NoC of Blockchain Enablers

Hyun-Bum Yang, Daniel Dai, Juliana Echternach, Shahin Nazarian

Blockchain has become a popular method of securing data is most popularly known by bitcoin and bitcoin mining. Blockchain performance is dependent on the hashing algorithm utilized. In this project, we will implement the SHAKE256 algorithm in hardware, and optimize the algorithm through multi-core parallelization with ParallelHash and pipelining to ensure maximum hardware performance in speed and power reduction. The goal of the project is to find areas for hardware optimizations in Verilog and parallelize our hashing through use of a multicore Network-on-Chip architecture.

I. INTRODUCTION AND MOTIVATION

Calculating the blockchain hash is highly dependent on the hashing algorithm. The bitcoin hashing algorithm for each block takes roughly ten minutes. There are many standard hashing algorithms, SHA256 being a popular one.

SHA3 is a new standard that is supposedly more secure than SHA1 and SHA2. Finding the speed increase in hardware can prove the speed-up of calculating blocks and in cryptocurrency mining; the comparative speed up will also prove the validity of SHA3 being more secure than its predecessors.

SHAKE256 is a variant of the SHA3 standard that is compatible with the ParallelHash algorithm that we will be using to parallelize the hashing. We were able to run ParallelHash successfully in software and are now looking to see how the algorithm will perform in hardware.

The bottlenecks existing in Blockchain concern hash algorithm speed and hardware performance. Large scale blockchains have run into issues of hardware cooling, resulting in large servers being built near rivers and glaciers. For example, Iceland has become a popular location for servers and bitcoin mining due to its abundance of glaciers and rivers. One way to try to allow for blockchain problems to scale large is to utilize hardware that consumes less power. Network-on-Chip architecture can utilize a multicore system that allows communication across cores and wires without the use of buses. This can be accomplished by taking a hashing string, and parsing it amongst many cores, allowing for a parallelization of the hash calculation. This has the potential to decrease hardware power consumption, which is a huge benefit to blockchain and bitcoin mining.

II. PREVIOUS WORK

There are several hashing functions that have been used in industry standard. MD-5 and SHA-1 have extensively been studied. However, it has been proven the MD-5 is prone to hashing collisions, which is something that needs to be avoided for large scale hashing [6]. SHA-1 has been continually updated in the algorithm with successive versions of SHA-2 and SHA-3. There have been several studies that analyze the hardware performance of SHA-3 type algorithms compared to SHA-2 type algorithms. The hashing function Keccak outperforms other SHA-3 and SHA-2 algorithms when compared in hardware due to the high throughput and parallelization [2]. The ParallelHash is a SHA-3 derivative hash by NIST that uses the cSHAKE hash [3].

Our work will attempt to implement the SHA-3 Keccak algorithm in hardware on an FPGA, and design a compact and efficient hardware core that can be utilized in a network on chip architecture. By utilizing the methods described in [2] we can learn how to implement SHA-3 on an FPGA, advance and pipeline this model so it can further increase hardware speed in hashing.

III. APPROACH AND PRELIMINARY RESULTS

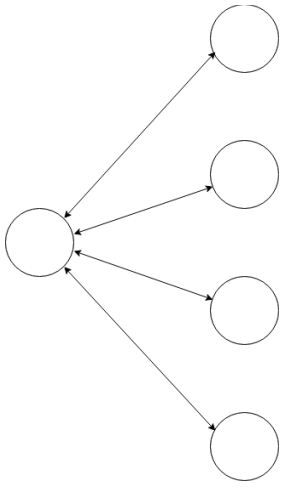

Figure 1. Core Communication Diagram

We have successfully ran the ParallelHash algorithm in Python and are searching for different bottleneck points that could be optimized rather in pipeline or parallelization. We have found that we can decompose a single hash into several with ParallelHash and are looking to create a state diagram that will implement that ParallelHash algorithm for one core, and then for many cores as shown in Fig. 1.

Running the algorithm on a chip requires implementation in a descriptive hardware language like Verilog. We are looking to implement the SHA3_256 algorithm, more specifically the SHAKE256 variant. Afterwards, we will investigate possible ways to pipeline and parallelize the hashing algorithm to for further optimization. We plan to check against a software implementation to compare and verify optimization results. We have already implemented the SHA3 algorithm in software, and are looking for points where we can potentially pipeline the algorithm in hardware. We will implement the code in Verilog, verify the generated hash with the software, or theoretical results. Once the Verilog code matches our software hash values, we will compare the performance results of the software implementation with the hardware implementation, and hopefully we can demonstrate how the hardware version greatly improve the algorithm performance.

After completing the hardware implementation, we plan to attempt to distribute the computational load across multiple cores through a NoC implementation to increase throughput while decreasing power consumption.

IV. CONCLUSION

As previously stated, the goal of this research is to find possible ways to improve the current Blockchain processes. Specifically, we are looking at how hashing is implemented in the process and how we can mitigate this bottleneck. We have successfully implemented the ParallelHash algorithm in software. The next steps in our project is to recreate the hashing with Verilog and measure the performance improvement of utilizing hardware acceleration. To do so, we will work closely with our software implementation to not only verify the validity of our hardware implementation but also to measure the performance values. We have identified the optimal areas for parallelization using an algorithm designed for this specific task. Going forward we will focus on finding the optimum performance of the ParallelHash with respect to the granularity. Next, we will apply these optimum parameters to our Network on Chip architecture.

V. REFERENCES

1.) Homsirikamol, Ekawat, Marcin Rogawski, and Kris Gaj. "Comparing hardware performance of round 3 SHA-3 candidates using multiple hardware architectures in Xilinx and Altera FPGAs." *Ecrypt II Hash Workshop*. Vol. 2011. 2011.

2.) Kerckhof, Stéphanie, et al. "Compact FPGA implementations of the five SHA-3 finalists." *International Conference on Smart Card Research and Advanced Applications*. Springer, Berlin, Heidelberg, 2011.

3.) Kelsey, Chang, Shu-jen Chang, Ray Perlner. "SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, ParallelHash" *NIST Special Publication* (2016): 800-185.

4.) Oder, Tobias, and Tim Güneysu. "Implementing the NewHope-Simple key exchange on low-cost FPGAs." *Progress in Cryptology–LATINCRYPT* 2017 (2017).

5.) Sakakibara, Yuma, et al. "A hardware-based caching system on FPGA NIC for Blockchain." *IEICE Transactions on Information and Systems* 101.5 (2018): 1350-1360.

6.) Wang, Zhenqi, and Lisha Cao. "Implementation and comparison of two hash algorithms." *Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on*. IEEE, 2013.

7.) Zaki, Mohammed J., and Ching-Jui Hsiao. "Efficient algorithms for mining closed itemsets and their lattice structure." *IEEE Transactions on Knowledge & Data Engineering* 4 (2005): 462-478.