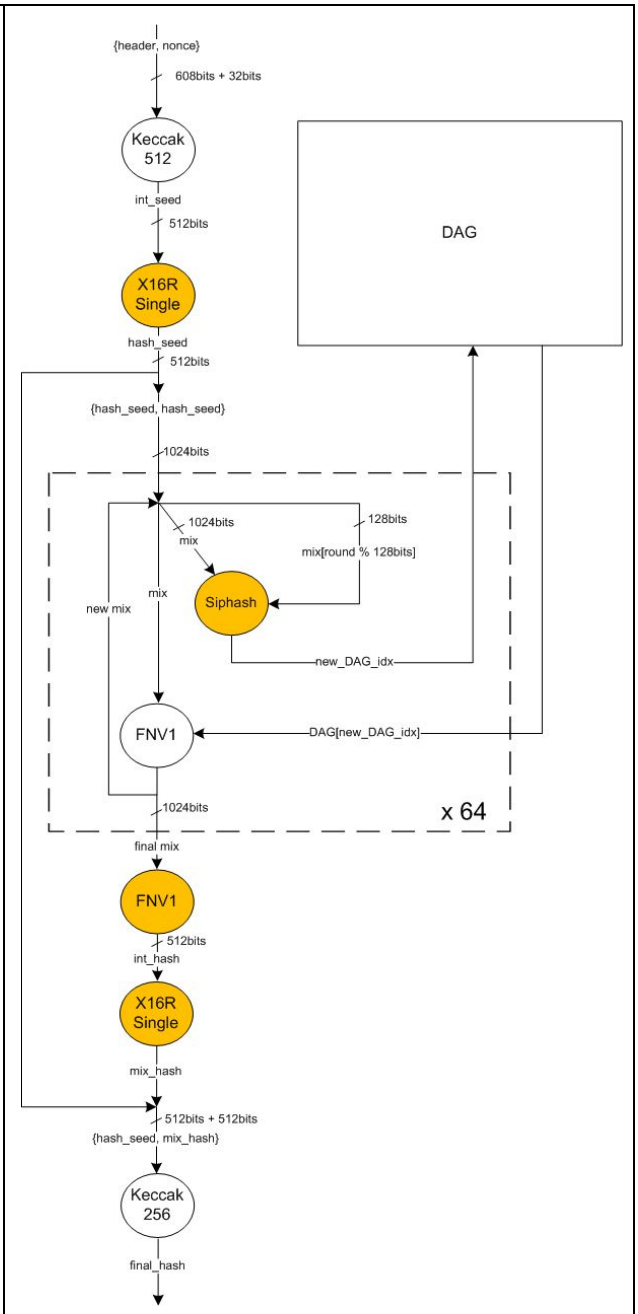

Figure 5 Alterhash hashing logical flow diagram

DAG split into multiple dies

https://linzhi.io/ managed to identify a way to split the Ethash DAG over 64 identical chips, each containing 72MB of on chip SRAM to be able to avoid using external memory. The overall design is estimated to produce in theory 1700Mh/s, but in practice 1400Mh/s hashrate at 1000W and even if instead of 1KW power consumption ends up being 1.3KW, it is still going to be called successful outcome. From public information we can confirm that this is TSMC manufactured ASIC, though we were not able to identify whether it is 28nm or 16nm technology or anything between.
The first 37 wafers have been ordered, which are supposed to produce 300Gh/s hashrate at 1.7Gh/s per device. Hence, they are expecting to produce first 300 / 1.7 = 175 working machines. This implies each wafer should produce 175 * 64 / 37 = roughly 300 chips.

Thanks to the following calculator:
https://anysilicon.com/die-per-wafer-formula-free-calculators/
we can assume that each die is around 190mm^2 size, which is pretty big die.
Here is the published top level architecture of the chip:

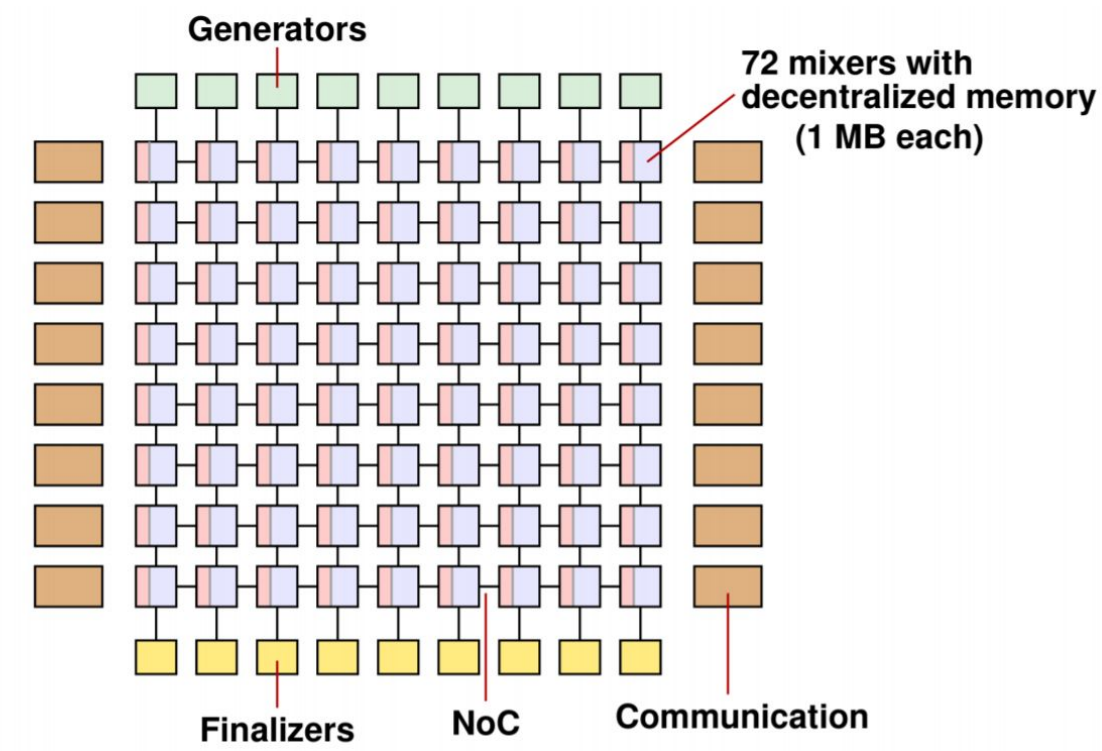

Figure 6 Linzhi E1400 diagram

Green boxes contain initial keccak512 and yellow boxes contain the final keccak256. Each mixer contains 1MB of on chip RAM. Each communication box is a quad box composed of four 32Gbit/s pins. There are 16 such boxes, hence 64 pins. Alterhash would force green & yellow boxes to be much bigger due to the fact that each of them must integrate 16 cores of X16R algorithm. Siphash is not much different than FNV when it comes to ASICs, hence we will ignore its effect on Linzhi E1400 type chip.

Alterhash hardware comparisons

Let's now run price / performance / power metrics for Alterhash on various hardware solutions and see where we end up.
The assumption, which is yet to be confirmed until after GPU testing is done, is that the combination of X16R single algorithms + Siphash don't force GPUs to become extremely compute bound. Based on GPU testing we will decide upon which Siphash version to integrate into final algorithm, either version 2-4, which is more secure but requires more compute resources, or 1-2 version, which is less secure but requires less compute resources. It is super important to keep compute minimal! All numbers below are based on the fact that 1080TI type of GPU ends up producing roughly 50MH/s on Alterhash.
Acceptable metrics would be something like up to 30MH/s, in which case ROI numbers for all hardware types will become 50/30 = 1.67x less. This is still ok. HBM type of FPGA should be able to produce roughly 70MH/s. Essentially the aim is to have HBM FPGA having a lower maximum 2x performance advantage over well known GPU like NVIDIA 1080TI, instead of the current 5x to 6x for X16RV2, which could even increase to 10x with further optimizations. Power efficiency metrics being around 3x is also acceptable. Anything more efficient or performant than that automatically gives ASICs way too much advantage and shortens their ROI metrics, creating a higher potential for investors to back an ASIC project.
A maximum 2x performance advantage and roughly 3x efficiency advantage for FPGAs remains justified due to the higher upfront cost of HBM FPGAs and their bitstream development and significantly lower production numbers. Pretty soon 7nm NVIDIA GPUs will hit the market, the efficiency and performance gap will become even smaller. At the current time, hashrate from Altered Silicon's FPGA bitstream occupies only about 1% of total network hashrate for Ravencoin.

Multi die ASIC miner for Alterhash:

Let's take the Linzhi example and see what ROI would look like if they targeted Alterhash. (Amounts in USD)
Linzhi was targeting roughly 4 months ROI per device for a customer at the time when the machine was able to produce $20 per day. That means they were very likely planning to price it at 20 x 120 = $2400-$2500 per unit.
Let's assume they produced chips on 28nm technology. As each wafer cost at 28nm is roughly $4,500 and the die would cost $4,500 / 300 = $15.
Let's assume with packaging the cost ends up at $18 per chip.
The full device requires 64 such chips, hence 18 * 64 = $1,150.
The final device would probably settle at around $1,300 per unit as a production cost.
As NVIDIA 1080Ti makes around 30mh on X16R, and if the Ravencoin network was forking to an Ethash type of algo, the overall network hashrate should be settling at around (50 / 30) x 10^{TH} = 16.5^{TH}.
To hit 16.5TH, Linzhi would need to manufacture roughly 11,785 devices. The production cost would be $15.3M. On top of that we need to add $2M for the full maskset cost. As they have already done full design cycle we will not add any other cost. Say final investment settles at $17.3M.
The gross profit at current Ravencoin prices, if a company ends up controlling 65% of network would be = 180K x 0.65 = $117K per day.
Power cost = 11,785 x 1KW x 24 x $0.05 = $14K per day.
Net profit would be equal to $103K.
ROI would be equal to $17.3M / $103K = about 168 days.

Now, those numbers would be true, if we did not modify the original ETH hashing algorithm. The 9 green and 9 yellow boxes would need to be much bigger in case of Alterhash. Each of them would have to include, in addition to Keccak512 / Keccak256, all 16 cores of X16R chain. We don't know whether Keccak512 / Keccak256 are unrolled or rolled, so let's assume they are rolled once based on the fact that this chip retrieves data from SRAM every 2 cycles. Taking into account the fact that each core unrolled on average occupies 0.55mm^2 area in 28nm technology, we can take 0.3mm^2 as a number for the rolled core and calculate that each green and yellow box would have to add 0.3 x 16 x 1.3 = 6.25mm^2 area, where 30% is added to count Place & Route penalty. Overall the chip die would be increased by 6.25 x 9 x 2 = 112.5mm^2

As a result the die would be roughly 300mm^2 instead of 190mm^2.
X16R does not really affect GPUs and FPGAs, whereas in this scenario it adds significant die area for such an ASIC. Due to the die size increase, each chip would cost roughly $27 to manufacture.
Full device production cost would be roughly $1,950.
Final production cost to hit 50% of Raven network and take control of 65% of overall hashrate would be: $1,950 x $11,785 + $2M = $25M.
ROI on production cost would be equal to $25M / $103K = **243 days**.

Now lets see what the numbers would look if ASIC company does not have a choice but to simply use external memory.
We will be using Whattomine reported numbers.
Obviously things may be further fine tuned, but let's for the sake of calculations use those numbers:

a) A 1080TI currently makes around 50Mh on ETH at 190W.
b) A 1080TI uses 11GBytes of RAM composed of 11 x 8Gbit GDDR5X chips each reporting 11Gbit/s bandwidth per data pin. There are 32 data pins, hence each chip can achieve 11 x 32 / 8 = 44GByte/s bandwidth.
c) 11 memory chips together can achieve 484Gbyte/s overall bandwidth.

Let's look at two different scenarios. An ASIC project chooses the relatively cheaper option of GDDR5/5X vs more expensive HBM2.

Cheaper Option (less efficiency): ASIC with GDDR5/5X
At high quantity GDDR5X 8Gbit chip should cost roughly $4 per chip.
Now for an ASIC company to match 1080TI performance, it needs to purchase 11 x 4 = $44 worth of chips. Lets add die cost + packaging + PCB and call it $50.
In terms of power consumption, a realistic number would be 80W, which is what 11 GDDR5X memory chips + GDDR5X PHY + memory controller would consume when full memory bandwidth is utilized.
Essentially 80W at $50 makes roughly same 50Mh as a 1080Ti at 190W costing $500 - $800.
As a 1080Ti makes around 30mh on X16R, then if Ravencoin network was forking to Ethash type of algo, the overall network hashrate should be settling at around (50 / 30) x 10^{TH} = 16.5^{TH}
An ASIC company using GDDR5X memory chips to hit 50% of the hashrate and essentially gaining 65% of the network by causing some of the usual GPUs to be forced out of the network, would require:
16.5 x 1,000 x 1,000 / 50 = 330,000 devices similar to 1080TI GPU.
Cost == 330,000 x $50 = $16.5M
Power = 330,000 x 80W = 26.4MW
Number of GDDR5X chips = 330,000 * 11 = 3.63M chips

If we add up the design & engineering cost of the ASIC chip (Non Recurring Engineering, or NRE), which may be $500K to $1M, on 65nm / 40nm technologies, we have $17.5M to ROI and require pretty complicated logistics to set up larger mining facilities at 26.4MW as well as purchasing a huge amount of 3.65M memory chips.
Let's see how long it could take to ROI this investment at current prices.
RVN makes roughly $180K per day today, network-wide.
65% of that is $117K.
At 26,500KW x 24 x $0.05 (price per KW) = roughly $30K per day is going to be used on electricity.
So ROI will become $17.5M / ($117K – $30K) = **200 days**
On top of that instead of something like 2MW, an ASIC company has to sort out serious logistics of 27MW facilities as well as find a way to purchase so many memory chips.

Expensive Option (better efficiency): HBM/2
The 2 x 4GB of HBM2 stacks should cost roughly $70-$100. As this tech is relatively new, we were not able to find HBM PHY older than 28nm tech. Let's assume the corresponding die would be on 28nm technology.
8GB of HBM2 at 600GByte/s bandwidth should produce something like 70MH at probably 30W, which includes HBM2 memories + PHY + controller. It could be actually more but let's take 30W as a number for calculations.
Let's assume die + interposer + packaging + HBM2 memory chips together will cost $100 per such device, consuming 30W and producing 70Mh.
Now again to hit the 50% of hashrate that would be:
16.5 x 1,000 x 1,000 / 70 = 236,000 devices
Cost == 236,000 x 100 = $23.6M
Power = 236,000 x 30 = 7MW
Number of HBM2 4GB chips = 236,000 * 2 = 472,000 chips
Let's add NRE, which should be another $2.5M at 28nm.
Overall investment will be $26M.
Logistics here are less, though still bothersome.
Power cost = 7,000 x 24 x $0.05 = $8.4K per day.
ROI time = $26M / ($117K – $8.4K) = **239 days**.

Conclusion
Obviously one may argue, that an ASIC company may not target 50%, but rather less % of the network. Yes this could be the case, but then it is no longer an existential threat for the network. Plus, NRE becomes an even bigger portion of the investment risk, so the ASIC project must attempt to take as much network as possible in order to get a return on its engineering investment. So there is more uncertainty, which inhibits investment.

In addition to this, for an ASIC company to successfully offer its miner to the retail consumer, its sale price should be set to ROI (for the consumer) in about 11 - 12 months. With such high costs of engineering and production, these two ASIC scenarios don't offer much of a compelling business case for investors to fund the NRE costs required in order to develop the product. The GPUs, FPGAs and potentially even CPUs of the future will all have residual value and additional computing value such that investors will choose them over single-purpose ASIC miners.

Summary of the Alterhash proposal

1) Take the original Ethash codebase and introduce changes depicted on Figure 4 and Figure 5.
2) Start with either 3GB or maybe even 5GB DAG size and keep it growing by 8MB every 5.2 days.
3) Ethereum blocks are found every 15 seconds, whereas, in Ravencoin it is 60 seconds, so the suggestion here would be to decrease epoch length from 30,000 blocks to 7,500 blocks
4) Run enough testing to make sure combination of additional single X16R algorithms with Siphash don't add way too much compute for GPUs, making them heavy compute-limited rather than memory-limited.

Notes

Nothing is designed in a vacuum and in the emerging technology of blockchain, it has been shown many times before that incredible global communities with a wealth of knowledge will come together around the right kind of effort. One that acknowledges the need for commercialization, while still seeking to advance a less tangible but potentially much greater value. There is an opportunity here to collaborate and contribute to a way of securing and decentralizing blockchain processing that could dramatically advance the spread of networked high performance computing across the world - and we invite the engineering and enthusiast community to comment and contribute.

About Altered Silicon

Altered Silicon was founded by Dave Carlson and Arsen Julhakyan in January 2019 to pursue a vision of combining blockchain processing (hashing) with other forms of high performance computing, in order to facilitate the true decentralization of mining across the world.

Altered Silicon is a High Performance Computing software engineering company. We are concentrating on developing specialized hardware and software solutions utilizing FPGA (Field Programmable Gate-Array) and ASIC (Application Specific Integrated Chips) processors both in and out of the cryptocurrency space. We are where Hyperscale computing meets Blockchain.

For inquiries please contact
Molly Hunter, VP Sales & Business Development
molly@altered-silicon.com