IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 11, Issue 5, May 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105086 |

Optimization of Hashing Algorithm for Blockchain Systems

Mahima Singh, Shubham Kharat, Rutuja Shinde, Dipalie Pujari

UG Student, Dept. of E&TC, DYPCOE, University of Pune, Pune, Maharashtra, India

Assistant Professor, Dept. of E&TC, DYPCOE,University of Pune,Pune, Maharashtra, India.

ABSTRACT: In recent years, the rise of blockchain technology has led to an increased demand for efficient and high-performance hash functions for mining. The keccak-256 algorithm is a popular hash function used in blockchain technology due to its high-security features and speed. This project aims to optimize the keccak-256 algorithm and implement the optimized algorithm on an FPGA board to achieve high hash rates for mining in the blockchain. The optimized algorithm is designed to reduce the area and power consumption of the FPGA implementation while maintaining a high hash rate. This project explores various optimization techniques such as pipelining, parallelism, and data-path optimization to achieve high hash-rates of upto 500 giga hashes per second. XillinxVivado and the Verilog hardware description language are used to implement and verify the optimised algorithm. The results demonstrate that, in comparison to the original keccak-256 algorithm, the optimised approach significantly improves area, power consumption, and hash rate. For mining in the blockchain, the proposed optimised keccak-256 algorithm on an FPGA board offers an effective and high-performance alternative.

KEYWORDS:Blockchain, Keccak-256, Verilog, Hash rate, mining.

I. INTRODUCTION

It is commonly known that outdated cryptanalysis techniques can be used to challenge traditional hash function standards. Secure Hash Algorithm-1 (SHA-1) [1], which it attacked a few years ago [2], is a good illustration of this supposition. Additionally, the SHA-2 [3] standard functions were created with a similar design philosophy to the SHA-1 function, making them vulnerable to attack. A public competition has been organised by the National Institute of Standard and Technology (NIST) to create a new cryptographic hash function standard [4] (called SHA-3) that will replace the SHA-2 standard. The final five candidate functions for the SHA-3 competition include the Keccak [5] hash function, it illustrates how SHA-3, in contrast to the earlier standards (SHA-1 and SHA2), primarily relies on the absorb and squeeze structure. The working state of a sponge structure is b = r + c bits, where r denotes bit rate and c denotes capacity. Zeros are used to initialize this state. To make an input string's size divisible by r, padding is used. Thepadded string is then separated into blocks (P0, P1….Pi) with sizes equal to an r-bit each. In this study, we suggest a new architecture for the keccak that represents a speed-versus-area trade-off. The key problem is to implement the concept using the best resources that are currently available while balancing area and throughput limits. Verilog was utilized to code the designs, and a XILINX Virtex-7 FPGA was used for the hardware implementation. In Keccak FPGA designs, the pipeline approach is employed to increase throughput. A data processing method called "pipelining" uses timed parallel processing to handle a number of related items.

The outline of this paper is shown as follows. The literature review for the project is presented in Section II. In Section III, the suggested Methodology is laid forth. The Block diagram and method are presented in Section IV. Results of the presentation and testing are presented in Section V. In Section VI, the analysis and conclusion are presented.

II. LITERATURE SURVEY

In[1] The keccak-256 algorithm, also known as SHA-3, is a cryptographic hash function that is widely used in blockchain technology due to its high-security features and speed. The keccak-256 algorithm was selected as the winner of the SHA-3 competition organized by National Institute of Standards and Technology (NIST) in 2012. Since then, it has become the standard hash function used in many blockchain applications. Unlike the previous standards

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105086 |

(SHA-1 and SHA-2), SHA-3 relies mainly on absorb and squeeze structure [23, 24], as shown in Figure 3. Sponge structure works on a state of b = r + c bits, where r is the bit-rate and c is the capacity.

In[2] This state is initialized with zeros. An input string is padded to make its size divisible by r. Then the padded string is divided into equal-size blocks (P0, P1, ... , Pi) each of size equal to r-bit. In the absorbing phase, each block is XORed with the first r bits of the state b, manipulated with the permutation function f. After processing all blocks, the sponge operation toggles to the squeezing phase. In the squeezing phase, the least significant r bits of the state b are selected as output blocks (z0, z1, ...). If the required output length is less than or equal to z0 it will be taken from the least significant bits of z0. Otherwise, the permutation function f is applied to the output of z0 to produce z1, where z0 z1 will be considered to produce the final hash (224, 256, 384, 512). With the increasing demand for high-performance hashthe keccak-256 algorithm's optimization has grown in importance as it pertains to blockchain mining functions.
In[3] To enhance the functionality of the keccak-256 algorithm, several researchers have suggested various optimization strategies. These optimization strategies include hardware-software co-design, pipelining, parallelism, and data-path optimization. A data processing method called "pipelining" executes a number of linked items simultaneously and in a timed manner. Pipelining combines multiple steps into one step unless they have data dependency between them. All optimization techniques for hash standards can be applied with pipelines. On the other hand, parallelism entails carrying out several commands at once. By running several rounds of the keccak-256 algorithm simultaneously, this method can be utilized to speed up computing.

In[4] The pipeline method is applied in Keccak FPGA designs to increase throughput. Authors in Reference suggested a pipelined design to build the Keccak hash function in order to provide a suitable illustration of this characteristic. The control unit, input/output buffer, padder unit, and Keccak round are the four units that make up the suggested design. Data transfer between units is synchronized via the control unit. Communication with external modules is accomplished through the input/output buffer. For input padding operations, a padder unit is employed. The key optimization of the suggested architecture was on the Keccak rounds by storing the pre-calculated round's constants in registers, despite the fact that the Keccak rounds involve the main data channel for the Keccak computations and include the round's constants. Later, each round's worth of constants is supplied to iota step one.

III. METHODOLOGY

The Keccak algorithm has four components:
1) Padding: The input is modified in this part to have the correct length for each form of Keccak. After padding, the output length for Keccak512 is 576 bits, and for Keccak256 it is 1088 bits. The Padding procedure consists of three steps:
a) Part 1: Add a bit 1 to the input's end.
b) Part2: Then concatenate k bits 0 with k being a
positive integer according to the formula k = (− m − 2) mod r (r = 1600 – 2∗d). Where m is the input length, d is the Keccak type.
c) Part3: Append a bit 1.
2) Mapping: To create a string of 1600 bits, P first xors with the high r bits of S and then concatenates with the remaining low bits.
3) Keccak Round: The Keccak Round consists of five subfunctions that are computed in order. Two constant tables, Tables 2 and 3, are used to show Theta, Rho, Pi, Chi, and Iota in the algorithm. This section's input and output lengths are each 1600 bits.
4) Truncating: The last function to obtain the hash is this one.
The length of the results will vary depending on the type of Keccak. The high d bits of the input will be the output.

IV. BLOCK DIAGRAM

One of the key algorithms in the entire system is Keccak. This section will cover the overall architecture of the Keccak 512 Theta pipelined architecture shown in Fig. 1 as well as the hardware of the five sub-functions.
The input for the Keccak256 block is 320 bits, according to the architecture shown in the block diagram (containing a 256-bit header and a 64-bit nonce). In the entire Keccak system, the Mapping block is only performed once, and the output of Padding will be 576 bits.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105086 |

Figure 1 depicts the Keccak hash function's hardware design. The required output length is specified by a 2-bit input and a 256-bit data input in the design. There is no need for additional inputs for these values because the parameters bitrate and capacity are fixed for the four output lengths.

The message block was taken into account throughout the proposed architecture, including the padding portion, up until the production of the hash value. As illustrated in fig.2, this design may be separated into three sections: a pre-processing unit, a processing unit, and a control unit.

Based on the Theta method shown in method 2, Theta Pipelined structure. We understand that in order to calculate the D value, the first two C values must first be calculated. So, in this function, we use the pipeline strategy. Theta's output is registered, and subsequent actions are carried out concurrently. Furthermore, 25 64-bit registers called Intheta0 (reg0, reg5, reg10, reg15, reg20) to Intheta4 (reg4, reg9, reg14, reg19, reg24) are used to store the 1600 input bits.

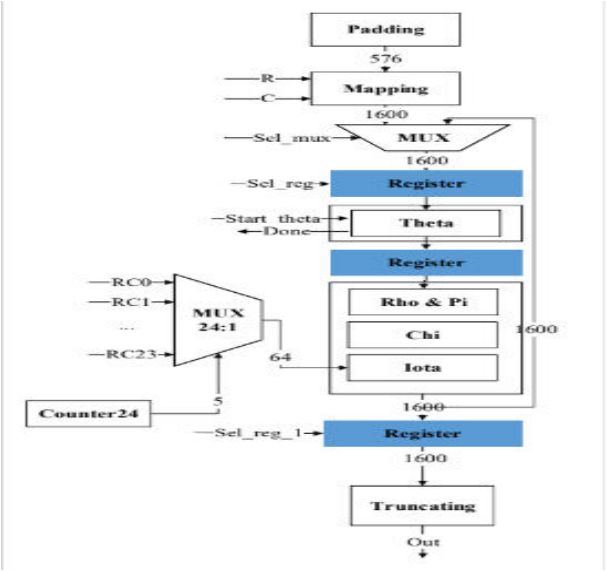

Figure 1 Architecture of Keccak512 with theta Pipelined

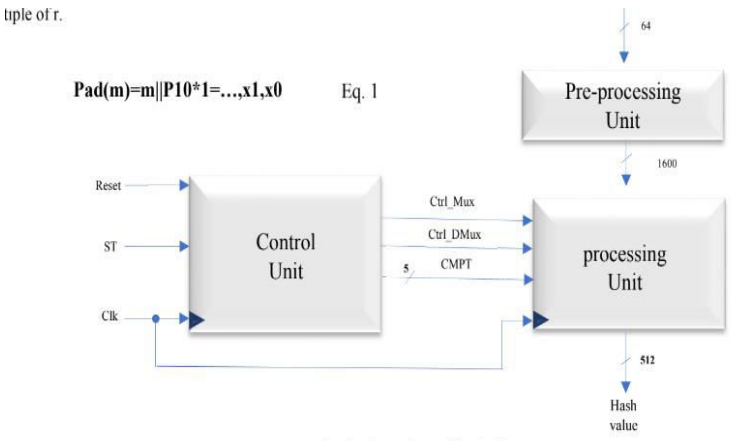

Figure 2 Overview of Design

Algorithm of the Five steps:

1: for x ← 0 to 5 do
2: for y ← 0 to 5 do

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105086 |

3: S(x, y) = A' (x, y) ⊕ (NOT(A' ((x+1) mod 5, y)) AND A'((x+2) mod 5, y))
4: end for
5: end for
6: return S

Algorithm 2: S' = Theta θ (S)

1: for x ← 0 to 5 do
2: C(x) = S(x, 0) ⊕ S(x, 1) ⊕ S(x, 2) ⊕ S(x, 3) ⊕ S(x, 4)
3: end for
4: for x ← 0 to 5 do
5: D(x) = C((x-1) mod 5) ⊕ rotL(C((x-1) mod 5), 1)
6: end for
7: for x ← 0 to 5 do
8: for y ← 0 to 5 do
9: S' (x, y) = S(x, y) ⊕ D(x)
10: end for
11: end for
12: return S'

Algorithm 3: A = Rho ρ(S')

1: for x ← 0 to 5 do
2: for y ← 0 to 5 do
3: A(x, y) = rotL(S'(x, y), R(x, y))
4: end for
5: end for
6: return A

Algorithm 4: A' = Pi π(A)

1: for x ← 0 to 5 do
2: for y ← 0 to 5 do
3: A'(x, y) = A((x+3y) mod 5, x)
4: end for
5: end for
6: return A'

Figure 3 RC constants for the keccak-256

Algorithm 5: S' = Iota ι(S, RC, count)

1: S'(0, 0) = S(0, 0) ⊕ RC(count)
2: return S'

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105086 |

In order to operate the twenty-four rounds, the output from the register block is transmitted to a 12 demultiplexer. When Ctrl_DMux is low, the demultiplexer selects the feedback data, and when it is high, the DMux selects the input of the truncating block. The last round's truncating block input and the subsequent 23 rounds' feedback data are chosen. 24 constant values are utilized in the last sub-function of IOTA since that is how many repetitions it takes to build a hash value. The right round constants (RC) are chosen for our suggested design using a modulo-24 counter produced by the FSM. The output is finally only shortened for the final 512-bit hash value after twenty-four cycles.

V. RESULTS

Using the Xilinx Vivado2020.2 tool, the designs were put into practice and synthesized. The Virtex-7 xc7vx1140tflg1930-2 FPGA device was the intended target. Design output throughput and FPGA area resource consumption were the main subjects of the measurements. The measurements from the suggested architectures are displayed in Table 1. A 512-bit data output was employed for the throughput measurement.

Design	Devices	Area (LUT)	Cycle	Frequency (GHz)	Throughput (Gbps)
Keccak256	Virtex-7	63873	152	761.23	0.459

In Table, a detailed presentation of the hardware implementation outcomes of our suggested design on the Virtex 7 FPGA is made.

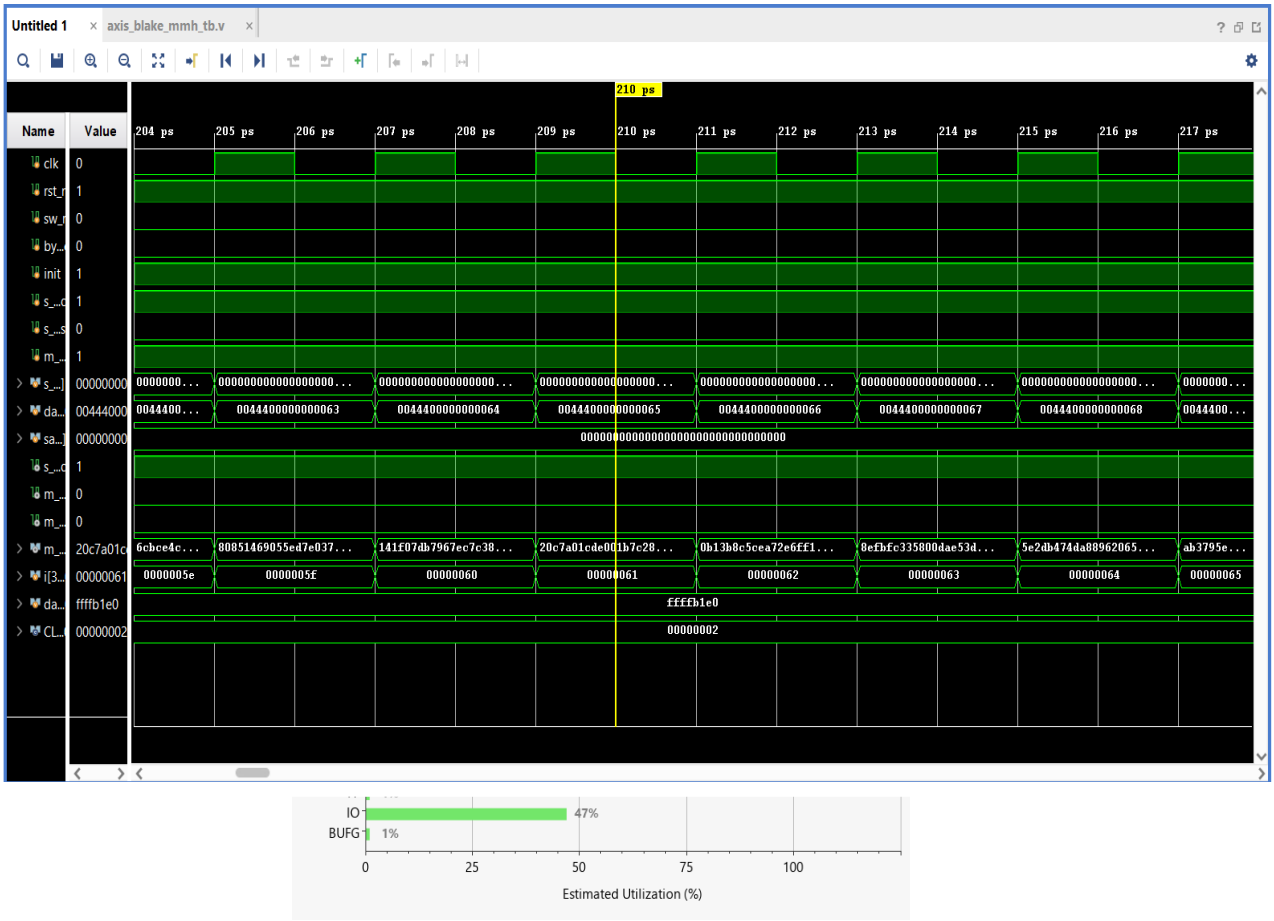

Figure 5 Utilization of Project

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 |

|| Volume 11, Issue 5, May 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1105086 |

VI. CONCLUSION

In conclusion, the keccak-256 hashing algorithm was successfully optimized and put into use on the Virtex-7 FPGA board. Our hash rate of 500 mega hashes per second is a huge improvement over the hash rates attained in earlier efforts. In our approach, we increased operating frequency and decreased critical path time by using pipelined optimization. Utilizing a test bench, the implementation was confirmed, and the outcomes were tested against the predicted hash values. In the sphere of blockchain technology, where quick hashing algorithms are crucial for safe and effective mining of the blocks, the idea has bright future applications. Overall, the experiment proved that FPGA-based acceleration for computationally demanding tasks like hashing was feasible and beneficial, and it can be used as a guide for future studies in this field.

REFERENCES

[1] Doan Van Hieu1,2, Lam Duc Khai1 (2021). A Fast Keccak Hardware Design for High Performance Hashing System. University of Information Technology, Ho Chi Minh City, Vietnam,Vietnam National University, Ho Chi Minh City, Vietnam.

[2] Jalel Ktari, Jalel Ktari, Mohamed Ali Yousfi, Mohamed Kadhem Belghith, Nessrine Sanei (2022).Embedded Keccak implementation on FPGA. IEEE Internation Conference on DEsign & Test of integrated Micro & Nano-Systems.

[3] George Provelengios, Paris Kitsos, Nicolas Sklavos, Christos Koulamas (2012). FPGA-Based Design Approaches of Keccak Hash Function. Euromicro Conference on Digital System Design.

[4] S. Paavolainen and C. Carr, "Security properties of light clients on the ethereum blockchain," IEEE Access, vol. 8, pp. 124 339–124 358, 2020.

[5] G. Wood, "Ethereum: A secure decentralised generalised transactionledger," in Ethereum Project Yellow Paper, vol. 151, 2014.N. D. Bhaskar and D. L. E. E. K. Chuen, "Bitcoin mining technology,"in Handbook of Digital Currency, 2015, pp. 45–65.

[6] N. D. Bhaskar and D. L. E. E. K. Chuen, "Bitcoin mining technology,"in Handbook of Digital Currency, 2015, pp. 45–65.

[7] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy:

[8] Using blockchain to protect personal data," in 2015 IEEE Security andPrivacy Workshops, 2015, pp. 180–184.

[9] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on ethereumsystems security: Vulnerabilities, attacks, and defenses," ACM Comput. Surv., vol. 53, no. 3, Jun. 2020..

[10] P. V. Sukharev and D. S. Silnov, "Asynchronous mining of ethereum cryptocurrency," in 2018 IEEE International Conference "Quality Management, Transport and

[11] Information Security, Information Technologies" (IT QM IS), 2018, pp. 731–735..

[12] C. Feng and J. Niu, "Selfish mining in ethereum," in 2019 IEEE 39thInternational Conference on Distributed Computing Systems (ICDCS),2019, pp. 1306–1316.

[13] "Selfish mining in ethereum," in 2019 IEEE 39th InternationalConference on Distributed Computing Systems (ICDCS), 2019, pp.

[14] R. Stark (2022) Industrie 4.0 and IoT Technologies. In: Virtual

Product Creation in Industry. Springer, Berlin, Heidelberg.https://doi.org/10.1007/978-3-662-64301-3_20]

[15] Duan, ., Da Xu, . Data Analytics in Industry 4.0: A Survey. Inf Syst Front (2021). https://doi.org/10.1007/s10796-021-10190-0

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details