Beam Mining Algorithm Design

Wilke Trei

December 2018

1 What is Equihash?

Despite its name Equihash is no hash function in the narrow sense. Instead an Equihash solution is the solution to a mathematical problem. The problem can be formulated like follows.

Definition 1 *Let n and k be the Equihash parameters and let $work$ and $nonce$ be given bitstreams. Then we define $m := \frac{n}{k+1}$ to be the* collision *length for n and k. Further let*

$$B(k) := Blake2B\left(\text{concat}\left(work, nonce, l\right)\right)$$

be the output of the Blake2B hash function for $l = 0 \dots \frac{2^{m+1}}{\lfloor \frac{512}{n} \rfloor}$.

Finally compute $s := \lfloor \frac{512}{n} \rfloor$ disjoint sections of length n bits out of each $B(l)$ and index them first in order of l and then in their local position within $B(l)$.

Then a valid solution of the Equihash problem is a set of 2^k indexes such that

- *all indexes are pairwise distinct,*
- *for any $1 \leq i < k$ the exclusive or (xor) of all elements referenced by an 2^{i+1} elements index block is 0 on the first $i \cdot m$ bits ,*
- *the exclusive or of all elements indexed by the solution equals 0.*
- *the indexes are sorted in a way such that for two index blocks with 2^i indexes each the one with the lowest leading element leads first. E.g. $I_{2 \cdot j} < I_{2 \cdot j+1}$ for all $0 \leq j \leq 2^{k-1}$.*

Note that this is no full formal description but it will be enough to discuss the characteristics of Equihash. First of all we note that an Equihash problem instance may or may not have a solution. In fact there are on average approximately two solutions for each concrete problem instance. The proof for this is out of scope for this document.

2 Difficulty of an Equihash Problem

The algorithm behind the computation of an Equihash solution is the Wagner algorithm. The basic idea is to find pairs of bit streams matching on its lowest m bits. The XOR of each such pair will then give a bitstring that is shorter by m bits and is input to the next round. Iterating this process $k-1$ times will create bitstreams of length $2m$ that each have 2^{k-1} full length ancestors. If two such elements match on the remaining $2m$ bits and its ancestors are pairwise distinct they will give raise to an Equihash solution.

The matching can be achieved by sorting the elements by their lowest bits in each of the overall k rounds. The complexity of sorting is quasi-linear in is the number of elements to be sorted, which equals 2^{m+1} in the beginning. It can be shown that the expected number of pairs generated in each round is again 2^{m+1} except for the last round where we only expect two outputs because we match elements on twice the number of bits.

The following table lists the number of elements to be generated for the most frequently used Equihash instances.

Coin [a]	MinexCoin	ZCash	AION	BitcoinZ	Zero	BEAM
n	96	200	210	144	192	150
k	5	9	9	5	7	5
m	16	20	21	24	24	25
Elements	2^{17}	2^{21}	2^{22}	2^{25}	2^{25}	2^{26}
Memory Use [b]	≈ 7	≈ 200	≈ 600	≈ 1600	≈ 3000	≈ 3800

[a] First appearance of the Equihash parameters as primary PoW

[b] In MBytes per instance for current GPU efficient implementations

We see that the memory use as well as the number of elements to be processed is dominated by the number of collision bits m, which is a better indicator for the algorithms complexity then the bitlength of the elements generated n. In the following section we will discuss the advantages FPGAs and ASICs have over GPUs when computing Equihash solutions and our proposal to reduce their efficiency margin.

3 Analysis of ASIC and FPGA advantages over GPUs for performing Equihash

Current ASIC implementations of the Equihash 200/9 algorithm make use of large amounts of on chip memory offering a low latency and high memory bandwidth. For example the Bitmain Z9 Mini features 144 Mbyte of on Chip memory which is close to the theoretic minimal memory for performing the algorithm on a CPU. Given the performance of the ASIC the used bandwidth exceeds 5 Tbytes per second.

In contrast an efficient GPU implementation will usually use a larger memory footprint because the off chip memory controllers benefit from read and write access that is at least 32bit, better 128 bit aligned. Off chip memory has a bandwidth ranging of about 256 Gbyte/s on medium range GPUs to trice the value on high end GPUs equipped with HBM2 memory.

Beside the advantages of high amount of on chip memory and better packed access patters an ASIC as well as an FPGA can trade time spend for parallelizeable compute operations by chip space and power consumption by assigning more circuits to the task. For Equihash the fraction offering most potential for this trade is the Blake2B algorithm.

A massively increased performance of the Blake2B algorithm may be beneficial to trade compute power against bandwidth like follows. In the first rounds of the Equihash matching phases less bits then required for performing the full algorithm can be stored and loaded. As soon as the abandoned bits get required for continuing with the algorithm the chip can recover them by computing the hashes again from the indexes carried to this round.

This approach is in particular efficient in the early rounds when the bandwidth required to transfer the indexes is low compared with the transfer of the original workload bits. Also in this early rounds less different blake2b passes have to be used to recover the concrete bits. For GPUs this approach is not an option, because adding blake2b computations increases duration instead of space of the algorithm. Also this operations consume too much time to be hidden when performing memory operations and they would increase power consumption as well.

4 Applicability towards Equihash 150/5

In the previous section we defined three potential benefits of ASICs compared to GPUs that are large on chip memory areas, a better packing and coalescing of off-chip memory operations and time-space algorithm trade-off.

Regarding the first issue we have strong confidence that a specialized chip for Equihash 150/5 will not take this exact approach due to its increased memory footprint. When completely computed the output of the first round is $(26 + 150) \cdot 2^{26}$ bits and thus approx 1.4 Gbytes. We claim that with growing index tables of matched elements this is a sharp lower bound. This even applies when using the trick described in the time memory trade-off part of the previous section, because even the indexes written in the first 4 rounds compressed to 36 bits per matched element plus the required remaining bits for performing the final matching round requires at least this space.

In practice we even can estimate that the real memory consumption of an efficient implementation is at least twice the size, similar to the ZCash parameters.

Therefore a single chip ASIC holding enough memory to perform all operations at Tbytes per second bandwidth range would be of 10 to 20x the size of the chip to perform ZCashs Equihash 200/9 algorithm on the same manufacturing process. This would increase the cost for producing the chip dramatically because on the same production processes a higher yield can be expected and also larger chips are more costly.

Of cause this approach may and will get feasible in the future with advanced production methods and new technologies regarding fast on chip memory or staking chips with height bandwidth. Never the less the mentioned amount of memory is high enough that the time to market of such new inventions will be longer then our planned PoW review and adjustment period.

The second benefit is of architectural nature with ASICs as well as FPGAs being able to perform memory operations that use bitlength that are not a multiple of 32 more efficiently. Also adding more circuits in the implementation to ensure a better memory coalescing when using external memory chips is exclusive to these chips, while a GPU can improve its memory access patterns only by using software solutions.

We therefore believe this benefit can not be mitigated directly by an algorithm change unless changing the parameters in a way that most memory operations are forced to be in ideal range for GPUs. This is out of scope for

Equihash with nowadays capacities.

Anyways we claim that the total effect of this benefit is limited by the capacities of the external memory controller. For the already well tuned Equihash 144/5 parameters it is known that a single run requires approximately 5 Gbytes of memory bandwidth in total. Each run will produce 2 solutions on average. Modern implementations can run up to 60 solutions per second on an unmodified Nvidia GTX 1080. Therefore on this cards about 150 Gbytes/s of the total available 352 Gbyte/s are effectively used. So a specialized chip equipped with the same memory but more efficient memory access patterns may have a gain limited to a factor of about 2.5 if the algorithm behind is forced to write the same data.

This enforcement aligns with the potential space to memory and bandwidth trade-off discussed before. The BEAM project proposes an algorithm change that makes the trade-off more costly in terms of space requirement and power consumption of the resulting chip.

Note that specialized devices that are programmable as GPUs but are reduced to and specialized for the required operations to mine crypt-currencies and offer additional optimizations for memory access still may be able to perform the algorithm 2-3x faster then a GPU equipped with the same memory and that at a potentially lower energy consumption. On the long term preventing such devices from mining the algorithm is not feasible. At the time of writing no such designs are publicly available nor is there a concrete announcement of such, so we assume that the desired head start for GPU mining BEAM is given.

5 BEAM modification of Equihash 150/5

By the nature of the Equihash 150/5 algorithm it has a blocking rate of 3 in its Blake2b phase. This is that if the original algorithm is modified in a way that in later rounds the Blake2b algorithm has to be performed again, for each hash string we require 3x the computation amount of the initial calculation, because in the first round 3 hash strings of 150 bit length are generated in one computation.

This also applies to the verification of an Equihash 150/5 solution, but is no issue for the verification because the total number of Blake2b runs to verify one solution is 32 while the avergage number of runs for generating one solution exceeds 11184810 that is approx $\frac{2^{26}}{3 \cdot 2}$.

We propose to increase the blocking factor to 48 by the following scheme.

Algorithm 2 *Let $B(l)$ be the 512 bit string corresponding to the Blake2b hash for input index l and let $B(l)_j$ be the j-th 32 bit component interpreted as 32 bit integer. Then let $B'(l)$ be the 512 bit string computed like follows:*

```
$c \leftarrow 16 \cdot \lfloor \frac{l}{16} \rfloor$
$B'(l) \leftarrow 0$
while $c \leq l$ do
    $B'(l)_j \leftarrow B'(l)_j + B(c)_j$ for all $0 \leq j < 16$
    $c \leftarrow c + 1$
end while
```

Thus in order to compute the modified hash string B' it may be required to know all 15 other hashed in the same group of 16 hash elements. On a GPU this can be cheaply achieved by using the local memory on the device in the initial computation phase. Also on modern GPUs neighbored threads with index only varying on the lowest four bits run in lock steps, so the sharing over the local memory does not introduce new synchronization barriers.

Therefore for performing the algorithm the intended way the slow down by the extra computation of the sums will be negligible. On the other hand when it is intended to run Blake2b later again to recover bits from known indexes this is up to 16 times more costly then before and overall up to 48 times so costly then in the initial round with an average extra cost factor of 24.

By this approach we aim towards forcing the algorithm to follow the same algorithm implementation as done on GPUs or else to use more chip-space and drastically increased power consumption, because performing the Blake2b algorithm is the most power consuming component of Equihash.

Note that also the verification of solutions get more costly by an average factor of eight. But since only few solutions needs to be verified and due to the still high asymmetry of generation effort compared to verification effort the drawback is acceptable. Overall with this modification the verification of an Equihash 150/5 solution can be done at lower average cost as the verification of an Equihash 200/9 solution while the worst case costs are equal.