ATLANTIS PRESS

Advances in Computer Science Research, volume 87

3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)

Optimization of Hash Function Implementation for Bitcoin Mining

Xiaohan Zhang [a], Honggang Hu[b]

Key Laboratory of Electromagnetic Space Information, CAS University of Science and Technology of China Hefei, 230027, China.

[a]zxh666@mail.ustc.edu.cn, [b]hghu2005@ustc.edu.cn

Abstract. As the origin of blockchain technology, Bitcoin has received a lot of attention in recent years. Hash function calculation is the key problem in proof-of-work mechanism. The main work of this paper focuses on the FPGA implementation and optimization of the hash function. Firstly, this paper determines iteration bound of the algorithm by using the iteration bound theory. Secondly, this paper uses carry-save adder and the retiming method to make the computation time of critical path equal to the iteration bound. After that, this paper uses the constant input part and the requirements of the result to further reduce the number of cycles of the hash function calculation. Finally, this paper performs simulation, and the results show that this implementation improves the efficiency of the hash function calculation.

Keywords: SHA256, bitcoin mining, FPGA.

1. Introduction

The success of Bitcoin is inseparable from its stable network consensus, and the most important step in maintaining consensus is called proof-of-work, in which the most tedious task is to repeat the SHA256 calculation. Hence, improving the efficiency of the SHA256 calculation can enable miners to achieve an advantage in the mining process. This paper mainly focuses on optimizing the efficiency of the SHA256 calculation in the process of Bitcoin mining.

Regarding the structure of the SHA256 algorithm, there has already been a sufficient amount of existing research on the optimization of its computational efficiency. L. Dadda et al. [1] proposed an optimization method using the carry-save adder; Ricardo Chaves et al.[2] reordered the SHA256 algorithm; Ito K et al.[3] proposed the concept of iteration bound after studying the loop bound; Yong Ki Lee et al.[4] analyzed the iteration bound of SHA256 in details; and Nicolas T. Courtois et al.[5] summarized the optimization methods for the SHA256 algorithm in Bitcoin mining applications, but they demand higher area consumption, which increases the mining cost.

In this paper, a new structure using the above-mentioned two optimization methods is designed for the SHA256 algorithm in the Bitcoin mining scenario, which improves its computational efficiency. What follows are the details: Firstly, the theoretical optimal efficiency of the structure is determined by the iteration bound analysis. Secondly, the carry-save adder, the retiming method and the unfold method are used to reach the bound value. Next, the constant inputs and difficulty requirements in the process of Bitcoin mining are used to reduce the number of cycles per SHA256 operation. Finally, a complete architecture is proposed to achieve the above-mentioned optimization. The corresponding simulation is carried out in this paper and the results show that the architecture implemented in this paper has a significant improvement in computational efficiency.

2. Hash Function Optimization

2.1 Iteration Bound Theory

Firstly, we analyze the determinants of the maximum frequency of the FPGA. The computation time between the register i and the register j is defined as t_{ij}. In order to guarantee that the registers are correctly flipped, the cycle length T must be longer than t_{ij}. Therefore, the maximum frequency in the FPGA is determined by:

$$Fmax = 1/Tmin = 1/max(tij) \quad (i \neq j) \tag{1}$$

The algorithm block diagram can be regarded as a directed graph, and the computation time of any of its path is expressed as t_{path} the critical path is defined as the path with the longest computation time and without any register. It is expressed as follows:

$$CriticalPath \in \{Path \mid t_{path} = max(t_{ij}) \} \tag{2}$$

The upper limit of the clock frequency is determined by the computation time of the critical path. Therefore, in order to improve the computational efficiency of the algorithm, it is necessary to reduce the computation time of the critical path.

The path in which the start point and the end point are the same is called a loop, and the loop is essentially a feedback. In order to analyze the influence of the loop on the maximum frequency, the loop bound is defined in:

$$T_l = t_l / w_l \tag{3}$$

Where, t_l refers to the total computation time of the loop, and w_l refers to the total number of registers in the loop.

The loop bound represents the maximum frequency that a loop can withstand. The maximum value of all loop bounds of the algorithm is called the iteration bound and defined in:

$$T_\infty = max\ (T_l) = max\ (t_l / w_l) \tag{4}$$

Under the premise of maintaining the same operation flow of the algorithm, the iteration bound is the lower bound of the theoretical cycle length, which determines the lower limit of the computation time of the critical path as well as the maximum frequency of the system.

2.2 The Analysis and Optimization of the SHA256 Iteration Bound

Firstly, the CSA(carry-save adder) and the retiming method are introduced. The CSA is a three-input two-output adder that retains the carry without adding it to the result. For continuous additions, the intermediate results are processed using CSA before they are calculated using the adder to produce the final result, which will effectively reduce the delay.

Retiming is based on the following principle: for any cut set in the graph, specify a particular direction between the subgraphs, and if a register is deleted for each edge in the same direction while a register is added to each edge in the opposite direction, then the algorithm result remains unchanged[6]. By iteratively using retiming, the computation time of the critical path can reach the iteration bound.

For the functions defined by the SHA256 algorithm, since they are all operations such as bitwise operations and bit shifts, we think that the computation time roughly meet the following:

$$T(\sigma_0) = T(\sigma_1) = T(\Sigma_0) = T(\Sigma_1) = T(Maj) = T(Ch) = T(CSA) = t_0 \tag{5}$$

$$T(Adder) = t_1 \gg t_0 \tag{6}$$

In the SHA256 compression algorithm, the iteration bound is the loop bound of the path from Register E to itself and its value is $3t_0+t_1$. We copy the computation time of the path where H→E and H→A are overlapped and place the operation summing it with Register D before summing it with Σ_1. After that, we use CSA instead of continuous additions and obtain its iteration bound $2t_0+t_1$. The retiming method is subsequently used to adjust the position of the registers so that the computation time of the critical path of the compression algorithm reaches its iteration bound. Its block diagram

is shown in Fig. 1. The CSA has two outputs which are indicated by two-headed arrows in the block diagram. The registers in the figure are represented by D, while the square nodes are not registers.

In the extension algorithm, the iteration bound is the loop bound of the path from w_{t+1} and w_{t+2}, and its value is t_0+t_1. Since it is smaller than the iteration bound of the compression algorithm, we only need to adjust the computation time of the critical path to be equal to that of the iteration bound. The result is shown in Fig. 2, in which w_t^* not a register.

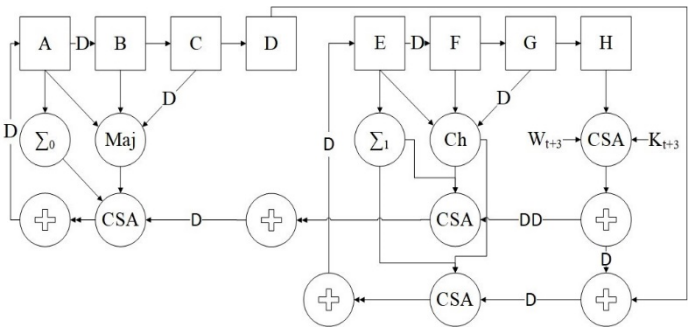

Fig 1. The Optimized SHA256 Compression Algorithm Block Diagram

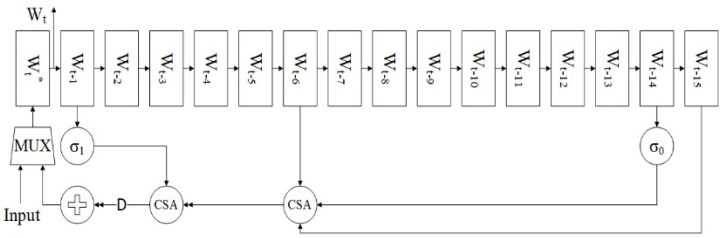

Fig 2. The Optimized SHA256 Extension Algorithm Block Diagram

It should be noted that the retiming operation causes a negative delay in the path from the initial vector input to Register A-H and the path from the registers to the output. In the optimized implementation, the initial vector changes to the following:

$$IV_t=A_t||B_t||C_{t+1}||D_{t+2}||E_{t+1}||F_{t+1}||G_{t+2}||H_{t+3} \tag{7}$$

The time difference between the vectors causes the requirement of additional cycles for the algorithm to complete all the work. Our solution requires 3 additional cycles to balance the effects of retiming.

3. Optimization Concerning Proof-of-Work

The Bitcoin's proof-of-work algorithm is closely related to the head structure of the block. The data structure of the block header is shown in TABLE 1.

The entire mining process, which is shown in Fig. 3, consists of three SHA256 calculation processes, denoted respectively as SHA#0, SHA#1, and SHA#2. Firstly, IV is used as the initial value and Data1 is used as the input to produce the intermediate value Midstate. Then Midstate is used as the initial vector and Data2 is used as the input for the second calculation to produce the output H1. Finally, IV is used as the initial value and H1 is used as the input for another calculation to output the final result H2.

Table 1. The Data Structure of the Block Header

Name	Version	PreHash	MerkleRoot	Time	nBits	Nonce
Bytes	4	32	32	4	4	4

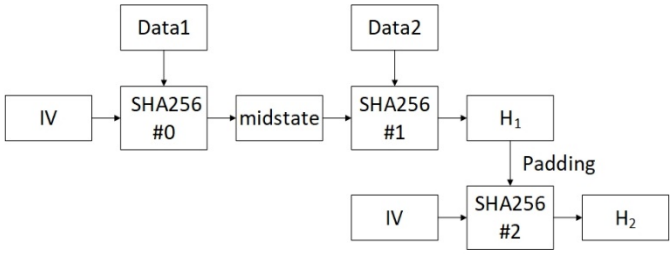

Fig 3. The Proof-of-work calculation Diagram

3.1 Optimization with the Constant Input Part

In the process of Bitcoin mining, Version, PreHash, and nBits have been determined before the block is generated. Although Time and Merkle Root will change during the block generation, the frequency of change is considerably low as compared with the hash function calculation. In other words, they can be considered as constant values during the mining process. Therefore, in most cases, SHA#0 needs to be calculated only once for each mining step. On the other hand, it is noted that SHA#1 processes 32-bit inputs in proper order in every round of the first 16 rounds. However, the inputs of the first 3 rounds remain constant for a long time, so it can start directly from the 4th round to reduce 3 cycles.

3.2 Optimization with the Requirements of the Result

A legal proof-of-work requires that its hash value be less than its difficulty value. The current Bitcoin difficulty has requirements for the first 96 bits of the hash value (the first 72 bits should be 0, and the last 24 bits should be less than the constant value) (Bitcoin's current block height is about 545,000, which comes from www.blockchain.com). According to the calculation process of SHA256, the 96^{th} to 128^{th} bit of the hash function can be observed in the Register A of the compression algorithm in the 61^{st} round; the 64^{th} to 95^{th} bit can be observed in the 62^{nd} round; and the 32^{th} to 63^{th} bit can be observed in the 63^{rd} round. Once a value that fails to meet the requirements is observed, the calculation can be withdrawn in advance without the need of further calculation in the subsequent rounds. With this method, at most 3 cycles of SHA256#2 can be reduced.

3.3 Overall Architecture

We integrate the above-mentioned optimization of proof-of-work into the optimized compression algorithm of Section 2.B. The new compression optimization structure is shown in Fig. 4, in which the square grid indicates the optimized positions corresponding to the original registers, and the rest are the same as described above. The initial vector enters the algorithm at different times through the multiplexer. When the round count reaches the requirement, the existing difficulty values are compared to determine if reset should be performed, and the result is output after the required number of rounds is reached.

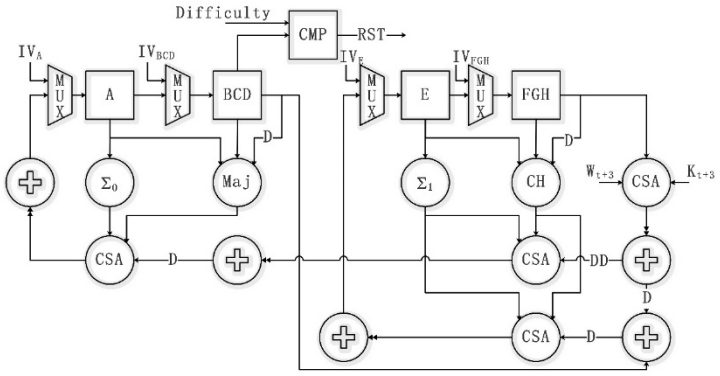

Fig 4. The Final Structure of the Compression Algorithm

4. Simulation Analysis

Intel's Stratix IV EP4SGX230KF40C2 was used for simulation with the Verilog language for code writing. In order to facilitate the comparison of simulation results, a complete hash function calculation was performed for analysis. The comparison between the simulation result of the SHA256 algorithm in this design and that from its predecessors' research is shown in TABLE 2.

Table 2. Comparison of Simulation Results

Implementation	**Results**			
	Chip Model	***Frequency (MHz)***	***Area***	***Throughput (Mbps)***
[2]	XC2V	121	1666 Slice	1534
[2]	XC2VP	141	1667 Slice	1780
[7]	Virtex 7	367	350 Slice	1470
[8]	Virtex 6	218.9	1301 ALUT	1660
This Paper	Stratix IV	309	1996 ALUT	2364

5. Conclusion

The iteration bound theory is detailed in this paper and used as the core to discuss the optimization of SHA256 on FPGA, and the corresponding block diagram is provided. Next, the computational efficiency is improved based on the characteristics of the Bitcoin mining scenario. Finally, the efficiency of the design is analyzed by the simulation experiment, whose result shows that the efficiency of the hash function calculation can be effectively improved at the smaller expense of areas.

References

[1]. Dadda, Luigi, Marco Macchetti, and Jeff Owen. "The design of a high speed ASIC unit for the hash function SHA-256 (384, 512)." Proceedings of the conference on Design, automation and test in Europe-Volume 3. IEEE Computer Society, 2004.

[2]. Chaves, Ricardo, et al. "Improving SHA-2 hardware implementations." International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2006.

[3]. Ito, Kazuhito, and Keshab K. Parhi. "Determining the minimum iteration period of an algorithm." Journal of VLSI signal processing systems for signal, image and video technology 11.3 (1995): 229-244.

[4]. Lee, Yong Ki, Herwin Chan, and Ingrid Verbauwhede. "Iteration bound analysis and throughput optimum architecture of SHA-256 (384,512) for hardware implementations." International Workshop on Information Security Applications. Springer, Berlin, Heidelberg, 2007.

[5]. Courtois, Nicolas T., Marek Grajek, and Rahul Naik. "Optimizing sha256 in bitcoin mining." International Conference on Cryptography and Security Systems. Springer, Berlin, Heidelberg, 2014.

[6]. Chao, Liang-Fang, and E. Hsing-Mean Sha. "Scheduling data-flow graphs via retiming and unfolding." IEEE Transactions on Parallel and Distributed Systems 8.12 (1997): 1259-1267.

[7]. Kahri, Fatma, et al. "An FPGA implementation and comparison of the SHA-256 and Blake-256." Sciences and Techniques of Automatic Control and Computer Engineering (STA), 2013 14th International Conference on. IEEE, 2013.

[8]. S binti Suhaili, Shamsiah, and Takahiro Watanabe. "Design of high-throughput SHA-256 hash function based on FPGA." Electrical Engineering and Informatics (ICEEI), 2017 6th International Conference on. IEEE, 2017.