Memory-Bound Proof-of-Work Acceleration for Blockchain Applications

Kun Wu[†‡§], Guohao Dai[†‡], Xing Hu[§], Shuangchen Li[§], Xinfeng Xie[§], Yu Wang[†‡], Yuan Xie[§]

[†] Department of Electronic Engineering, Tsinghua University
[‡] Beijing National Research Center for Information Science and Technology (BNRist)

[§] Department of Electrical and Computer Engineering, University of California, Santa Barbara
yu-wang@tsinghua.edu.cn,yuanxie@ece.ucsb.edu

ABSTRACT

Blockchain applications have shown huge potential in various domains. Proof of Work (PoW) is the key procedure in blockchain applications, which exhibits the memory-bound characteristic and hinders the performance improvement of blockchain accelerators. In order to mitigate the "memory wall" and improve the performance of memory-hard PoW accelerators, using Ethash as an example, we optimize the memory architecture from two perspectives: 1) Hiding memory latency. We propose specialized context switch design to overcome the uncertain cycles of repetitive memory requests. 2) Increasing memory bandwidth utilization. We introduce on-chip memory that stores a portion of the Ethash directed acyclic graph (DAG) for larger effective memory bandwidth, and further propose adopting embedded NOR flash to fulfill the role. Then, we conduct extensive experiments to explore the design space of our optimized memory architecture for Ethash, including number of hash cores, on-chip/off-chip memory technologies and specifications. Based on the design space exploration, we finally provide the guidance for designing the memory-bound PoW accelerator. The experiment results show that our optimized designs achieve 8.7% – 55% higher hash rate and 17% – 120% higher hash rate per Joule compared with the baseline design in different configurations.

1 INTRODUCTION

Blockchain has drawn great attentions from both academic and industry, due to its unique features of decentralization and anonymity and the potential of wide adoptions for various fields, such as financial services, voting, and Internet-of-Things (IoT) [25]. Walmart has been working with IBM on applying blockchain technology to the food supply chain, which can significantly reduce the time of tracking from several days to 2.2 seconds, compared with conventional mixed digital and paper-based methods.

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317862

Cryptocurrencies [10, 22] are one of the most widespread used applications of the blockchain. Mainstream cryptocurrencies (e.g., Bitcoin, Ethereum, etc.) have shown the potential to become the future currency due to their unique decentralized secure transaction processings. The core component within cryptocurrencies is the Proof of Work (PoW) mechanism. The PoW is the original consensus algorithm in a Blockchain network, which is used to confirm transactions and produce new blocks. Compared with traditional consensus algorithms, the PoW is more robust against premeditated attack. Conventional consensus algorithms, represented by Paxos [14], can only tolerate at most 1/3 faulty nodes [5] in a multi-node system, while the PoW tolerates at most 1/2 faulty nodes. This endows the PoW a superior characteristic when designing distributed systems. Consequently, PoW accelerator (i.e., coin miner) has emerged to speed up the coin founding or transaction with lower cost (power) driven by significant financial rewards.

Ethereum is one of the most popular mainstream cryptocurrencies [10]. It is a more advanced design that offers fast transaction and fixes many significant pitfalls of other cryptocurrencies like Bitcoin. Ethash is the PoW mechanism used in Ethereum, which is essentially a memory-hard hash function. The key idea of the Ethereum is adopting a memory-hard PoW (compute-hard in Bitcoin) to effectively deter the ASIC miners. In Bitcoin, compute-hard PoW significantly benefits ASIC miners compared with general purpose GPU [28], since compute resources are easy to scale up in the ASIC design. However, the performance of the memory-hard PoW depends on the off-chip DRAM bandwidth rather than compute resources on the chip, which is then referred to as hitting the "memory wall". As a result, an ASIC chip cannot outperform GPU when providing the same off-chip DRAM setup, making the ASIC design less attractive.

Despite the "memory wall" challenge, there are still some accelerators that have emerged in addition to the well-adopted GPU approaches for the Ethash [9]. Taking Bitmain E3 as an example, it leverages the 16-I/O DRAM component [7] and single rank per channel for achieving high off-chip bandwidth with fewest possible number of DRAM components (14.9GB/s × 8 Channels = 119.2GB/s for 32 × 1Gb DRAM components [30]), and hence provides much higher bandwidth (performance) per cost ratio. The success of this miner design inspires this work. It shows the potential benefits of the memory system optimization on such accelerators, while leaving the on-chip memory and a large design space yet to explore.

In this work, aiming to mitigate the "memory wall" challenge for Ethash, we explore the optimizations and design space of both the on-chip and off-chip memory architecture design. To this end, we first present a performance analysis for the Ethash accelerator. Based on the analysis, we then propose two techniques to improve effective bandwidth and hide latency, specifically the on-chip memory and the context switch. We further investigate the benefits of adopting the embedded NOR flash memory as on-chip memory. We finally conduct a comprehensive design space exploration for the various on-chip memory designs and off-chip DRAM choices. Our specific contributions are listed as follows.

- We propose a hardware performance evaluation for the memory-hard Ethash from measured throughput on available hardware platforms, followed by insightful performance analysis.
- We introduce two architectural optimization techniques to mitigate the "memory wall" problem, i.e., context switch for hiding memory latency and on-chip memory for additional bandwidth. We further evaluate adopting NOR flash as on-chip memory.
- We extensively evaluate possible design options. We consider a wide range of possible on-chip memory and off-chip protocol options. In showcase configurations, our methodology offers design that reaches 8.7% – 55% better hash rate and 17% – 120% better hash rate per Joule compared with the baseline design.

2 BACKGROUND AND MOTIVATION

This section describes the background and characteristics of Ethash.

2.1 Ethash as a Memory Hard Proof of Work

The key characteristic of Ethereum is its adoption of memory-hard Proof of Work. The procedure of Ethereum's Proof of Work, named as Ethash, adds random memory access requests to each primal operation. The addresses of these memory requests are computed depending on the immediate result just before the memory access instruction. The randomness of addresses makes it hard to exploit data parallelism during execution, which deteriorates the memory wall issue and hinders the computation speed. Therefore, Ethereum ASIC miners are effectively deterred. The data in memory is named as DAG for its generation method.

Memory-hard PoW is indispensable to the PoW family. In addition to Ethash, memory-hard PoW also includes Cuckoo Cycle, Itsuku, etc. [4, 29] Noteworthy, some cryptocurrencies descendant to Bitcoin also adopt memory-hard PoW mechanism. For example. Litecoin adopts Scrypt [24] as its PoW. But the basic idea of these memory-hard PoWs is the same as Ethash. In summary, memory-hard PoW is broadly used because of its characteristics of decentralization. In this work, we use the Ethash as a case study to analyze the micro-architecture characteristics and the potential acceleration towards memory-hard PoW mechanisms.

2.2 Ethash Algorithm Description

The Ethash procedure is to find an acceptable *nonce* for block creation. To this end, it concurrently runs many `Ethash_search` threads, each trialing with a unique *nonce*. Once an acceptable *nonce* is found, i.e. the corresponding `Ethash_search` yields a hash value smaller than the threshold set forth in the network target, Ethash stops.

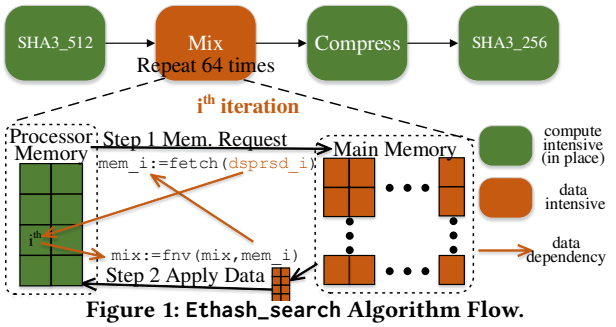

Figure 1: `Ethash_search` Algorithm Flow.

Each `Ethash_search` is made up of four stages: `SHA3`, `Mix`, `Compress`, and `SHA3` again. (Figure 1) Because of dependencies, parallelism can rarely be exploited among the stages and within each stage in a single `Ethash_search`.

1). **SHA3** as the first and last stage uses the *standard Secure Hash Algorithm*. They perform a series of dependent integer operations that cannot be parallelized.

2). **Mix** repeats 64 times non-sequential and wide-spread memory addressing, fetching 128 bytes in each iteration. The memory addresses are calculated by `fnv` (Fowler-Noll-Vo hash function, a bitwise primitive). Because of its dispersion properties, `fnv` produces pseudo-randomized and non-predictable addresses, ruling out caching and prefetching for `Mix`. `Ethash_search` is highlighted by the intensive random memory access during `Mix`. The large amount of memory accesses in `Mix` constitute the memory working volume of Ethash. In addition, the loop-carried dependency in `Mix` precludes loop-level parallelism. Therefore, the `Mix` stage takes up the most of the execution time.

3). **Compress** contracts the 1024-bit input into 256 bits through three iterations of `fnv` bitwise primitives. Bit parallelism can be exploited in this stage.

3 PERFORMANCE ANALYSIS OF ETHASH

This section conducts performance analysis of Ethash by applying the roofline model and the analytical model.

Throughput is the most important performance metric in Ethash, similar to the hash-rate in Bitcoin. Therefore we use it as our evaluation metric of performance and it is measured as the number of `Ethash_search` threads completed in an unit time.

Roofline Model. First, we evaluate the Ethash performance based on the roofline model. Roofline model provides the upper bound to the performance of the application. It is made up by a horizontal and a diagonal roof, meaning the application is compute bound and memory bandwidth bound, respectively. At the ridge point, i.e., the point where horizontal and diagonal roof meet, the system exactly saturates both its compute units and memory bandwidth, so that the compuation and memory demands are balanced.

Figure 2 shows the Ethash roofline model on several general purpose platforms. The original operational points on each platform are marked with ▲. Reducing off-chip memory accesses increases the Ethash throughput: It increases along the slope on the roofline. At the ridge point, off-chip memory access is reduced to zero. This indicates that the Ethash throughput is bottlenecked by the memory access, consolidating the memory bandwidth as the severe bottleneck to Ethash on general purpose platforms.

Therefore, the acceleration by utilizing memory architecture design would be beneficial. To propose the architecture techniques for accelerator design, we quantitatively analyze the performance in the further step.

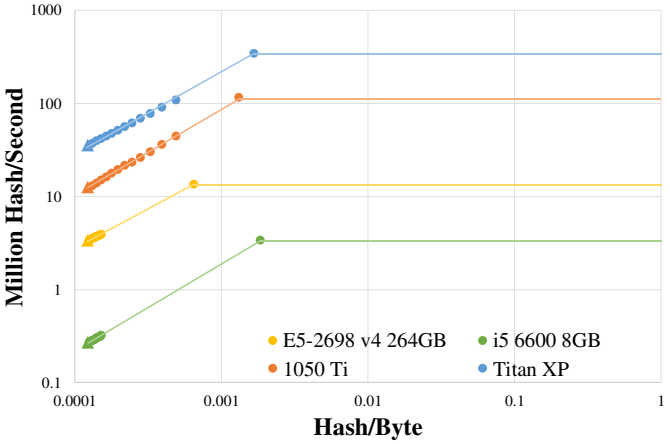

Figure 2: Ethash Roofline. Samples are marked on roofline.

Performance Analysis. We analyze the impact of each design factors on the Ethash performance in this section.

Since the hash core is an in-order core, the average execution time of single `Ethash_search` thread can be described as,

$$T = Time(Com) + Time(Mem) \tag{1}$$

where $Time(Com)$ is the execution time of computation and $Time(Mem)$ is the memory access time.

For the computation time, we have:

$$Time(Com) = O\left(N_{OP,hash}/N_{PE}\right) \tag{2}$$

where the $N_{OP,hash}$ is the total number of operations involved in performing a single `Ethash_search` execution and N_{PE} is the number of the Processing Element.

For the memory access time, we have:

$$Time(Mem) \in \left[\frac{N_{OP,mem}}{BW}, N_{OP,mem} \cdot Lat_{mem}\right] \tag{3}$$

where the $N_{OP,mem}$ is the total number of memory operation volume for one hash operation. The Lat_{mem} is the latency for one memory access. When the architecture design perfectly hides the memory latency, memory access is bottlenecked by bandwidth. When the architecture poorly exploits the memory level parallelism, the memory access is bottlenecked by memory access latency. Therefore, to reduce the memory access overhead, we should 1) **design an architecture for larger memory level parallelism to hide the long memory access latency**; and 2) **provide a larger effective memory bandwidth.**

The Ethash throughput can be described as follows:

$$Ethash_{throughput} = \frac{N_{PE} \cdot N_{thread}}{T} \tag{4}$$

where N_{thread} is the number of hardware threads for supporting context switch. As noted in Section 2.2, it is hard to exploit parallelism inside the `Ethash_search` thread. But it is feasible to enable concurrency among the `Ethash_search` threads. **Increasing hardware parallelism for more threads improves the Ethash throughput.**

In summary, we observe that several architecture optimizations can be made to increase the Ethash throughput during performance analysis: 1) hiding the latency with more parallelism; and 2) increasing the memory bandwidth. Therefore, we make architecture optimization as described in the following section.

4 ARCHITECTURE OPTIMIZATION

4.1 Hiding Latency with Context Switch

As analyzed in Section 3, the ideal throughput of the Ethash accelerator is bottlenecked by its memory bandwidth. However, when parallelism is insufficiently exploited, the throughput is further bottlenecked by the latency.

Context switch is one of the most important techniques to exploit the inter-thread parallelism. It suspends the active thread when the latter is awaiting the fulfillment of memory request so that some stand-by thread whose data are ready can proceed.

In our design, context switch is enabled to hide the severe memory latency in the `Mix` module. As shown in Figure 1, each iteration in the `Mix` requests memory for once and computes, depending upon the requested data, the new in-place data and the address to request in the next iteration. The data and address calculation can be performed within one cycle, while waiting for memory takes for way longer time, severely slowing down the hash core. Therefore, we introduce context switch for `Mix`. Other stage modules process one thread until finish before processing other threads, if any.

The `Mix` module has the capability to hold two standing-by threads. For each standing-by thread, it uses 1) a 128-byte SRAM to store its in-place data, 2) a 128-byte SRAM to store the incoming memory data, 3) a 32-bit register to store the memory address of which it has issued a memory request, or is going to. To fully exploit the inter-thread parallelism, the `Mix` module will signal the first `SHA3` stage to initiate a new thread if the number of stand-by threads `Mix` holds are lower than its capability. In the in-order hash core, the three stages other than `Mix` have exact cycles because they do not involve memory operations. Therefore it is predictable whether initiating a new thread will cause hazard. As a result, the `Mix`, equipped with some control logic, is able to signal thread initialization at the correct cycle. Threads are created and proceeded if the corresponding stage module is idle and if overall storage will not at any time exceed the capacity of the hash core.

4.2 Larger Bandwidth with On-Chip Memory

As analyzed in Section 3, the effective memory bandwidth is important to the Ethash throughput. Therefore, we adopt on-chip memory for larger effective memory bandwidth. Modern chips normally involve on-chip memory to boost performance. For example. Embedded DRAM (eDRAM) has been adopted in IBM Power8, Intel Xeon Phi and Intel Graphic Cards. The on-chip memory capacity can be as large as 64 – 256 MB [3, 11, 17].

Note that the overall effective bandwidth consists both the off-chip effective bandwidth and on-chip counterpart,

$$BW_{Eff,Overall} = UTIL_{Off} \times BW_{Off} + UTIL_{On} \times BW_{On} \tag{5}$$

where the utilization $UTIL$ is the ratio of cycles when memory units are servicing memory requests. Under-utilization of memory bandwidth, indicated by the low $UTIL$, ultimately hinders the Ethash throughput (Equation 3).

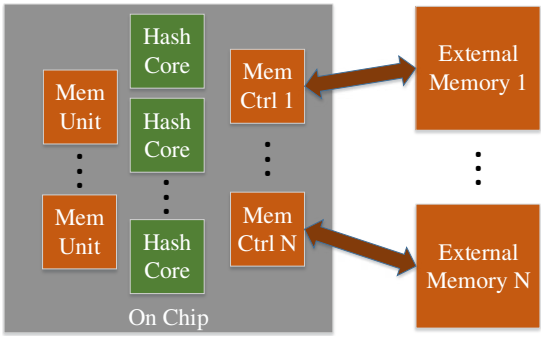

Figure 3: General Design Abstraction in our Work.

In summary, we introduce our architecture as shown in Figure 3. The overall effective bandwidth is contributed by on-chip memory (Mem Unit) and external one. The external memory is accessed via corresponding Memory Controllers. For on-chip memory, SRAM, eDRAM and NOR flash (discussed in Section 4.3) are considered. For external memory, DDR4, GDDR5, HBM2 are considered.

4.3 Introducing the Embedded NOR Flash

Embedded NOR flash is a mature memory technology that is byte-addressable and very dense ($8F^2$ [23] vs. 140 F^2 of SRAM [8]). The downside of NOR flash is that (1) it is difficult to integrate with CMOS beyond 45nm; (2) the write operation takes significant long latency and energy; (3) the write endurance is as low as 10^5 [19]. Due to these drawbacks, embedded NOR flash is typically used as Read Only Memory (ROM), but rarely as data buffer.

However, **we find that the embedded NOR flash technology would be a perfect match for the Ethash accelerator.** First, the fast byte-addressable access and high density meet Ethash accelerator's requirement for the on-chip memory (see Section 4.2). Second, previous research [12] has shown that using old technology like 65nm provides higher cost efficiency and lower Total Cost of Ownership (TCO) in the data center. With the old technology node being used, integrating NOR flash is not a problem anymore. Third, the high cost of NOR flash's write operation does not bother the Ethash application. The on-chip memory only stores the DAG data, which is refreshed about every 121 hours [9]. Fourth, NOR flash's write endurance is not a problem due to the extremely slow DAG data refreshing rate. Given 10^5 endurance [19], the Ethash accelerator with such NOR flash can continuously work for 138 years.

Though other emerging memory technologies, such as embedded MRAM and ReRAM, can potentially provide better performance than NOR flash, they are not widely available for either large capacity or low cost. Therefore, they are not considered in this paper.

5 DESIGN SPACE EXPLORATION

In this section, we conduct experiments and design space explorations (DSE) for our memory optimization. We break down the problem of finding the optimal design into four questions. We present design guidelines for these questions, followed by full DSE results.

5.1 Experiment Setup

Table 1 summarizes the design knobs considered in our DSE. To evaluate the circuit parameters (including delay, area, and energy), for the hash core part, we use frond-end synthesis of the Verilog based on an open-source SHA3 implementation [13] with FreePDK [16, 26] and 740MHz; for the memory controller IP core, we use McPAT [15]; for the on-chip memory part, we use NVSim [6] and CACTI [21] ; for the DRAM part, we use standard parameters. To evaluate the Ethash runtime performance, we developed an event-driven performance simulator, which is validated by the profiling results on GPU (shown in Figure 2).

Table 1: Design knob summary and their setups

Knobs	Options	Delay/Area/Power setup
Hash core counts		synthesis [27] and McPat [15]
On-chip memory capacity, width		NVSim [6] and CACTI [21]
Mem. technology	SRAM, eDRAM, NOR	with device configs. [18]
Technology node	14nm, 45nm	FreePDK [16, 26]
Off-Chip DRAM	DDR4, GDDR5, HBM2	Spec. [1, 2, 20], close page

5.2 Necessity of the Context Switch

Figure 4 shows the proposed context switch technique significantly improves the performance. The figure shows the performance results with various number of hash core setting. First, it shows that more hash cores does not always bring higher performance, due to the memory-bound nature mentioned in Section 3. Second, without context switch (the blue roof), 31 more hash cores are needed to achieve the peak performance, since it is bound by memory latency even before it is bound by memory bandwidth. On the contrary, context switch (the orange roof) achieves the peak performance with the minimal possible number of hash cores.

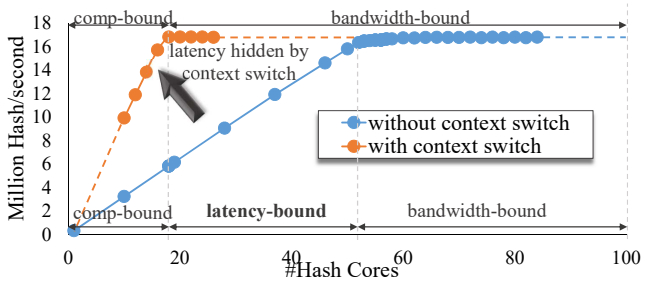

Figure 4: The impact of context switch and hash core number on Ethash performance (GDDR5, w/o on-chip memory).

The hardware overhead of the context switch includes an extra SHA3 (keccak) module, SRAM buffers, and controllers. The area and power overhead of context switch for a single hash core are listed in Table 2. Though the overhead is noticeable for a single core, we show 57.4% and 61.7% saving for chip-level area and power, respectively. This is because with context switch less hash cores are required to reach the peak performance.

5.3 Overall Impact of the On-Chip Memory

Figure 5 shows how the on-chip memory improves the performance and energy efficiency of the Ethash accelerator. The line with the right y-axis shows the hash rate without the on-chip memory for various off-chip DRAM setting. The bars then show that the on-chip memory (with the optimal setting) can offer up to *8.7% – 55%*

Table 2: Context switch (CS) overhead for single core and total savings for a chip.

		# Core	Area		Power	
Single Core	*w/o. CS*	1	.172mm^2	36.5%	160mW	51.6%
	w/ CS		.261mm^2		220mW	
Whole Chip	*w/o. CS*	57	9.81mm^2	-57.4%	9.16W	-61.7%
w/ peak perf.	*w/ CS*	16	4.17mm^2		3.51W	

higher performance and 17% – 120% improvement for energy efficiency, respectively. In addition, we further conclude two design insights: (1) On-chip memory is *more effective for accelerators with small DRAM bandwidth*; (2) On-chip memory is more attractive for improving energy efficiency than for improving performance.

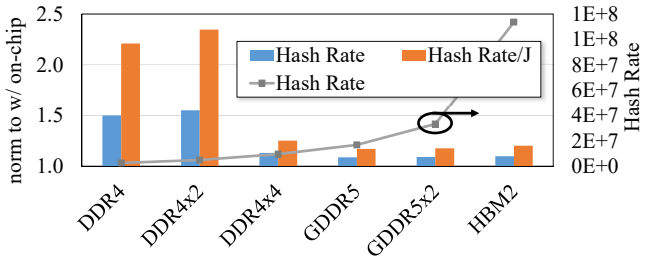

Figure 5: On-chip memory's impact on performance and energy efficiency varies with different DRAM choices. (w/ 256MB on-chip memory .

5.4 Impact of the On-Chip Memory Capacity

We take a closer look at the impact of the on-chip memory from the point of view of the capacity. Figure 6(a) shows the performance (y-axis) and total area (x-axis, including DRAM) with various on-chip memory capacity settings. The two orange dots show the results without any on-chip memory, with one of them as a single node and the other as a dual-node. This represents the scale-out solution. All the other data points show the results of increasing on-chip memory to the single node, representing the scale-up solution. First, we can conclude that *larger on-chip memory provides better performance for the scale-up solution*. Second, *the scale-up on-chip memory solution fails to provide better area efficiency* (perf/mm^2), compared with the scale-out solution. The larger the on-chip memory is, the worse the area efficiency becomes. However, it provides extra design points for fine-tuning in addition to the coarse-grained scale-up solution.

Moreover, Figure 6(b) shows the impact of the on-chip memory capacity on energy, with the same settings as Figure 6(a). We conclude that scaling up with on-chip memory can significantly save energy and hence improve energy efficiency, compared with that of the multi-node scale-out solution. Furthermore, *larger on-chip memory offers better energy efficiency*.

5.5 Impact of the On-Chip Memory Width

Figure 7 further answers how wide the on-chip memory's read port should be. In general, a wider on-chip memory provides larger bandwidth while consuming more area and energy. From the performance and energy efficiency result for various memory width setting, we conclude that *larger width on-chip memory is not necessarily better*, and there is a sweet spot to find (64 Byte in the case of Figure 7). It again motivates our optimization for the DSE.

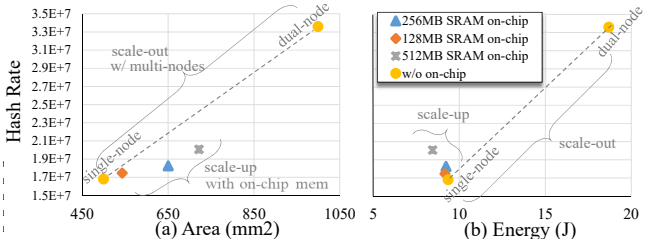

Figure 6: The impact of on-chip memory capacity (w/ GDDR5 and 36 hash cores).

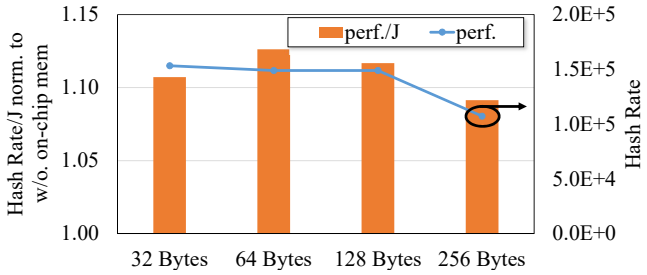

Figure 7: The impact of on-chip memory width (w/ 256MB on-chip memory, GDDR5, and 36 hash cores).

5.6 Impact of the On-Chip Memory Technology

Figure 8 answers the question of what on-chip memory technology is preferred. In this figure, we compare the usage of SRAM and eDRAM with different off-chip DRAM settings regarding the energy efficiency (normalized to the no on-chip memory baseline). Note that each of them has the same number of hash cores, memory capacity and width. We get non-trivial results, i.e., there is not an overall winner. *The choice between SRAM and eDRAM depends on the off-chip DRAM choice.* For example, when using one GDDR5, eDRAM is 16.1% better than SRAM, whereas when using HBM2 they are almost the same. In addition, Figure 9 brings the embedded NOR

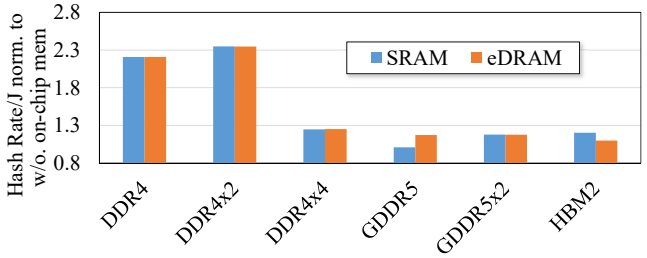

Figure 8: SRAM vs. eDRAM on different DRAM settings (w/ 256MB on-chip memory and 36 hash cores).

flash into the picture when using older technology node (45nm) for cost-efficient accelerator design [12]. We show a case study under the 2x GDDR5 setup, compared the area efficiency and energy efficiency among SRAM, eDRAM, and NOR flash. We conclude that *NOR flash, usually overlooked, turns out to be the best on-chip memory choice* under some DRAM setting, when using older technology node for cost-efficient Ethash accelerator design.

5.7 Putting Them Together: An Extensive DSE

Putting all the design knobs together (on-chip memory capacity, width, technology, off-chip DRAM choice, number of hash cores),

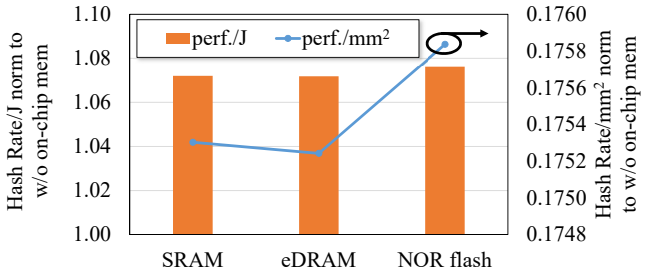

Figure 9: SRAM vs. eDRAM vs. NOR Flash at 45 nm (w/ 256MB on-chip memory, HBM2, and 250 hash cores).

we show the large design space in Figure 10, which is *not only large but also widely distributed*. This demonstrates the importance of finding an optimal result. Our DSE framework not only captures the difference of various design knob, but also provides a heuristic method that searches the optimal results in a lexicographic order. Given a DRAM setting, our DSE finds optimal results with 8.7% – 55% higher performance and 17% – 120% higher energy efficiency (optimal results also shown in Figure 5).

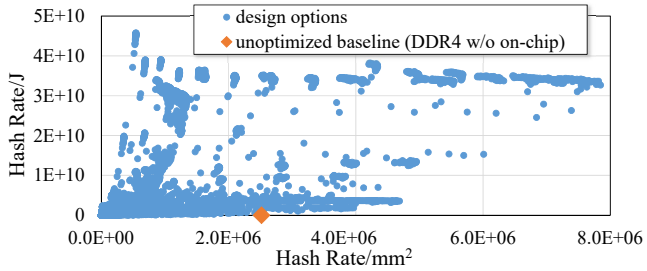

Figure 10: Finding an optimal result in the huge design space.

In addition, we point out that our DSE framework can be adopted for further cost analysis given vendor-specific cost information. The total cost of running Ethash contains (1) the accelerator chip, related to the area information provided by DSE; (2) the DRAM, related to the design choice in DSE; (3) the on-chip memory IP (eDRAM/NOR), related to another design choice in DSE; (4) the electricity, related to the energy consumption provide by DSE. Putting them all together, users can optimize their design for best performance per dollar.

6 CONCLUSION

In this work, we explore the design space of Ethash accelerators. We first conduct the performance analysis of Ethash accelerator and introduce three memory architecture optimization techniques, i.e. context switch, introduction of on-chip memory and embedded NOR flash. Then we effectively explore the huge design space by breaking it down into several questions. We find designs delivering 8.7% – 55% higher hash rate and 17% – 120% higher hash rate per energy compared with the baseline design in different configurations. Our optimization and design space exploration techniques can be adopted for other memory-hard Proof of Work mechanism.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their helpful comments.

This work was supported by National Key Research and Development Program of China (No. 2017YFA0207600), National Natural Science Foundation of China (No. 61832007, 61622403, 61621091), Beijing National Research Center for Information Science and Technology. This work was supported in part by CRISP, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and NSF 1816833, 1500848, 1719160, and 1730309.

REFERENCES

[1] AMD 2015. High-Bandwidth Memory (HBM) reinventing memory technology. (2015). Retrieved November 20, 2018 from www.amd.com
[2] Anton Shilov. 2016. GDDR5X Standard Finalized by JEDEC : New Graphics Memory up to 14 Gbps. (2016). Retrieved November 20, 2018 from www.anandtech.com
[3] John Cazes et al. 2016. Intel Knights Landing Hardware. (2016). Retrieved November 20, 2018 from portal.xsede.org
[4] Fabien Coelho et al. 2017. *Itsuku: a Memory-Hardened Proof-of-Work Scheme.* Technical Report.
[5] Danny Dolev et al. 1986. Reaching approximate agreement in the presence of faults. *J. ACM* 33, 3 (1986), 499–516.
[6] Xiangyu Dong et al. 2014. NVSim: A circuit-level performance, energy, and area model for emerging non-volatile memory. *Emerging Memory Technologies: Design, Architecture, and Applications* 9781441995, 7 (2014), 15–50.
[7] Elite Semiconductor Memory Technology 2017. M15T1G1664A Datasheet. (2017). Retrieved November 20, 2018 from www.datasheetspdf.com
[8] Siddharth Gaba et al. 2014. Memristive devices for stochastic computing. *ISCAS* 8, 1 (2014), 2592–2595.
[9] Github 2017. Mining-ethereum/go-ethereum Wiki. (2017). Retrieved November 20, 2018 from www.github.com
[10] Github 2018. Ethereum: a Secure Decentralised Generalised Transaction Ledger. (2018). Retrieved November 20, 2018 from ethereum.github.io
[11] Intel 2017. Intel HD Graphics P530 & Intel Iris Pro Graphics P580 Performance Guide. (2017). Retrieved November 20, 2018 from www.intel.com
[12] Moein Khazraee et al. 2017. Moonwalk: NRE Optimization in ASIC Clouds or, accelerators will use old silicon. *ASPLOS* (2017), 511–526.
[13] Kazuyuki Kobayashi et al. 2010. *Evaluation of Hardware Performance for the SHA-3 Candidates Using SASEBO-GII.* Technical Report.
[14] Leslie Lamport. 1998. The Part-Time Parliament. *TOCS* 2, August (1998).
[15] Sheng Li et al. 2013. The McPAT Framework for Multicore and Manycore Architectures. *ACM Transactions on Architecture and Code Optimization* 10, 1 (2013).
[16] Mayler Martins et al. 2015. Open Cell Library in 15nm FreePDK Technology. *ISPD* (2015), 171–178.
[17] Alex Mericas. 2014. Performance Characteristics of the POWER8 Processor. (2014). Retrieved November 20, 2018 from www.hotchips.org
[18] Farnood Merrikh-Bayat et al. 2017. High-Performance Mixed-Signal Neurocomputing With Nanoscale Floating-Gate Memory Cell Arrays. *IEEE Transactions on Neural Networks and Learning Systems* (2017).
[19] Micron 2004. NOR Flash Cycling Endurance and Data Retention. (2004). Retrieved November 20, 2018 from www.micron.com
[20] Micron 2017. Calculating Memory Power for DDR4 SDRAM. (2017). Retrieved November 20, 2018 from www.micron.com
[21] Naveen Muralimanohar et al. 2009. *CACTI 6.0: A Tool to Model Large Caches.* Technical Report.
[22] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008). Retrieved November 20, 2018 from www.bitcoin.org
[23] Taku Ogura et al. 2011. A fast rewritable 90nm 512Mb NOR "B4-Flash" memory with 8F2 cell size. *Symposium on VLSI Circuits - Digest of Technical Papers* (2011).
[24] Colin Percival. 2009. Stronger key derivation via sequential memory-hard functions. (2009). Retrieved November 20, 2018 from www.tarsnap.com
[25] Marc Pilkington. 2016. Blockchain Technology: Principles and Applications. *Research handbook on digital transformations* (2016), 225.
[26] James E. Stine et al. 2007. FreePDK: An open-source variation-aware design kit. *IEEE International Conference on Microelectronic Systems Education* (2007).
[27] Synopsis. 2005. Design Compiler User Guide. (2005). Retrieved November 20, 2018 from beethoven.ee.ncku.edu.tw/testlab/course/VLSIdesign_course
[28] Michael B. Taylor. 2013. Bitcoin and the age of bespoke silicon. *CASES* (2013).
[29] John Tromp. 2015. Cuckoo cycle: A memory bound graph-theoretic proof-of-work. *Lecture Notes in Computer Science* 8976 (2015), 49–62.
[30] Youtube 2018. Bitmain Antminer E3 - Ethereum ASIC - Fist look and Tear down. (2018). Retrieved November 20, 2018 from www.youtube.com